Contribution à la surveillance distribuée des systèmes à événements discrets complexes

Amine Boufaied

Thèse

présentée devant
L'Université Paul Sabatier (sciences)

en vue de l'obtention du
Doctorat de l'Université Paul Sabatier de Toulouse

Spécialité : **Systèmes Industriels**

par
Amine BOUFAIED
Ingénieur en Informatique

Contribution à la Surveillance Distribuée Des Systèmes à Evénements Discrets Complexes

---

Soutenue le 18 Décembre 2003 devant le jury :

Président	**L. TRAVE-MASSUYES**
Rapporteurs	**E. CRAYE**
	B. DESCOTES-GENON
Examinateurs	**M. COMBACAU**
	P. BERRUET
	L. TRAVE-MASSUYES
Directeur de thèse	**A. SUBIAS**

Avant-Propos

Le travail réalisé dans ce mémoire de thèse a été effectué au sein du groupe de recherche DIagnostic, Supervision et COnduite qualitatifs (DISCO) du Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS) du Centre National de la Recherche Scientifique (CNRS) à Toulouse. J'exprime ici ma profonde gratitude à Monsieur Malik Ghallab, directeur du LAAS, de m'avoir accueilli dans son laboratoire ainsi qu'à tout le cadre administratif et personnel en particulier, Christophe Benazeth, Julien Libourel et Christian Berty.

Je remercie également Monsieur Joseph Aguilar Martin Directeur de recherche CNRS et responsable du groupe DIagnostic, Supervision et COnduite qualitatifs pour m'avoir accueilli dans son groupe.

Je remercie Mme Louise Trave-Massuyes, directeur de recherche CNRS, de l'honneur qu'elle m'a fait en acceptant de présider ce jury de thèse.

Je tiens à exprimer ma gratitude également à :

- Monsieur Etienne Craye, Professeur des Universités à L'école Centrale de Lille
- Bernard Descote-Genon, Professeur des Universités à l'IUT 1, Grenoble,
- Pascal Berruet, Maître de conférences, Université Bretagne Sud
- Michel Combacau, Professeur à l'Université Paul Sabatier
- Audine SUBIAS, Maître de conférences, INSA de Toulouse.

qui m'ont honoré en apportant leur compétence pour l'évaluation de ce travail et en acceptant de participer au jury de ma thèse, et à M. E. Craye et B. Descote-Genon qui ont accepté d'être rapporteurs et contribué à l'amélioration de ce mémoire par leurs critiques constructives.

Je voudrais témoigner toute ma reconnaissance à Audine Subias et à Michel Combacau pour m'avoir orienté et encouragé dans mes travaux de recherche et pour les conseils qu'ils m'ont apporté tout au long de la thèse. Leur soutien, leur confiance, leur disponibilité, leurs critiques et leur sympathie m'ont été d'une grande utilité. Pour tout cela, je leur dis merci.

Enfin, je dédie ce travail à tous ceux qui m'ont soutenu et aidé pendant cette période et à tous les membres des groupes DISCO et MOGISA. J'adresse une pensée particulière à Robert Valette, Abd-El-Kader Sahraoui et Boutaib Dahhou.

Table des matières

Introduction

La surveillance n'améliore pas seulement les performances et la productivité d'un système complexe mais aussi protége les vies et les biens. Pour ces raisons, des approches de surveillance ont été largement étudiées dans la littérature. La plupart des approches de surveillance dans la littérature ont été développées pour des systèmes où l'information utilisée par la surveillance est centralisée ou hiérarchique. Or, la majorité des systèmes complexes (réseaux de communication, systèmes manufacturiers, systèmes de puissance, etc.) sont informationnellement distribués.

Récemment, l'attention s'est portée sur la distribution de la commande et de la surveillance de systèmes de natures diverses. Dans un contexte économique devenu austère, la distribution tend à doter la commande et la surveillance de caractéristiques leur permettant de palier aux carences existant au niveaux des architectures conventionnelles (centralisée, hiérarchique…). La conséquence la plus visible est une amélioration de la disponibilité et de la fiabilité du système de commande-surveillance, d'où une augmentation des performances et des gains de l'entreprise. Plus particulièrement, *la détection*, comme étant une fonction de la surveillance ayant pour rôle de détecter toute violation des spécifications de bon fonctionnement du système surveillé, doit être distribuée sur plusieurs sous-systèmes de détection communicants et coopérants. Cette fonction, comme toutes les fonctions de surveillance, a besoin d'informations concernant l'état du procédé. Cette information provient des capteurs renvoyant des événements relatifs au déroulement des opérations réalisées. L'intégration d'un modèle du procédé dont l'état est en permanence remis à jour par les évolutions provoquées par la commande et les signaux émis par les capteurs est nécessaire pour vérifier les contraintes structurelles régissant le fonctionnement du procédé. Les contraintes temporelles peuvent, quant à elles, être exprimées à l'aide d'un modèle temporel distribué tout en respectant la structure distribuée de la détection.

La distribution nécessite la résolution de problèmes jusque là non traités. La décomposition du système physique en sous-systèmes doit être spécifiée, les délais de communication et les délais opératoires doivent être pris en compte dans les mécanismes de détection et enfin, une modélisation de tous les mécanismes proposés doit être faite avec un outil adéquat.

Le travail que nous présentons dans ce document propose d'apporter sa contribution au domaine de la surveillance en temps réel des systèmes à événements discrets complexes et plus particulièrement au niveau de la fonction détection. Il s'intéresse à la distribution des mécanismes de détection. Ainsi, la détection peut être locale à un site de détection ou distribuée mettant en jeu une communication et une coopération entre différents sites de détection. L'accent est mis sur la distribution du modèle temporel compte tenu de la distribution physique des éléments du procédé. L'approche proposée est basée sur un échange d'occurrences d'événements permettant aux sites de reconnaître des sous-modèles temporels qui leur sont assignés. La non reconnaissance d'un sous-modèle traduit l'apparition d'un symptôme de défaillance au niveau du procédé. L'application des mécanismes développés est réalisée sur un système de transport représentant *un terminal intermodal* que nous nous proposons de surveiller de manière distribuée.

Dans le premier chapitre de cette thèse, nous allons présenter les différentes fonctions de supervision-surveillance et plus particulièrement la fonction détection. Ensuite, nous présenterons l'évolution des architectures de commande-surveillance. Enfin, l'architecture distribuée sera décrite ainsi que les solutions apportées par une telle architecture vis à vis des problèmes rencontrés dans les architectures conventionnelles.

Dans le deuxième chapitre, nous allons exposer les concepts de base de la distribution du système de surveillance-commande. La surveillance distribuée y sera abordée à travers la présentation de différentes approches et d'architectures pour la surveillance existantes dans la littérature.

Dans le troisième chapitre, nous allons nous intéresser à la distribution de la fonction détection avec prise en compte des délais. Ces délais peuvent être des délais de communication entre sous-systèmes de détection ou des délais représentant des durées opératoires. En se basant sur la décomposition physique effectuée au niveau du procédé, le modèle temporel de comportement, que nous appellerons chronique, va être distribué sur plusieurs sous-systèmes de détection. Cette méthode de distribution, les protocoles de communication et de coopération, et les mécanismes de détection vont être spécifiés dans ce chapitre.

Dans le quatrième chapitre, nous présenterons les principes de modélisation d'une chronique par réseaux de Petri p-t-temporels. La distribution de la chronique nous permettra d'obtenir un ensemble de sous-chroniques communicantes et coopérantes. Les sous-chroniques seront obtenues à partir d'un modèle réseau de Petri représentant une chronique, ou à partir de modèles réseaux de Petri représentant les contraintes temporelles composant les sous-chroniques. Nous allons étendre les réseaux de Petri p-t-temporels aux réseaux de Petri p-t-temporels flous afin de modéliser les mécanismes associés à la reconnaissance d'une sous-chronique dans le cas où un délai existe ou pas.

Dans le cinquième chapitre, nous allons traiter le cas d'un terminal de fret intermodal dans lequel seuls des conteneurs sont manipulés. Une architecture de surveillance distribuée sera développée se basant sur un ensemble de sous-chroniques représentant une portion du plan opératoire journalier du terminal. Les mécanismes de communication et de coopération entre les différents superviseurs seront mis en évidence. Les résultats obtenus permettront la détection distribuée d'un retard dans le plan opératoire et l'inachèvement de certaines opérations.

Chapitre I

Cadre de l'étude

I.1. Introduction

Durant les trente dernières années, plusieurs types d'architectures de commande-surveillance ont été adoptées pour supporter des systèmes de natures diverses (de production automatique, informatiques, médicaux…). La surveillance, composée de plusieurs fonctions (détection, diagnostic pronostic, décision, reprise, etc.) a été généralement implantée sur la base d'une architecture centralisée ou hiérarchique.
L'architecture distribuée permet de résoudre les problèmes associés à ces architectures conventionnelles. De nouvelles architectures pour la conception de systèmes complexes comme les architectures holarchiques, bioniques, fractales se basent sur une structure distribuée. Une architecture distribuée de surveillance est une architecture qui repose sur un ensemble d'entités autonomes et coopérantes.

La première partie de ce chapitre est une présentation générale des différentes fonctions de supervision-surveillance et plus particulièrement de la fonction détection. Dans la deuxième partie de ce chapitre, nous présentons l'évolution des architectures de commande-surveillance. Ceci nous permettra de dégager les avantages et les inconvénients de chacune afin de pouvoir les comparer. La troisième partie est consacrée à la description de l'architecture distribuée ainsi qu'aux solutions apportées par une telle architecture vis à vis des problèmes rencontrés dans les architectures conventionnelles.

I.2. Concepts de base et définitions

Nous présentons ici quelques définitions prises de (Combacau et al., 2000) et de (Patton et al 1999). Il existe dans la littérature de nombreuses autres définitions des termes présentés par la suite. Celles que nous considérons sont issues d'un travail de réflexion mené dans le cadre du Groupement de Recherche en Productique, auquel participent la plupart des équipes de recherche françaises travaillant dans les domaines de la supervision, surveillance, commande.

- *Commande* : elle déclenche l'exécution d'un ensemble d'opérations en donnant des ordres aux actionneurs. Il peut s'agir :

- d'un ensemble d'opérations correspondant à une séquence d'opérations pour la réalisation d'un produit ou d'un service.
- d'un ensemble d'opérations exécutées afin de restaurer la fonctionnalité du procédé offerte durant l'exécution normale.
- d'actions avec un degré élevé de priorité engagées afin de protéger les opérateurs humains et prévenir les évolutions catastrophiques.
- d'opérations de test, de réglage ou de maintenance exécutées afin de garder le procédé dans son état opérationnel.

La définition de la commande inclut toutes les fonctions agissant sur le procédé comme nous le verrons au § I.3.

- *Surveillance* : elle collecte les données du procédé et du système de commande. Elle détermine l'état actuel du système contrôlé et produit les inférences nécessaires à la production de données additionnelles (historique, détection, diagnostic, etc.). La surveillance est limitée au traitement des données et ne possède pas d'actions directes sur les modèles ou sur le procédé.
- *Supervision* : elle calcule et met à jour les paramètres de la séquence de commande à exécuter en prenant compte de l'état du système de commande et de l'état du procédé. Elle inclut les opérations normales et anormales (inattendues).
  - Durant une opération normale, la supervision prend des décisions pour lever l'indécision dans le système de commande (ordonnancement temps-réel, optimisation, mise à jour de la commande, et remplacement d'une loi de commande par une autre).
  - Quand un symptôme de défaillance est identifié, la supervision prend toutes les décisions nécessaires pour permettre au système de reprendre son fonctionnement normal (ré-ordonnancement, actions de recouvrement, procédures d'urgence, etc.).

Quelques termes supplémentaires que nous serons amenés à utiliser par la suite doivent être définis. Certains d'entre eux peuvent être trouvés dans (Laprie 1992).

- *Faute* : action, volontaire ou pas, ne prenant pas en compte toutes les spécifications.
- *Défaut* : différence entre la valeur actuelle et nominale d'un paramètre.
- *Erreur* : une partie du modèle ne correspondant pas aux spécifications du système physique et/ou logique. Une erreur est la conséquence d'une faute.
- *Erreur latente* : une erreur est qualifiée de latente aussi longtemps que la partie erronée du modèle n'a pas été utilisée. Après utilisation de la partie erronée du modèle, l'erreur devient effective.
- *Défaillance* : un événement caractérisant une situation dans laquelle une opération n'est pas exécutée par une ressource parce que son état ne correspond pas aux spécifications nominales.
- *Etat d'échec* : un état d'une ressource à partir duquel le système ne peut pas fournir le service spécifié. Cet état est une conséquence d'une défaillance.
- *Symptôme* : un événement ou des données à partir desquels le système de détection identifie une opération anormale du procédé. Le symptôme est la seule information connue par le système de surveillance au stade de la détection.
- *Disponibilité* : c'est l'aptitude d'une entité à être en état d'accomplir une fonction requise, dans des conditions données, à un instant donné, en supposant que la fourniture des moyens extérieures nécessaires est assurée.

- *Sécurité* : c'est l'aptitude d'une entité à éviter de faire apparaître, dans des conditions données, des événements critiques ou catastrophiques. Ces événements peuvent être critiques pour l'opérateur, le système ou son environnement.
- *Fiabilité* : c'est l'aptitude d'une entité à accomplir une fonction requise, dans des conditions données, pendant un intervalle de temps donné. La fiabilité traduit une notion de continuité de fonctionnement.

Grâce à ce qui a été défini précédemment, nous pouvons maintenant présenter les différentes fonctions de supervision-surveillance.

I.3. Les fonctions de supervision-surveillance

Dans ce paragraphe, nous définissons les fonctions élémentaires du système de supervision et de surveillance. Les lettres M, S ou C indiquent à quel système précédemment étudié (surveillance (M), Supervision (S), Commande (C)) est rattachée la fonction (Combacau et al 2000).

- *Détection (M)* : elle détermine la normalité ou l'anormalité du système en fonctionnement. Deux classes d'opérations anormales sont considérées :
  - la première inclut les situations dans lesquelles les contraintes opérationnelles du procédé sont violées (par exemple les situations de collisions),
  - la deuxième regroupe les situations dans lesquelles la loi de commande n'est pas respectée (par exemple, les situations de retards dus à des délais de fabrication trop long).
- *Suivi (M)* : il maintient un historique des traitements exécutés et une trace des événements observés par le système de commande-supervision.
- *Diagnostic (M)* : il cherche une causalité liant le symptôme, la défaillance et son origine. Classiquement, trois sous-fonctions sont distinguées :
  - localisation : détermine le sous-système responsable de la défaillance,
  - identification : identifie les causes de la défaillance,
  - explication : justifie les conclusions.
- *Pronostic (M)* : le but de la fonction pronostic est de déterminer les conséquences d'une défaillance sur le fonctionnement futur du système. Deux types de pronostic peuvent être distingués (Boufaied et Combacau 2001). Le premier, appelé *pronostic préliminaire*, cherche les conséquences inévitables d'une défaillance (identification de l'ensemble des tâches qui ne peuvent plus être exécutées en respectant l'ordonnancement). Le deuxième, appelé *pronostic préventif*, est centré sur l'erreur latente et sur la propagation de défaillance (la répétition de la même défaillance). Ce type de pronostic est utilisé afin d'éviter de l'exécution d'activités menant à la détection du même symptôme.
  - Pronostic préliminaire :
    à ce stade du traitement correctif, seules la défaillances et ses causes sont connues. Afin de prévoir les effets de la faute, trois types d'information sont prises en compte :
    - l'ordonnancement des tâches à exécuter sur la ressource défectueuse,
    - l'analyse hors ligne des conséquences de la défaillance (AMDE pour l'Analyse des Modes de Défaillance et de leurs Effets) (Villemur 1998),
    - les données statistiques concernant l'état d'échec de la ressource (MTTR pour Mean Time To Repair).
  - Pronostic préventif : seules les défaillances dont les effets ne sont pas immédiatement détectés sont considérées dans ce cas. En effet, une propagation

d'erreur peut avoir lieu suivant le routage d'un produit défectueux. Le pronostic préventif détermine s'il y a d'autres produits qui sont affectés par le même défaut qui induirait le même symptôme dans le futur.

- *Décision (S)* : elle détermine l'état qui doit être atteint afin de continuer l'opération normale, ensuite elle détermine la séquence d'actions correctives à réaliser pour atteindre cet état.
- *Recouvrement (C, S)* : c'est une fonction intervenant après l'occurrence d'une panne (détectée et diagnostiquée). Il détermine un nouvel état du système, nécessitant un ensemble ordonné d'actions correctives, qui vont modifier le comportement du procédé et de la commande, afin d'assurer la sécurité et la disponibilité du système (Berruet et al 1998a). Trois types d'action peuvent être distinguées :
  - l'arrêt complet du système et réparation de la ressource en panne,
  - la marche dégradée de la ressource affectée par la défaillance,
  - la reconfiguration du système.
- *Reconfiguration (C, S)* : elle agit sur le procédé pour changer l'état de la ressource ou de l'équipement et sur le système de commande en changeant les lois de la commande, la gamme d'une pièce, etc. (Berruet et al 1998b). Trois classes de reconfiguration peuvent être définies :
  - Mineure, seules les lois de commande sont adaptées,
  - Significative, d'autres ressources sont réallouées,
  - Majeure, les ressources réallouées ont besoin d'être préparées pour exécuter la reconfiguration.

Nous nous intéressons par la suite à la fonction *détection* comme étant la première fonction sollicitée dans un processus de surveillance. En effet, les fonctions de surveillance citées précédemment s'intègrent dans le processus de surveillance dans un ordre dépendant de la situation de défaillance et permettant au système, suite à l'apparition d'un symptôme de défaillance, de reprendre un état de fonctionnement normal. Cet ordre n'est pas limité à la séquence classique (détection, diagnostic, décision et reprise) (Zamai et al 1998). Les fonctions de surveillance sont associées entre-elles pour constituer des « activités de surveillance » qui peuvent s'enchaîner selon plusieurs ordres possibles. La fonction détection est chargée de détecter l'occurrence d'un symptôme de défaillance, par conséquent, elle est à la base de toute activité de surveillance et initie tout le processus de surveillance.

Nous présentons cette fonction dans un cadre général indépendant d'une structure ou d'une implémentation particulière.

I.4. La fonction détection

La surveillance d'un système complexe doit permettre de traiter tous les comportements qui s'écartent du comportement attendu en déterminant l'état actuel du système. Cette fonction est un des principaux éléments d'un système de surveillance (Valette et al 1994). Dans un scénario simpliste, la détection (Holloway et al 1990) (Combacau 1991) (Cruette 1991) (Toguyeni 1996) de symptômes de défaillance peut être une observation par un opérateur que le système ne fonctionne pas correctement. Des techniques plus avancées se basent sur la détection « rapide » d'un symptôme de défaillance afin que les mécanismes de recouvrement ou de reconfiguration puissent être exécutés ou que le système soit arrêté pour éviter une propagation des dommages. Ces techniques s'appuient sur les dates limites d'occurrence des signaux attendus, le suivi de l'évolution de l'état du système, les systèmes experts, l'utilisation de capteurs spécialisés et les techniques d'analyse de fréquence. Toutes ces techniques se basent sur l'information envoyée par les capteurs. L'ajout de nombreux capteurs dédiés à la surveillance n'est pas toujours possible pour des raisons techniques

et de coûts. Mais surtout, il est désormais largement admis que dans les systèmes à événements discrets (SED), les éléments tombant le plus souvent en panne sont les capteurs. De ce fait, si leur ajout augmente la sécurité, par contre le système de surveillance perd beaucoup de sa disponibilité. Cette thèse traite en partie l'utilisation de l'information non envoyée par les capteurs dans les mécanismes de détection.

Les défaillances dans les systèmes peuvent être divisées en deux classes :

1) les défaillances liées à la commande incluant des bogues logiciels : l'application des techniques de tolérance aux fautes permet de limiter considérablement le nombre de défaillances dues au système de commande. Ceci techniques seront explicitées ultérieurement dans le cadre d'une architecture de commande-surveillance distribuée.

2) les défaillances physiques d'éléments du procédé.

La détection de symptômes de défaillance liés aux éléments du procédé requiert généralement l'élaboration d'un modèle du système à surveiller (procédé). Ce modèle peut être un modèle de bon fonctionnement ou un modèle de dysfonctionnement. Par exemple, dans le cas de systèmes discrets, un modèle correspondrait à un réseau de Petri, et dans le cas de systèmes continus, un modèle correspondrait à un ensemble d'équations différentielles. Il existe aussi un certain nombre de méthodes de détection qui n'utilisent pas de modèles, c'est à dire qu'il n'est pas fait appel à une description du système surveillé. Il s'agit essentiellement d'outils statistiques, qui permettent des tests sur des grandeurs caractéristiques du système.
Dans le cas où un modèle est utilisé, deux mécanismes de base sont utilisés pour la détection. Le premier consiste à comparer les évolutions du système observé avec celles d'un modèle du procédé ou de signatures (séquences datées d'événements) de bon fonctionnement qui évoluent de façon synchrone, c'est à dire en temps-réel avec le système. Ce modèle est intégré dans le système de commande-surveillance. Le second consiste à rechercher des signatures connues de défaillances. Ces signatures décrivent des défaillances passées ou théoriques connues du procédé ou d'éléments du procédé. Sans modèle du système à surveiller, la stratégie adoptée consiste en l'exploitation des informations données par les capteurs et les détecteurs au niveau local du procédé.

I.4.1. L'approche modèle du procédé

Le modèle du procédé a été conçu à des fins de surveillance et plus particulièrement de détection. Toutes les contraintes liées à la structure physique du procédé sont rejetées dans ce modèle. L'état du modèle du procédé est mis à jour en permanence par les évolutions provoquées par la commande et par les signaux émis par les capteurs appelés comptes rendus. Ces comptes rendus sont envoyés par le système commandé en réponse à une requête de commande. Le mécanisme qui a été développé par (Combacau et al 1991) pour la détection de symptôme de défaillances est basé sur l'étude de l'impact de comptes rendus sur le modèle du procédé, appelé aussi modèle de référence, et celui de commande. Toutes les commandes et les sorties des capteurs sont des entrées du modèle. Lorsqu'une requête de commande est envoyée vers le système commandé, le système de détection se met en attente du compte rendu correspondant spécifié dans le modèle de commande. Lorsque ce dernier arrive, il est naturellement comparé à celui attendu par le modèle de commande. Trois situations peuvent être distinguées :

- le compte rendu traduit une évolution possible du modèle de commande. Dans ce cas, aucun symptôme de défaillance n'est détecté,

- le compte rendu est rejeté par le modèle de commande mais traduit cependant une évolution du modèle du procédé. Ceci traduit le fait que le système commandé n'a pas réalisé l'objectif de la commande, mais n'a cependant pas transgressé de contraintes physiques. La fonction détection déclare alors l'apparition d'un symptôme de défaillance.
- le compte rendu ne traduit aucune évolution possible, ni dans le modèle de commande, ni dans le modèle du procédé. La fonction détection déclare alors l'apparition d'un symptôme de défaillance.

L'exploitation conjointe de ces deux modèles ne permet pas de caractériser tous les symptômes de défaillance du système commandé comme par exemple l'absence de comptes rendus. Ceci peut être le cas par exemple lorsqu'un capteur est hors service ou que la requête de commande ne s'est pas exécutée. Afin de prendre en compte ces problèmes, (Combacau et al 1991) a proposé d'intégrer des mécanismes de « chiens de garde » aux modèles de commande et de procédé. Dans le cadre de la détection de symptômes de défaillance du procédé, ces mécanismes s'appuient sur deux dates fournies par l'ordonnancement : *la date de début au plus tôt* notée ΔT_m et *la date de fin au plus tard* notée ΔT_M d'une opération de commande. En effet, à toute opération A_i, à laquelle correspond un compte rendu CR_i, est associée une fenêtre temporelle d'exécution $[\Delta T_m, \Delta T_M]$. CR_i est valide uniquement à l'intérieur de cette fenêtre. ΔT_m et ΔT_M sont des dates relatives à l'instant de déclenchement de A_i. Cet instant est appelé « *start-event* » (Toguyeni 1992) et est considéré comme un événement déclencheur du processus de détection.
Deux symptômes peuvent être distingués compte tenu de la fenêtre temporelle bornée par ces deux dates et associée à une opération de commande :

- occurrence d'un compte rendu avant la date de fin au plus tôt,
- absence de réception du compte rendu. Ceci est systématiquement détecté par le « chien de garde » utilisant la date de fin au plus tard de l'opération. Le « chien de garde » est activé au commencement de l'opération de commande et peut être désactivé avant un temps maximum donné à partir de la date de fin au plus tard. S'il n'est pas désactivé avant ce temps un symptôme de défaillance est détecté.

L'approche de détection proposée dans cette thèse se base sur la fenêtre temporelle définie à partir de l'ordonnancement des opérations de commande. Elle se base donc sur l'analyse temporelle des symptômes de défaillance (Azua 2002).

I.4.2. L'approche signature de défaillance

Dans cette approche, la détection du symptôme de défaillance est réalisée essentiellement par comparaison avec des descriptions abstraites de ce qui est considéré comme une signature de défaillance (Toguyeni 1992). Cette approche tente de mettre en forme des règles qui décrivent les comportements non désirés du procédé, en s'appuyant sur des symptômes de défaillance passées ou des faiblesses connues du système (Dorothy 1987). Les règles peuvent être faites pour reconnaître un seul événement synonyme de défaillance du système ou une signature d'événements débouchant sur une défaillance. L'efficacité de cette détection repose sur la couverture de tous les symptômes de défaillance possibles par les règles.

La mise en œuvre de cette approche se base généralement sur deux techniques :

- la première consiste à coder les signatures de défaillance avec des règles d'implication *si…alors*. Il est alors possible de rentrer de nouvelles règles pour détecter de nouveaux symptômes de défaillance,

- la seconde utilise un modèle d'état-transition pour représenter les signatures de défaillance dans lequel l'état initial n'est pas un état d'échec. Le système exécute alors une série d'actions qui provoquent des évolutions du modèle pouvant conduire vers des états pour lesquels le système est considéré défaillant.

Nous pouvons rapprocher les méthodes utilisées à celles que l'on peut rencontrer dans le domaine de l'intrusion de virus dans les systèmes informatiques, où l'on recherche la signature de certains programmes dans les fichiers du système informatique, ou encore dans le domaine de la génomique où l'on recherche une séquence d'ADN (Acide DésoxyriboNucléique) dans un brin, ou plus généralement dans tout domaine approchant le problème de l'appariement de signatures (Denning 1987).

Cette approche présente certains inconvénients dont nous citons :

- la difficulté de construire une base de signatures. En effet, il est inconcevable, même pour un expert, de prévoir à l'avance tous les symptômes de défaillance possibles,

- la non détection de symptômes de défaillance non prévus. Comme la base des signatures développée ne peut pas être complète, certains symptômes de défaillance ne peuvent pas être détectés. Ceci peut avoir des conséquences dramatiques sur le fonctionnement du système,

- le risque de fausses alarmes dues à l'imprécision de la signature de défaillance. Ceci englobe le cas où les signatures de défaillance sont mal définies, et par conséquent mal exprimées. Cette imprécision peut aussi être due à la variation du contexte lié au procédé dans lequel les signatures ont été définies. La variation du contexte est relative à des données dynamiques (vitesse d'exécution, nombre de produits…) du système.

I.4.3. Outils statistiques pour la détection

Les outils statistiques de détection consistent à supposer que les signaux fournis par les capteurs possèdent certaines propriétés statistiques. De nombreux tests permettent de vérifier si ces propriétés sont présentent dans l'échantillon des signaux mesurés. Le signal mesuré est considéré comme étant une variable aléatoire (Isermann 1984). Le test le plus simple qui puisse être envisagé est la comparaison des variables observées (continues, discrètes, symboliques ou non, …) avec des valeurs limites fixées à l'avance. Cela revient à supposer que la grandeur mesurée traduit un fonctionnement normal du système lorsque sa valeur de mesure est comprise dans certaines limites. Deux types de seuils peuvent être utilisés : le *seuil de prévention* dont le dépassement indique l'apparition probable d'un symptôme de défaillance, et le *seuil d'anomalie* au delà duquel l'évolution des variables est considérée comme anormale. Ces limites sont usuellement choisies de façon à être suffisamment distantes de l'apparition d'une défaillance tout en maintenant un faible taux de fausses alarmes (dépassements qui ne correspondent pas à des défaillances). Une autre technique de test consiste en une *fenêtre d'observation*. Une fenêtre d'observation est constituée par un ensemble fini de mesures successives des grandeurs du procédé sur un certain horizon temporel. Les tests statistiques effectués sur ces fenêtres consistent à vérifier les propriétés statistiques que les signaux observés dans cette fenêtre sont supposés posséder.

La fonction détection est, comme nous l'avons défini une fonction de surveillance. Elle doit par conséquent faire partie d'une structure de commande-surveillance particulière. Plusieurs

architectures de commande-surveillance sont décrites dans la littérature. Le paragraphe suivant donne un aperçu de leur l'évolution au cours de ces dernières années.

I.5. L'évolution des architectures de commande-surveillance

Les avancées technologiques rapides ont permis de considérer un large spectre de possibilités pour la conception des architectures de surveillance-commande. Cinq structures de base sont généralement identifiées allant de l'architecture centralisée à l'architecture hétérarchique ou distribuée, (figure 1).

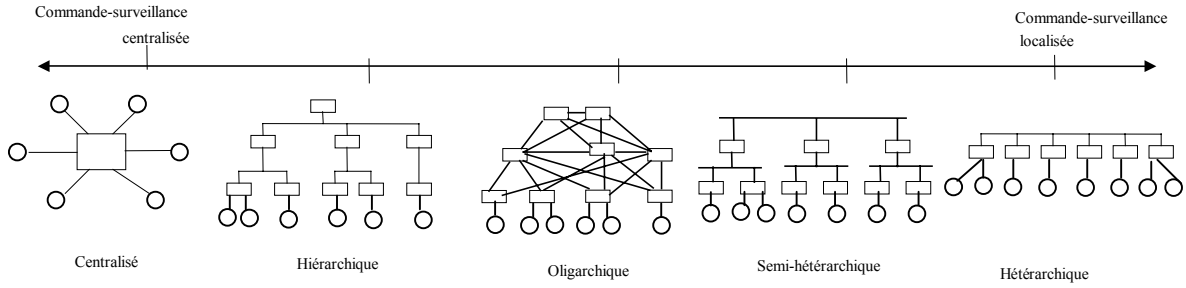


Figure 1. *Architectures de commande-surveillance.*

Chacune de ces structures est composée d'un ensembles de modules ou nœuds connectés entre-eux et liés aux entités physiques et/ou logiques du système (robots, machines, processus etc.). Nous donnons dans cette section une brève description de ces différentes architectures :

- La structure *centralisée* (Rana et al 1988) est caractérisée par une unité centrale (mainframe) puissante qui réalise la planification, le traitement de l'information et la mise à jour d'une base de données globale. Elle présente l'inconvénient d'être peu fiable en cas de défaillance et difficilement modifiable (Dilts et al 1991). Ceci est dû d'un côté, au fait que le système de commande-surveillance est implémenté sur une seule unité. Toute panne de cette unité entraîne la paralysie des fonctions de surveillance et de commande. Dans le cas de systèmes critiques comme la commande-surveillance d'un réacteur nucléaire ou d'un système embarqué, la paralysie peut avoir des conséquences catastrophiques. D'un autre côté, les besoins d'extension et de modification sont difficilement satisfaits dans une telle structure vu la grande complexité du système de commande-surveillance.

- La structure *hiérarchique* (Jones et al 1986) (Combacau et al 1990) (Chaillet-Subias 1995) a été utilisée pour la conception d'une architecture de commande-surveillance modulaire et hiérarchique. La complexité des systèmes basés sur des structures hiérarchiques croit rapidement avec la taille générant ainsi des coûts élevés de développement, d'installation, d'exécution, de maintenance, de surveillance et de modification ou de modernisation. En effet, de tels systèmes doivent prendre en compte des changements rapides des technologies et de l'environnement socio-économique. Il est aussi difficile d'introduire les mécanismes de tolérance aux fautes (voir § I.6.3) dans les structures hiérarchiques sans pour autant augmenter de manière significative la complexité du système. Dans ce type de structure, l'autonomie des différents nœuds de commande-surveillance est très limitée. En effet, la défaillance d'un nœud ou d'une connexion affecte le fonctionnement des nœuds des niveaux plus bas.

- (Piche et al 1983) a suggéré une structure *oligarchique ou hiérarchique modifiée* dans laquelle le niveau le plus haut est composé d'un nombre relativement petit de coordinateurs communicant entre eux et où les modules d'un même niveau peuvent communiquer. Cette structure a été créée dans le but de palier aux carences de la structure hiérarchique. Elle partage pourtant plusieurs des caractéristiques de l'architecture hiérarchique mais se distingue par un degré d'autonomie plus élevé des niveaux inférieurs. Cette structure présente tous les avantages de l'architecture hiérarchique. Par contre, l'augmentation de l'autonomie implique le traitement de plus d'information par le niveau inférieur. Cette structure présente aussi la plupart des inconvénients de la structure hiérarchique.

- La structure *semi-hétérarchique* est une structure hiérarchique composée d'un niveau supérieur formé d'un ensemble de coordinateurs et d'un niveau inférieur formé par des modules de commande-surveillance locaux. Ces modules communiquent seulement avec les coordinateurs auxquels ils sont rattachés. En effet, le besoin de coordonner les activités de commande et de surveillance entre les différents modules nécessite une prise de décision sur la base de l'information disponible localement au niveau des modules. Cette prise de décision est effectuée par les coordinateurs. La centralisation de la décision au niveau des coordinateurs donne lieu à un certain nombre d'inconvénients qui ont été présentés dans le cas d'une structure centralisée. En plus, cette structure présente tous les inconvénients de la structure hiérarchique (Duffie et al 1996).

- Les nouvelles avancées dans les domaines de la puissance du calcul distribué et des réseaux de communication (plus particulièrement les réseaux locaux) ont ouvert la voie aux structures *hétérarchiques ou distribuées* (Rana et al 1988). Les structures hétérarchiques ou distribuées (Dilts et al 1991) (Lin et al 1992) sont une alternative prometteuse permettant de trouver des solutions aux problèmes cités.

Les avancées technologiques dans les domaines du calcul distribué et des communications par réseau sont aussi à l'origine de l'intérêt croissant accordé aux architectures distribuées. Récemment, l'utilisation des architectures distribuées, pour la conception de systèmes de commande, de surveillance, etc., a permis d'améliorer la fiabilité, la disponibilité, l'autonomie, etc. du système considéré. La notion centrale d'une architecture distribuée est la considération d'entités totalement autonomes et coopératives permettant ainsi une prise de décision distribuée.

Nous nous intéressons plus particulièrement dans la prochaine section, à la description détaillée de l'architecture hétérarchique.

I.6. Les architectures hétérarchiques ou distribuées

Le terme hétérarchique est utilisé pour caractériser des systèmes à caractère local comme le montre la figure 1, (Duffie et al 1988) (Duffie et al 1996). En effet, ces systèmes collectent de l'information locale et agissent sur une partie du système contrôlé. Ces structures distribuées sont des structures coopératives regroupant des membres ou entités ou sites, ayant chacun un rôle à jouer et communiquant entre eux afin de réaliser un but global. (Hatvany 1985) suggère « une architecture distribuée coopérative » dans laquelle et en dépit de l'absence d'entités de haut niveau, chaque entité doit se conformer à certaines règles afin d'obtenir certains privilèges. (Duffie et al 1988) a proposé une structure distribuée de commande dans laquelle les relations entre entités ne suivent pas le modèle maître/esclave.

Les architectures distribuées offrent un certain nombre d'avantages dont :

- la réduction de la complexité et l'amélioration de la tolérance aux fautes (voir §I.6.3) grâce à la localisation de l'information ;
- la réduction des coûts de développement des logiciels grâce à la suppression des niveaux supérieurs ;
- l'amélioration de la maintenabilité, de la modifiabilité/extensibilité, de la reconfigurabilité/adaptibilité et de la disponibilité/tolérance aux fautes grâce à l'introduction de la modularité et de l'autonomie. La maintenabilité englobe une maintenance facilitée du système. Un système est dit modifiable si des changements au niveau des éléments du système peuvent être facilement effectués. Un système est extensible si de nouveaux éléments peuvent être facilement ajoutés au système afin d'augmenter ses fonctionnalités (Dilts et al 91). La reconfiguration du système est souvent nécessaire lorsqu'il s'agit d'ajouter ou de supprimer des composants du système quand il est en cours de fonctionnement, c'est à dire en ligne ;
- le maintien de la robustesse et de la flexibilité.

En abordant le problème de la caractérisation de systèmes sur la base d'une architecture distribuée, (Duffie 1990) propose de décomposer les besoins fonctionnels du système en un ensemble d'entités quasi-indépendantes et communicantes. Le paragraphe suivant explique une manière de caractériser des systèmes distribués à base d'entités.

I.6.1. Caractérisation de systèmes distribués à base d'entités

L'adoption de l'approche distribuée pose la question de la manière de décomposer le système en entités, d'éliminer l'information globale, de définir le rôle de l'opérateur dans une telle structure et les mécanismes de tolérance aux fautes à insérer. Nous présentons dans cette section les principes proposés par (Duffie 1990) pour la caractérisation de systèmes distribués à base d'entités. Cette approche, bien qu'elle ne soit pas la seule pour la résolution de ce problème, présente l'avantage d'être générale et indépendante de la nature du système considéré.

Ces principes sont décrits comme suit :

- il y a une décomposition naturelle associée au système ;
- le résultat de la décomposition est un ensemble d'entités quasi-indépendantes avec des interactions relativement faibles ;
- toutes les communications entre les entités prennent la forme de messages transmis sur un réseau de communication ;
- la configuration physique du système doit être transparente aux entités, et les entités ne doivent pas avoir besoin de connaître les localisations des autres entités.

Afin de doter les entités obtenues par la décomposition des caractéristiques recherchées pour les systèmes distribués, l'ensemble des principes généraux suivants (Duffie et al 1988) peut permettre d'obtenir des entités autonomes, coopérantes, tolérantes aux fautes et modifiables :

- les relations maître/esclave doivent disparaître ;
- les entités doivent coopérer quand c'est possible ;
- les entités doivent retarder l'établissement de relations avec d'autres entités le plus longtemps possible et les terminer le plus tôt possible ;

- l'information générée par une entité doit être maintenue localement et communiquée sur demande aux autres entités.

Les entités, comme elles ont été définies possèdent des droits et obligations vis à vis des autres entités. Une synthèse des principaux droits et obligations des entités est donnée par le fait qu'elles ont :

- des droits égaux d'accès aux ressources ;
- un accès mutuel et une accessibilité égaux à chaque entité ;
- des modes opératoires indépendants. Quand ces entités représentent des objets (produit, tâche) devant subir des opérations par des entités ressources, leurs opérations sont indépendantes ;
- une conformité stricte aux contraintes du système global. Les entités doivent assurer le respect des contraintes de fonctionnement global. La violation de celles-ci entraîne l'apparition d'un symptôme de défaillance.

Dans le paragraphe suivant, nous présentons un exemple d'un système manufacturier basé sur une architecture distribuée.

I.6.2. Exemple d'un système distribué sur la base d'entités

Nous présentons un système de production automatique de moules pour la fabrication de pièces en plastique, (figure 2). Le système doit accomplir les fonctions suivantes :

- conception assistée par ordinateur de pièces en plastique et de la géométrie des moules ;
- usinage des cavités des moules ;
- finition des cavités des moules ;
- inspection de la géométrie de la cavité du moule ;
- inspection de la géométrie de la pièce en plastique ;
- modification de la géométrie de la cavité du moule.

Les principes de caractérisation présentés dans la section précédente sont appliqués pour la décomposition du système en un ensemble d'entités fonctionnelles hétérogènes et communicantes. Ces entités englobent :

- les entités machine (fraiseuse, nettoyage, inspection, etc.) ;
- les entités de manipulation de matériel (robot, déménageur de palette) ;
- l'entité CAD / CAM ;
- l'entité base de données de la géométrie ;
- l'entité opérateur ;
- les entités des stations d'entrée et de sortie.

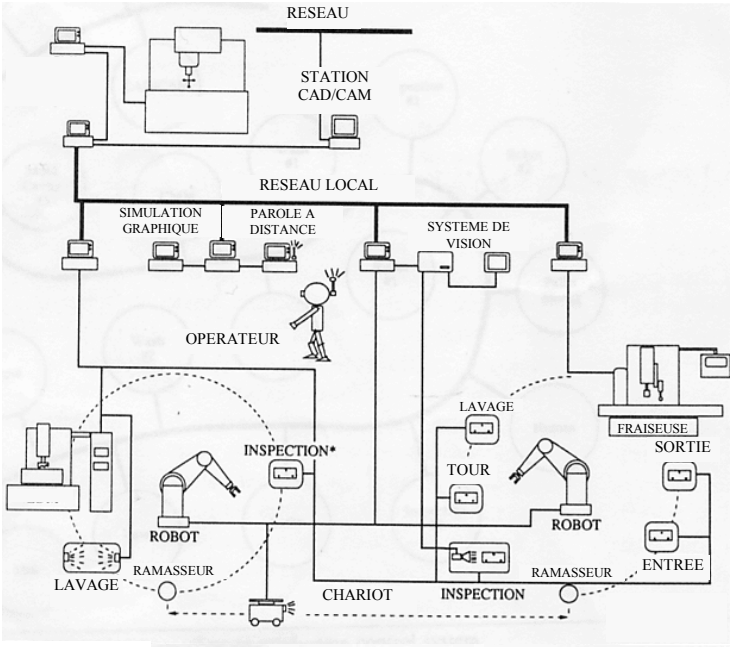


Figure 2. *Système de production expérimental de moule*.

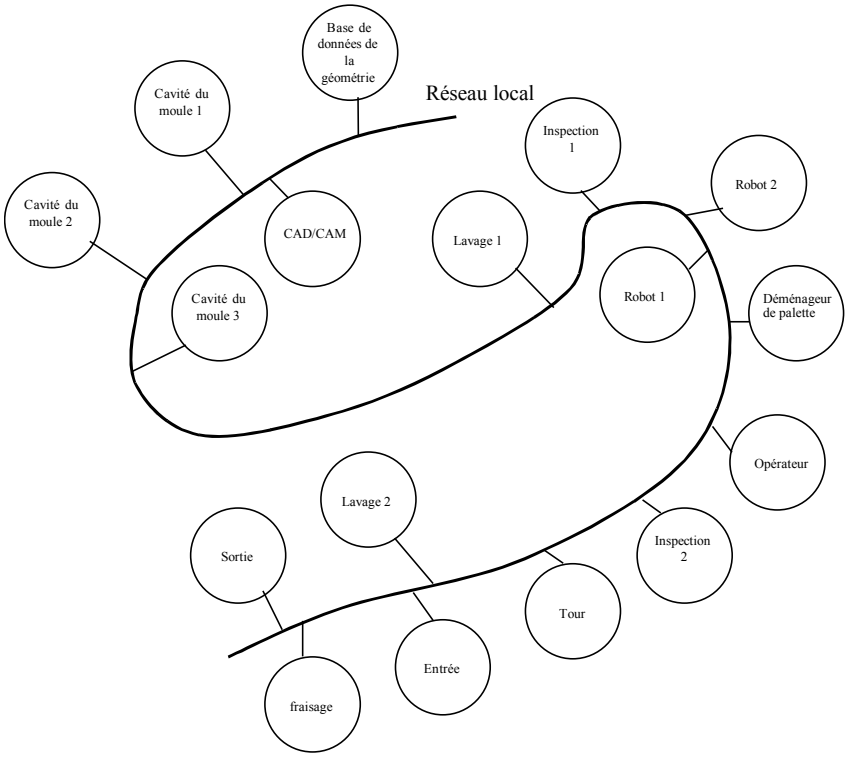


Figure 3. *Les entités d'un système expérimental de production de moules.*

Les entités communiquent entre elles à travers le réseau de communication. Elles peuvent résider sur n'importe quel processeur dans un système multi-processeurs, mais de préférence sur le processeur ayant des interfaces avec les machines ou autres dispositifs commandés par l'entité. La

connexion physique entre les entités est représentée sur la figure 3. Cette représentation ne considère pas la localisation physique de l'entité, sa fonctionnalité et le processeur sur lequel elle réside.
Les entités de ce système interagissent via un réseau de communication local sauf l'entité « Opérateur » qui utilise des modes de communication particuliers. En effet, l'entité « Opérateur » est intégrée dans le système comme une entité capable de converser avec les autres entités via la reconnaissance vocale, la synthèse vocale, les communications sans fil, et d'observer l'état des autres entités afin de donner des recommandations qui peuvent être suivies ou pas par les autres entités.

Après avoir présenté les principes de base des architectures distribuées, nous allons maintenant nous intéresser à un des principaux avantages des ces architectures qui est les mécanismes de tolérance aux fautes dans les systèmes de commande-surveillance distribués à base d'entités.

I.6.3. La tolérance aux fautes dans un système distribué à base d'entités

La tolérance aux fautes indique la capacité du système à continuer à fonctionner, éventuellement dans un état dégradé, malgré l'occurrence de défaillances logicielles. Elle inclut la détection d'un symptôme de défaillance logicielle, son diagnostic et la détermination des actions appropriées de recouvrement.
Dans la mesure où il n'est pas possible d'identifier toutes les situations de défaillance, il est impossible de définir tous les mécanismes de recouvrement nécessaires pour faire face à toute situation de défaillance du système de commande-surveillance. Dans ces conditions, une structure distribuée présente l'avantage d'améliorer la tolérance aux fautes du logiciel grâce à l'élimination de l'information globale et permet ainsi de réaliser :

- le confinement des fautes dans les entités,
- le recouvrement des fautes dans d'autres entités,
- la réduction de la complexité,
- la réduction du coût de développement.

De manière générale, un système de surveillance-commande composé d'entités fonctionnant de manière autonome perd certaines de ses fonctionnalités mais pas toutes lorsqu'une ou plusieurs de ses entités sont défaillantes. Les symptômes de défaillance peuvent avoir plusieurs sources dont la défaillance du réseau de communication, défaillance du processeur supportant l'entité, un problème de synchronisation entre entités ou encore des erreurs dans la logique de programmation des entités. Ces symptômes peuvent être détectés quand les messages attendus ne sont pas reçus par les entités ou sont reçus au mauvais moment, ou quand ces messages n'ont pas le bon contenu. Des mécanismes de surveillance de défaillances liées à des bogues logiciels doivent donc être réalisés au sein de toute entité.

Dans un système de surveillance distribuée à base d'entités plusieurs classes de symptômes nécessitent l'exécution des mécanismes de surveillance de manière complètement distribuée ou sous la conduite d'une ou de plusieurs entités. Les actions entreprises localement par les entités doivent être globalement cohérentes (Cammarata et al 1983). La tolérance aux fautes se trouve ainsi améliorée puisque la défaillance d'une entité participant à la résolution d'un problème distribué de surveillance n'influe pas considérablement sur le résultat obtenu, contrairement au cas ou une seule entité est dédiée à l'exécution des mécanismes de surveillance.

La résolution distribuée de problèmes par des systèmes à base d'entités nécessite une communication et une interaction entre les différentes entités. Les échanges de messages entre entités sont souvent structurés selon des schémas typiques appelés *protocoles d'interaction,* auxquels nous allons nous intéresser maintenant.

I.6.4. Les protocoles d'interaction dans un système distribué à base d'entités

Les protocoles d'interaction permettent de décrire explicitement les enchaînements conversationnels lors des communications entre les entités. Ils représentent un modèle de conversation commun à toutes les entités et décrivent comment les entités doivent réagir aux messages reçus durant les interactions. Trois types de protocoles d'interaction peuvent être distingués : les protocoles de coordination, de coopération et de négociation.

- *Les protocoles de coordination*, sont utilisés dans un environnement à ressources limitées. La coordination se traduit par un comportement des entités en vue de satisfaire leur propre intérêt ainsi que le but global du système. Une coordination peut, par exemple être réalisée pour un usinage de pièce, un transport de pièces, une exécution de tâches, etc. Le protocole de coordination le plus connu est le *Contract Net Protocol* (CNP) (Smith 1980). Afin d'illustrer les mécanismes de ce protocole dans le cadre d'un système manufacturier, nous considérons le cas d'une entité pièce qui doit progresser d'une station d'entrée vers une station de fraisage. Ce mouvement doit être accompli par un robot. Quand cette pièce arrive à la station d'entrée, la première étape consiste à diffuser un message demandant une opération de fraisage. Chaque entité de type *machine de fraisage* libre pour usiner cette pièce envoie un message de réponse à l'entité *pièce*, spécifiant sa disponibilité. L'entité *pièce* renvoie alors un message de réservation à une entité ayant répondu. Si plusieurs réponses sont reçues de plusieurs entités de fraisage toutes sauf une seront ignorées. L'entité de fraisage qui est acceptée est généralement celle qui a envoyé une réponse positive en premier. Si l'entité *machine de fraisage* approuve la réservation (et ignore alors tous les messages de réservation arrivant après) alors l'entité *pièce* considérée communique avec l'entité *robot* afin de mettre en place son transport vers la *machine de fraisage*. Autrement, l'entité *pièce* diffuse de nouveau un message de demande d'une station de fraisage. La figure 4 montre la coordination de trois stations de fraisage.

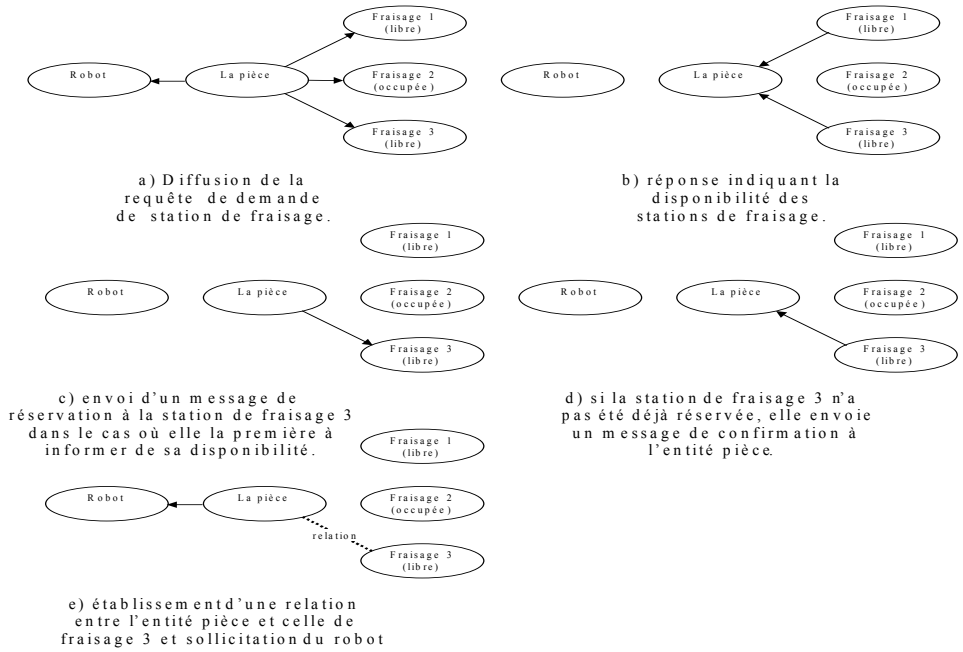


Figure 4. *Réservation d'une entité machine par une entité pièce.*

Si l'entité de fraisage 3 est défaillante ou supprimée du système, elle ne va plus répondre aux messages qui lui sont destinés, mais aucun changement n'est nécessaire dans le système afin de prendre en compte cette modification.
Des versions modifiées de ce protocole ont été proposées dans (Shen et al 1998) où l'entité initiant la coordination garde l'indication sur les ressources non sélectionnées comme alternatives à des situations imprévues. Dans (Kanchanasevee et al 1997), un protocole de contrat a été développé pour l'ordonnancement localisé et distribué sur des stations individuelles dans le cadre des systèmes manufacturiers.

- *Les protocoles de coopération* (Durfee et al 1990), consistent à décomposer un problème en tâches puis à les distribuer sur les différents entités. Les entités ont donc des responsabilités pour des tâches particulières et coopèrent pour la résolution du problème global. Ces protocoles sont particulièrement utilisés dans la surveillance d'un processus quelconque où plusieurs sous-systèmes de surveillance ou entités de surveillance coopèrent pour réaliser les différentes fonctions de la surveillance.

- *Les protocoles de négociation* (Huguet et al 1994), sont utilisés dans le cas où les entités sont en situation de conflit sur une ou plusieurs ressources communes. Une procédure de décision doit être définie pour chaque entité afin de déterminer ses positions, ses concessions et ses critères pour l'accord. La négociation réunit un ensemble d'entités et se déroule jusqu'à ce qu'un accord satisfaisant un pourcentage minimal d'entités participantes soit trouvé. Cet accord prendra la forme d'un contrat sur les ressources communes.

Ces protocoles sont utilisés pour la communication entre les entités d'un système distribué et plus particulièrement dans un système de commande-surveillance distribué. Dans le chapitre III, nous mettons en œuvre de tels protocoles pour un système de détection distribuée.
Plusieurs nouvelles architectures distribuées ont été développées parmi-elles : les architectures holarchiques, biologiques, fractales, les systèmes multi-agents, etc. Le paragraphe suivant donne un bref aperçu de ces architectures.

I.6.5. Les nouvelles architectures distribuées

Il existe plusieurs nouvelles théories de systèmes distribués, quels que soient leurs domaines d'application (Leitão et 1999). Contrairement aux approches conventionnelles, ces nouvelles structures sont capables de répondre promptement et correctement aux changements sans intervention externe. Ces structures présentent des concepts et des comportements similaires mais différents de par leurs origines. Parmi ces structures, nous citons :

- L'architecture holarchique (Winkler et al 1994) introduisant une structure hiérarchique dans une structure distribuée. Son origine relève du domaine de l'organisation sociale. Le mot « *Holon* » est utilisé pour décrire une unité organisationnelle basique dans des organismes vivants ou des organisations sociales. L'Holon est une entité autonome et coopérative d'un système qui inclut des caractéristiques opérationnelles, des compétences, un savoir et des buts individuels. Il peut représenter une activité logique ou physique, comme un robot, une machine, un ordre, un opérateur, etc. (Leitão et al 2000), (figure 5).

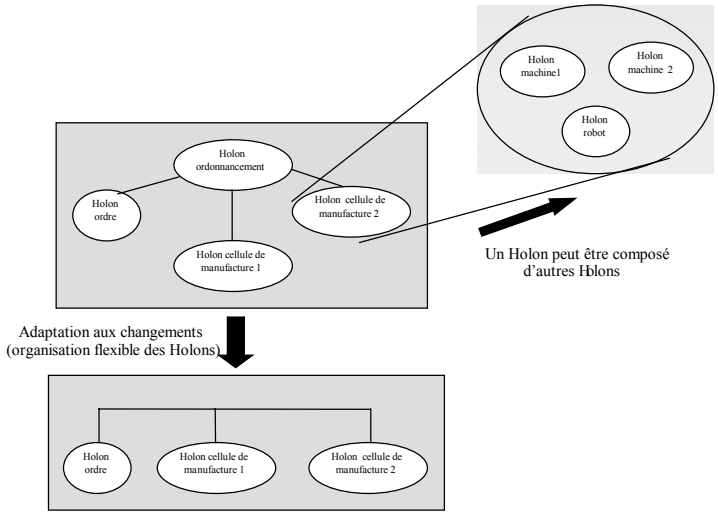


Figure 5. *L'architecture holarchique selon Leitão*.

- L'architecture bionique (Fujii 1997) traduisant les concepts de la biologie, comme l'auto-organisation et les mécanismes d'évolution, au niveau du système considéré. Cette structure se base sur le concept de « *modelon* » qui est composé par d'autres modelons formant ainsi une structure hiérarchique. La notion d'hérédité pour l'ADN (Acide DésoxyriboNucléique) est traduite dans le contexte manufacturier par les caractéristiques et les comportements transmis intrinsèquement aux modelons développés (Tharumarajah et al 1996). Le modelon est autonome et contrôle son comportement en se basant sur son environnement et sur son code génétique (ADN). Dans le domaine manufacturier, ces entités autonomes sont des cellules d'usinage, des cellules d'assemblage ou des unités de transport (Leitão et al 1999). Lorsque les possibilités offertes par la cellule et les besoins du produit concordent, le produit est alloué à la cellule d'usinage, et la cellule et le produit passent dans l'état occupé. Cette dernière action permet de garantir que le produit et la cellule d'usinage ne peuvent être alloués à d'autres unités. Le concept biologique d'« enzyme » et son rôle sont modélisés par des entités appelés superviseurs, qui sont responsables de la régulation et de la commande du système.
  L'aspect opérationnel de l'approche bionique, dans les systèmes manufacturiers, est illustré par un exemple de production de pièces (Okino 1992). Une pièce représentée par un modelon a l'information relative à son usinage ainsi que les outils et les équipements nécessaires. En utilisant cette information, ce modelon est capable de communiquer avec les modelons représentant les ressources nécessaires et de coopérer ainsi afin de produire la pièce.

- L'architecture fractale (Bischoff 1998) introduisant de nouveaux concepts qui ont pour but de résoudre les problèmes de flexibilité dus à des changements externes ou internes au sein du système (par exemple, une entreprise). Cette structure a une origine mathématique et se base sur le concept de « *fractal* ». Un fractal peut être décomposé en plusieurs fractals qui ont les mêmes structures organisationnelles et les mêmes buts.
  Les applications les plus importantes ont été développées en Allemagne, plus particulièrement dans le domaine Automobile (Leitão et al 1999). (Warnecke 1993) propose une application de cette structure aux entreprises fractales (fractal company) dans lesquelles ont été introduits des unités commerciales, des processus commerciaux, la segmentation du système manufacturier et des structures de production cellulaires.

- Les architectures ou systèmes multi-agents sont étudiés dans le cadre de l'Intelligence Artificielle distribuée (IAD) (Erceau et al 1994). Un système multi-agents est un ensemble organisé et distribué d'agents interagissants. Un agent (Erceau et al 1992) est défini comme une entité physique ou abstraite ayant les caractéristiques suivantes :

    - capacité à d'agir sur elle même et sur l'environnement,
    - utilisation d'une représentation partielle de cet environnement,
    - capacité de communication avec d'autres agents,
    - poursuite d'un objectif individuel.

    Le comportement d'un agent est posé comme étant la conséquence de ses observations, de ses connaissances, de ses compétences et des interactions qu'il peut avoir avec d'autres agents et l'environnement. « Dans un système multi-agents (SMA), tout est distribué, réparti : la connaissance, le contrôle, les compétences, l'activité, la planification, etc. » (Ferber 1997). Plus particulièrement, la surveillance est aussi distribuée dans un SMA. Les systèmes multi-agent (Ferber 1995) peuvent être définis comme un ensemble de nœuds désignés par agents et représentant les objets du système sous forme d'entités informatiques. L'architecture multi-agent remplit les besoins définis dans les sections précédentes dont :

    - *l'autonomie* : un agent peut opérer sans intervention externe et exerce un certain contrôle sur son comportement,
    - *la coopération* : les agents interagissent entre eux afin d'accomplir un but commun,
    - *la réactivité et la pro-activité* : les agents perçoivent leur environnement et répondent rapidement aux changements qui s'y opèrent. En plus, les agents n'agissent pas simplement en réponse à leur environnement, mais sont capables de prendre l'initiative en contrôlant leur comportement,
    - *L'adaptation* : les agents peuvent être organisés en une structure distribuée, et facilement réorganisés en d'autres structures organisationnelles différentes.

    Pour un système quelconque, les SMA fournissent un cadre générique pour une décomposition distribuée (cf. chapitre III), un agent pouvant être associé aux différentes fonctions de surveillance du système.

Les trois premières architectures possèdent des concepts et des caractéristiques similaires mais peuvent être distinguées par leurs origines : mathématique pour la fractale, nature pour la biologique et organisation sociale pour la holarchique. Les concepts de ces architectures s'unifient en proposant des systèmes distribués, autonomes et adaptatifs. Mais, leur mise en place diffère d'une architecture à une autre. En plus, dans un système multi-agents, un agent ne peut pas être décomposé en plusieurs autres agents, chose permise pour l'architecture holarchique.
Les principaux concepts de ces différentes architectures peuvent être implémentés en se basant sur la notion d'agent pour conduire à des architectures basées agents qui répondent aux des besoins de modularité, de distribution, de réduction de complexité, etc., (Parunak 1994).

I.6.6. L'architecture basée agent

Il existe dans la littérature beaucoup de travaux sur les architectures basées agents. Ces architectures mettent en jeu une variété d'agents dédié chacun à une tâche particulière. Par exemple, l'architecture proposée par (Leitão et al 2001) est une architecture basée agent formée d'un ensemble d'agents autonomes, intelligents et coopératifs. Cette architecture a la particularité de

présenter des mécanismes de surveillance intégrés dans tous les agents. L'architecture d'un agent générique est représentée figure 6.

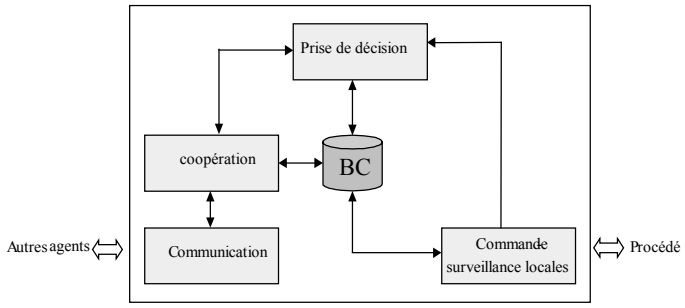


Figure 6. *Architecture d'agent générique (Leitão 2001)* .

Le module « prise de décision » contrôle les activités de l'agent en gérant la résolution de problèmes et la prise de décision. Pour cela, ce module utilise l'information en provenance de la Base des Connaissances (BC). Si cette information n'est pas suffisante à la résolution du problème, un processus de coopération est lancé.
Le module « coopération » gère la coopération avec des agents externes en demandant la coopération d'autres agents, en collectant les réponses, et en les envoyant au module « prise de décision ».
Le module « communication » gère la communication entre les différents agents en définissant un langage de communication.
Le module « commande-surveillance locales » gère la commande et la surveillance de l'exécution des opérations de l'agent. Il est divisé en quatre sous-modules : un gestionnaire de la commande, un régulateur, un module surveillance et la commande opérationnelle. Le gestionnaire de la commande et le régulateur gèrent les tâches à exécuter et les répartissent sur les éléments physiques du procédé. Le module de surveillance permet la surveillance passive demandée par le module de prise de décision pour avoir des informations sur le déroulement de l'exécution d'un ordre ou sur la capacité d'un site, et la surveillance active des activités des ressources à partir d'un système basé événements permettant de notifier l'occurrence de symptôme de défaillance au module prise de décision pour que ce dernier puisse trouver et appliquer la séquence de recouvrement adéquate.
La base de données locale (BC) contient toute la connaissance concernant le comportement de l'agent et la communauté (ressources, tâches) à laquelle il appartient. L'information contenue dans la base de données locale inclut plusieurs types de connaissance : contraintes, objectifs, procédures, règles et expériences, structures organisationnelles et techniques.
A partir de la définition d'un agent, l'architecture proposée considère quatre classes d'agents, (figure 7) :

- l'agent superviseur supervise plusieurs agents opérationnels ;
- l'agent interface gère l'interaction avec le système externe et les utilisateurs ;
- l'agent produit représente le produit et l'information qui lui est associée, comme les plans d'exécution ;
- l'agent tâche représente l'exécution d'une tâche et contient l'information qui lui est relative.

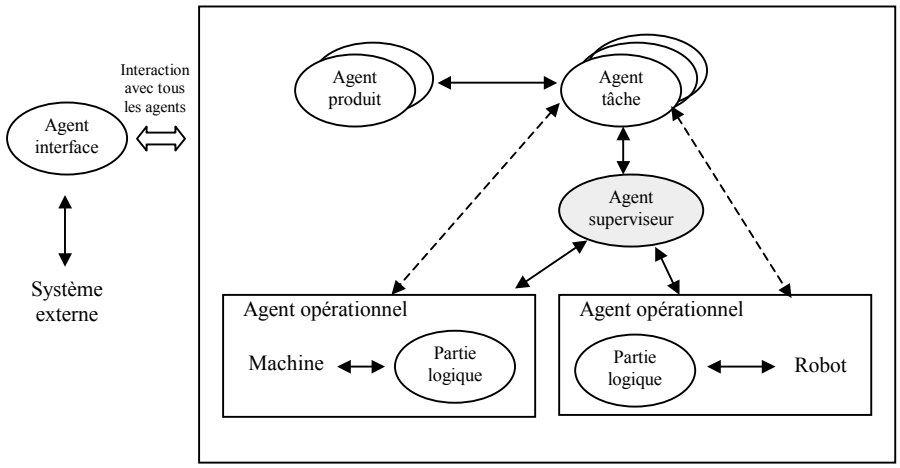


Figure 7. *Les classes d'agent dans l'architecture basée agent, Leitão 2001* .

En plus, les agents sont dotés d'une capacité d'auto-configuration (Coudert et al 2003) leur permettant de se réorganiser en différentes structures organisationnelles. Pour cela, il est nécessaire d'utiliser des méthodes pour représenter les relations entre les agents pour chaque structure organisationnelle et des techniques permettant de manipuler cette structure.

L'architecture proposée met en évidence la nécessité pour les agents de communiquer et de coopérer. Plus particulièrement, la communication et la coopération sont entreprises par les agents dans le but d'une prise de décision distribuée pour la surveillance et/ou la commande du système modélisé. L'organisation des agents dans la structure peut varier selon les besoins à exprimer et les mécanismes mis en œuvre dans un même agent ou entre plusieurs agents. Ces mécanismes dépendent des fonctionnalités dont le concepteur veut doter les différents agents.

I.7. Conclusion

Nous avons présenté dans ce chapitre un aperçu général des différentes architectures de surveillance-commande. Nous avons montré que les architectures distribuées présentent des solutions adéquates aux problèmes posés par les autres architectures conventionnelles. La réduction de la complexité, la tolérance aux fautes, la réduction des coûts, l'amélioration des caractéristiques tout en maintenant la robustesse et la flexibilité, sont les avantages des architectures distribuées. Les architectures holarchique, bionique et fractale se basent sur la notion d'agent et sont de nouvelles architectures flexibles adoptant les principes des architectures distribuées pour la conception de systèmes distribués complexes.

Les besoins de commande et de surveillance doivent être pris en compte et intégrés dans tout système distribué. Il apparaît donc nécessaire de définir des entités ou agents de surveillance-commande et les mécanismes associés aux fonctions de base de ces entités, notamment ceux liés à la fonction détection, objet de notre travail de recherche. Notre contribution se situe dans ce contexte et concerne la mise en place de la fonction de détection distribuée dans les différents agents de surveillance ainsi que les interactions nécessaires à la réalisation du but de la détection.
Avant d'aller plus avant dans ce problème de la détection distribuée de défaillances du procédé, nous allons dresser un rapide état de l'art des travaux menés à partir d'architectures distribuées de surveillance-commande.

Chapitre II

Distribution du système de surveillance-commande : prise en compte des aspects temporels dans les systèmes à événements discrets

II.1. Introduction

De nos jours la complexité et la taille des systèmes (réseaux de communication, systèmes de bases de données, circuits digitaux, systèmes manufacturiers, etc.) sont telles qu'il est nécessaire de s'intéresser non seulement aux aspects commande mais également aux aspects surveillance. La surveillance traite toute déviation du système par rapport à une évolution attendue. La détection du symptôme d'une défaillance à l'origine de cette déviation et l'isolation de l'élément responsable de cette défaillance sont des tâches prépondérantes de la surveillance qui ont suscité une attention considérable et donné lieu à de nombreux travaux de recherche. Un grand nombre d'approches allant de l'arbre de faute et des méthodes basées sur la redondance analytique aux méthodes basées raisonnement et systèmes experts, ont été proposées afin d'aborder ce problème dans le cadre des systèmes à événements discrets. Pour une brève description de ces méthodes le lecteur peut se référer à (Pouliezos et al 1994).

La plupart des approches présentées ci-dessus ont été développées pour des systèmes dans lesquels l'information utilisée pour la commande-surveillance est centralisée. Or, beaucoup de systèmes sont décentralisés par nature, notamment la majorité des systèmes complexes (ordinateur et réseaux de communication, les systèmes manufacturiers et de puissance, etc.). Pour faire face à la complexité des systèmes et à leur distribution tant géographique qu'informationnelle, et pour des raisons que nous avons évoquées au chapitre précédent, il devient nécessaire d'utiliser plusieurs sous-systèmes ou sites, de surveillance et de commande, autonomes et coopérants.

La première partie de ce chapitre est une présentation générale des concepts de base de la distribution du système de surveillance-commande. Elle englobe tous les aspects relatifs à un système de commande distribué en particulier sa représentation par réseau de Petri ou par automate à états finis, ainsi que la prise en compte des délais de communication. La surveillance distribuée y est aussi abordée à travers la présentation de différentes approches et d'architectures pour la surveillance. La spécification et la vérification des contraintes temporelles qui sont à la base des mécanismes de la fonction détection sont présentées. Ce qui nous conduira à considérer en détail le problème de la détection de la violation au plus tôt d'une contrainte temporelle. Dans la deuxième

partie, nous nous intéressons aux problèmes soulevés par la distribution du système de surveillance-commande et nous proposons certains éléments de réponse à ces problèmes. Dans la troisième partie, nous nous intéressons au modèle, sur lequel vont se baser les mécanismes de surveillance, appelé chronique.

II.2. La distribution du système de surveillance-commande

Dans la commande distribuée des systèmes à événements discrets, la commande est partagée entre plusieurs contrôleurs ou sous-systèmes de commande. Chacun des contrôleurs observe un sous-ensemble des événements en provenance du système commandé, et a une possibilité limitée d'en exercer la commande. Un des challenges de la commande distribuée est la synthèse d'un ensemble de lois de commande distribuées permettant d'atteindre les objectifs de la commande du procédé. Pour cela, les contrôleurs doivent se coordonner et partager de l'information.
Les systèmes de coordination de véhicule et les réseaux locaux sans fil comportent des sites spatialement séparés (par exemple, véhicules ou radios) d'activité semi-autonome. Puisque ces systèmes fonctionnent grâce à une commande distribuée, il est donc souhaitable que chaque site puisse surveiller l'évolution qui lui est associée et donc soit à même de détecter, localiser, ses propres symptômes de défaillance, d'où la notion de surveillance distribuée. En général, un site nécessite de l'information émanant d'un autre site afin de pouvoir accomplir sa surveillance. Ce partage d'information est réalisé à travers des protocoles exécutés moyennant un réseau de communication.

Dans cette section, nous allons présenter tous les aspects relatifs aux systèmes de commande distribués et aux systèmes de surveillance distribués.

II.2.1. Les systèmes de commande distribués

Les systèmes de commande décentralisés ou distribués se composent comme nous l'avons déjà dit, de plusieurs sous-systèmes de commande qui possèdent une autorité de commande distribuée, sur les différents actionneurs. La commande distribuée en l'absence de communication a été étudiée dans (Cieslak et al 1988) (Lin et al 1990) (Rudie et al 1992). Cependant, il existe des problèmes liés à la commande distribuée qui ne peuvent pas être résolus sans communication. Ces problèmes sont généralement liés à celui de l'estimation de l'état global du procédé et de la coordination entre les sous-systèmes de commande. Coordination qui permet de déterminer les actions correctes de commande à exécuter au niveau du procédé. Récemment, l'intégration de la communication dans les systèmes de commande décentralisés a été au cœur de plusieurs travaux comme ceux de (Hadj-Alouane et al 1994) (Ricker et al 1997) (Wong et al 1996) (Barrett et al 1998, 2000). La modification fondamentale qui en résulte est que les différents sous-systèmes de commande observent chacun un sous-ensemble des événements générés par le procédé et sont capables d'échanger des messages.
Dans la littérature, nous pouvons rencontrer plusieurs approches pour la commande distribuée qui diffèrent par le modèle utilisé pour la représentation du procédé. Les deux modèles les plus utilisés sont *les réseaux de Petri* et *les automates à états finis*.

II.2.1.1. Commande distribuée par réseaux de Petri contrôlé

Dans cette approche proposée par (Guan et al 1995, 1997), le problème de la loi distribuée de commande a été abordé au travers de réseaux de Petri contrôlés (Krogh 1987). Les réseaux de Petri contrôlés sont une classe de réseau de Petri permettant à un contrôleur d'influencer la progression des jetons dans le réseau. Un *réseau de Petri contrôlé* est un quintuplet $\Gamma = \{\Pi, T, E,$

X, B}, où *Π* est un ensemble fini de places, *T* est un ensemble fini de transitions, *E* est un ensemble d'arcs orientés entre transitions et places, *X* est un ensemble de *places de contrôle*, et *B* est ensemble d'arcs orientés des *places de contrôle* vers les transitions. La commande du réseau de Petri contrôlé est une projection $u : X \rightarrow \{0, 1\}$ associant une valeur zéro ou un à une place de contrôle.

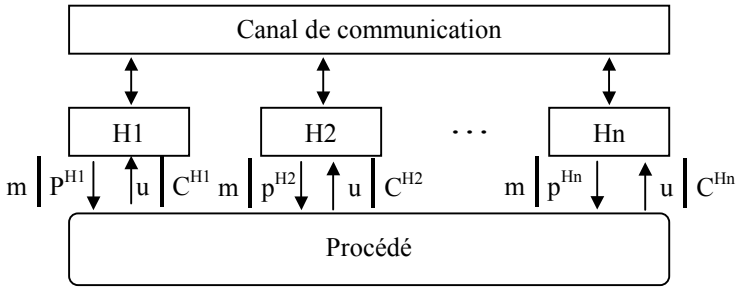


Figure 1. *Un système de commande distribué.*

L'état d'un réseau de Petri contrôlé est donné par son marquage, qui est une distribution des jetons dans l'ensemble des places. Formellement, un marquage est une fonction $m : \Pi \rightarrow N$, où *m(p)* est le nombre de jetons dans *p*. Dans la figure 1, les *Hi*, i=1..n, représentent un ensemble de contrôleurs (sous-systèmes de commande locaux) opérant sur un procédé modélisé par un réseau de Petri contrôlé et échangeant de l'information à travers le canal de communication. Dans cette structure, un sous-système de commande observe une portion limitée du système à commander, échange de l'information avec d'autres sous-systèmes de commande, et utilise cette information afin de déterminer les actions de commande appropriées.
Pour un marquage *m*, la notation $m|_P$ pour $P \subseteq \Pi$ représente la restriction de *m* sur le sous-ensemble des places P. La notation $u|_C$ pour $C \subseteq X$ est définie de la même manière comme la restriction de *u* sur le sous-ensemble *C*.

Le schéma de commande considéré est composé d'un ensemble H de contrôleurs, avec chaque $H \in H$ possédant *un domaine de contrôle* (P^H, T^H, C^H) constitué des places $p \in P^H$ pour lesquelles un marquage peut être observé par *H*, des places de contrôle $c \in C^H$, et des transitions $t \in T^H$ qui sont connectées à un élément de P^H ou de C^H. Les objectifs de commande distribuée considérés par (Guan et al 1997) sont d'éviter le marquage d'états interdits et sont spécifiés comme *des conditions interdites*. Une condition interdite $F \subseteq P$ est un ensemble de places qui ne doivent pas être simultanément marquées.
La figure 2 représente un réseau de Petri contrôlé avec deux conditions interdites $F_1 = \{p_1, p_3\}$ et $F_2 = \{p_2, p_4\}$. Considérons les domaines de contrôle comme indiqués sur la figure 2. Ceci implique qu'il y aura deux contrôleurs H_1 et H_2 avec les projections des conditions interdites $\Phi_1 = \{F_1, F_2\}$ et $\Phi_2 = \{F_1, F_2\}$, sachant que pour un contrôleur $H \in H$, la projection de Φ sur H est définie par :

$$\Phi^H = \{F \in \Phi \mid F \cap P^H \neq \varnothing\}$$

Connaissant les conditions interdites, un contrôleur agit sur les places de contrôle en associant une valeur permettant ainsi d'empêcher les marquages atteignables à partir du marquage courant de satisfaire une ou plusieurs des conditions interdites. Ceci est réalisé grâce aux observations du contrôleur recueillies du système.

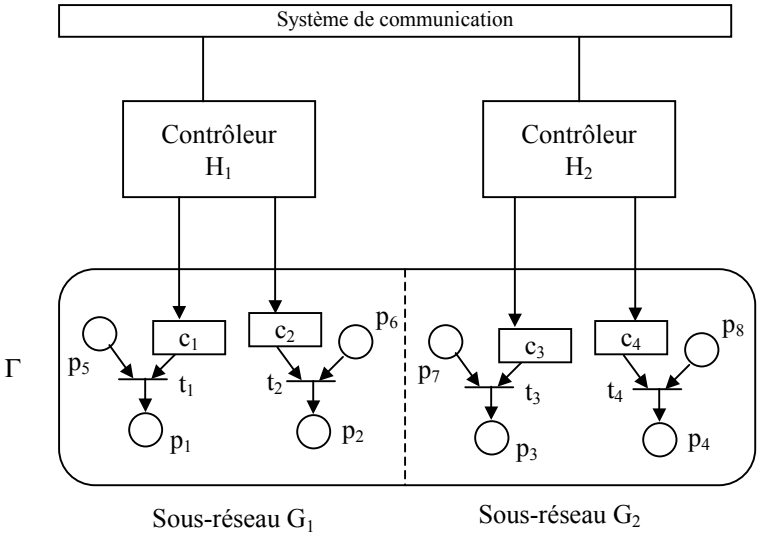


Figure 2. *Un exemple d'un système de commande distribué.*

Le système de communication considéré par (Guan et al 1995) modélise l'échange d'information par des variables de commande auxiliaires, chacune d'elle est déterminée par un contrôleur local spécifique, et observée ensuite par un sous-ensemble d'autres contrôleurs. Cette information échangée permet à chaque contrôleur moyennant son observation locale du modèle du procédé d'exercer la commande locale afin d'empêcher la satisfaction de conditions interdites qui peuvent être distribuées sur plusieurs contrôleurs moyennant la fonction projection définie précédemment, (figure 2).

La démarche proposée distribue les mécanismes de la commande et montre l'utilité de la communication entre plusieurs contrôleurs. En effet, les diffèrents contrôleurs coopèrent, moyennant leurs observations locales, afin d'éviter la satisfaction de conditions interdites distribuées relatives à leurs marquages respectifs. Cette coopération est due au fait que les contrôleurs ne peuvent pas prendre de décisions de commande de manière indépendante.

II.2.1.2. Commande distribuée par automates à états finis

De nombreux travaux se sont basés sur les modèles d'automate pour l'étude de la commande distribuée. Les comportements possibles du système sont décrits par un ensemble de séquences sur un alphabet d'événements définissant ainsi un langage. Ce langage peut être généré par un automate à états finis représentant le modèle de commande du système. Un automate est défini par $\Gamma = (Q, \Sigma, \delta, q_0, Q_m)$, où Q est ensemble non vide d'états, Σ est l'ensemble des événements, $\delta: Q \times \Sigma \rightarrow Q$ est la fonction transition sur les états, q_0 est l'état initial, et $Q_m \subseteq Q$ est l'ensemble d'états « marqués ». Γ représente le modèle du procédé appelé aussi *générateur* qui partant d'un initial q_0 génère une séquence d'événements Σ^*. Le langage fermé généré par Γ est défini par :

$$\Lambda(\Gamma) = \{s \in \Sigma^* / \delta(q_0, s) \text{ est défini}\}.$$

Dans l'approche classique de la commande supervisée, un *superviseur* est couplé au modèle de commande. Le rôle du superviseur se limite à une autorisation/interdiction de l'occurrence de certains événements alors que le rôle du *contrôleur* (système de commande) est de forcer la

production de certains événements dans le procédé. Les entrées du superviseur représentent également les sorties du contrôleur. Les sorties du superviseur, qui sont les entrées du modèle de commande, représentent la liste des *événements interdits* $\Phi(j)$ pour chaque état *j* du modèle de commande. Généralement, les travaux sur la commande supervisée existant dans la littérature classifient les événements qui interviennent dans le fonctionnement d'un système en deux catégories : *incontrôlables* et *contrôlables*. Un événement est dit incontrôlable si le superviseur ne peut pas interdire son occurrence permettant ainsi de respecter les spécifications du système. Par opposition, on dit qu'un événement est contrôlable si le superviseur peut toujours interdire son occurrence.

Dans l'approche de commande supervisée et distribuée, (Yeddes et al 1999) divise le procédé en différents sous-procédés P_i, i=1..n.. Ces procédés sont couplés avec *n* superviseurs et contrôleurs. Un superviseur S_i reçoit un sous-ensemble $\Sigma_j \subset \Sigma$ d'événements en sortie du contrôleur C_j (j n'est pas forcément i), et établit pour chaque état du contrôleur une liste d'événements interdits Φ_i faisant partie de l'ensemble des événements contrôlables. Ceci signifie que le sous-procédé supervisé ne doit générer que les événements de $\Sigma_i \backslash \Phi_i$.
Pour représenter le fait que les superviseurs ont seulement des observations partielles des traces dans $\Lambda(\Gamma)$, un opérateur de projection : $\Pi_{\Sigma_i}: \Sigma^* \rightarrow \Sigma^*_i$ est défini comme :

$$\Pi_{\Sigma_i}(\sigma) = \sigma \text{ si } \sigma \in \Sigma_i \text{ sinon } \Pi_{\Sigma_i}(\sigma) = \varepsilon, \text{ et } \forall(s\sigma \in \Sigma^*)\ \Pi_{\Sigma_i}(s\sigma) = \Pi_{\Sigma_i}(s)\ \Pi_{\Sigma_i}(\sigma).$$

avec ε la séquence vide.

Cette approche peut, par exemple, être appliquée à la commande supervisée d'une ressource partagée. La figure 3 illustre cet exemple dans le cas où le procédé est subdivisé en deux sous-procédés P_1 et P_2.

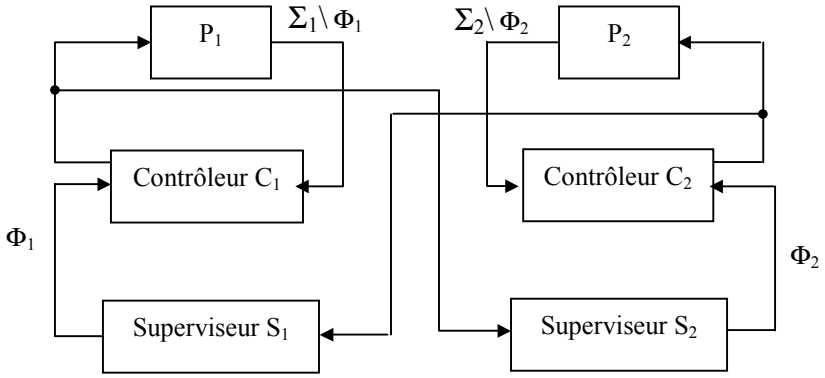


Figure 3. *Commande-supervision distribuées.*

Nous avons présenté dans ce paragraphe une deuxième approche de la commande distribuée dans laquelle les contrôleurs sont couplés à des superviseurs. Les mécanismes de communication développés concernent exclusivement l'échange d'événements entre superviseurs et contrôleurs. Ces événements sont générés par le procédé qui est subdivisé en plusieurs sous-procédés. Les principes sur lesquels se base cette subdivision n'ont pas eu l'attention nécessaire et sont restés mal définis. Nous nous proposons au chapitre III d'y apporter un élément de réponse. En plus, la plupart des travaux sur la commande distribuée avec communication sont basés sur l'hypothèse que les sous-systèmes de commande échangent leur information avec un délai nul. En d'autres termes, que le procédé ne peut réaliser aucune action entre la transmission et la réception d'un message entre les sous-systèmes (contrôleurs, superviseurs) communicants. Cette hypothèse, bien qu'elle soit utile à l'étude de la politique de communication entre sous-systèmes de commande, est souvent irréaliste

dans la pratique dans la mesure où bien souvent les sous-systèmes de commande utilisent des réseaux de communication à délai.

II.2.1.3. Prise en compte des délais de communication dans les systèmes de commande distribués

(Tripakis 2002) a étudié, sous certaines hypothèses, les problèmes de la commande distribuée avec communication par une modélisation et une prise en compte explicite des délais de communication. Deux modèles de communication ont été distingués : le modèle basé sur un délai de communication borné par une constante *k*, c'est à dire qu'entre la transmission et la réception d'un message, le procédé peut générer au plus *k* événements, et le modèle à délai non borné dans lequel le procédé peut générer un nombre quelconque d'événements entre un envoi et la réception d'un message.

(Balemi 92, 94) traite de problèmes similaires mais dans un contexte différent dans lequel l'échange de messages entre le procédé et un système de commande est affecté par les délais de communication. Ceci se traduit par le fait qu'un événement généré par le procédé n'est pas observé immédiatement par le système de commande mais après un certain délai. De manière similaire, une sortie du système de commande n'est pas observée immédiatement par le procédé.

Après avoir considéré les systèmes de commande distribués, nous détaillons maintenant la seconde composante d'un système de commande-surveillance distribué qu'est *la surveillance distribuée*.

II.2.2. Les systèmes de surveillance distribués

Surveiller un procédé distribué entraîne la conception de plusieurs sous-systèmes de surveillance communicants. Chaque sous-système de surveillance collecte les informations relatives au fonctionnement du procédé à travers la réception d'une sous-séquence de la séquence d'événements générés par le procédé, traite ces données et communique éventuellement avec d'autres sous-systèmes de surveillance à travers l'envoi et la réception de messages d'événements, détermine l'état du procédé (normal ou de défaillance) à partir de ses observations et de ses communications, et enfin, produit les inférences nécessaires à la production de données additionnelles localisées comme celles de la fonction détection ou du diagnostic. Les mécanismes utilisés pour la réalisation des fonctions de détection et de diagnostic se basent généralement sur des modèles de comportement local intégrant des aspects temporels, ainsi que sur des protocoles de communication entre les différents sous-systèmes de surveillance.

II.2.2.1. Les approches de la surveillance distribuée

Il y a une distinction entre un système de surveillance *sémantiquement distribué* et un système de surveillance *spatialement distribué* (Froehlich et al 1996). Les systèmes de surveillance sémantiquement distribués référent à un groupe hétérogène de sous-systèmes de surveillance, dans lequel chaque sous-système a sa propre vue du système. Ceci peut signifier que les sous-systèmes de surveillance sont dédiés à différents aspects du système ou encore utilisent différentes méthodes de surveillance. Les systèmes de surveillance spatialement distribués référent à un groupe de sous-systèmes qui surveillent conjointement un système spatialement distribué avec des relations entre ses équipements ou ressources. Chaque sous-système a une information détaillée concernant une petite partie du système. Un tel système de surveillance peut être modélisé comme un réseau d'automates communicants.
Par la suite, nous nous intéressons à la surveillance distribuée de systèmes spatialement distribués. Plusieurs approches de surveillance existent de nos jours.

- Ainsi, (Fabre 2002) a conçu, en parallèle du système, une architecture de surveillance composée de superviseurs locaux, un par équipement ou composant, qui se coordonnent figure 4. La même approche a été utilisée pour la surveillance des réseaux de télécommunication composés de plusieurs sous-systèmes (Pencolé 1999). Des systèmes de surveillance sont construits pour chacun de ces sous-systèmes. Chaque sous-système de surveillance effectue de manière locale la détection et le diagnostic. Toutes les informations de détection-diagnostic locales sont combinées, en utilisant l'historique des événements observés, afin de générer une détection et un diagnostic de niveau global.

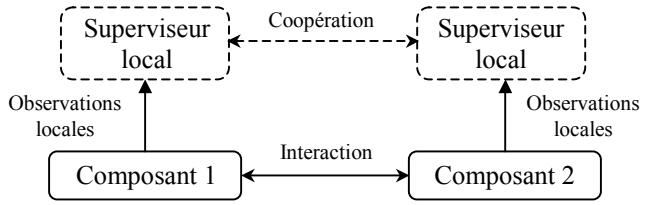


Figure 4. *Un système distribué et une architecture de surveillance modulaire.*

- Dans (Deb et al 1998), chaque unité (module) locale de surveillance connaît ses voisines immédiates, c'est à dire les unités surveillant les sous-systèmes qui affectent ou qui peuvent être affectés par cette unité locale. En permettant l'échange d'information entre unités locales voisines, un algorithme est proposé pour la détection et le diagnostic distribués permettant de réaliser un traitement spécial sur les entrées et sorties du sous-système. Cet algorithme traite chaque entrée du sous-système comme une source de défaillance potentielle et chaque sortie comme un point de test.
- Dans (Sengupta 1998), chaque site possède un sous-système de détection-diagnostic qui observe partiellement le système et qui est en charge de détecter-diagnostiquer les symptômes de défaillances locaux. Les sous-systèmes de détection-diagnostic peuvent échanger de l'information par messagerie. Un problème se pose quand la détection et le diagnostic dépendent du ré-ordonnancement des messages échangés entre sous-systèmes de surveillance et que l'ordre initial de leur émission ne peut pas être rétabli. Ce problème peut être évité dans les systèmes distribués en synchronisant les horloges. Dans les systèmes où les horloges locales ne peuvent pas être synchronisées, ces problèmes restent posés.
- D'autres approches de détection-diagnostic de symptôme de défaillance ont été proposées notamment pour la supervision des systèmes de chauffage et de ventilation dans les bâtiments, et la propagation d'appel dans les réseaux téléphoniques.

Le temps, est une donnée importante associée au fonctionnement d'un système qui a été intégrée dans les mécanismes de la surveillance distribuée. Ce type de surveillance est appelé *surveillance à base de modèles temporels* (Holloway et al 1994). Un modèle temporel consiste, par définition, en un événement déclencheur et en un ensemble de conséquences. L'ensemble des modèles temporels modélisant le système est distribué sur différents processeurs disponibles pour la surveillance et inter-connectés. Les processeurs utilisent les ensembles des contraintes temporelles et de séquencement disponibles à travers les modèles qui leurs sont assignés, pour établir quand sont attendus certains événements, et pour déterminer si un événement est arrivé dans un contexte approprié à un événement le précédent.

Nous présentons par la suite deux architectures de surveillance décentralisée utilisées à des fins de détection et diagnostic dans les systèmes à événements discrets.

II.2.2.1.1. Architecture de surveillance décentralisée proposée par l'Université du Michigan

(Debouk et al 2000a) traitent les problèmes de la détection et du diagnostic de symptôme de défaillance dans les systèmes à événements discrets en se basant sur une information décentralisée.

Dans cette architecture, le système à surveiller est à nouveau modélisé par un automate à état fini :

$$\Gamma = (X, \Sigma, \delta, x_0)$$

Le modèle Γ représente le comportement normal et anormal du système. L'ensemble des événements est $\Sigma = \Sigma_o \cup \Sigma_{no}$ où Σ_o représente l'ensemble des événements observables et Σ_{no} l'ensemble des événements non observables.
Soit Σ_f l'ensemble des événements représentant des symptômes de défaillance qui sont à détecter et à diagnostiquer, avec $\Sigma_f \subseteq \Sigma_{no}$. Le cas où $\Sigma_f \subseteq \Sigma_o$ est simple à traiter puisque les mécanismes de détection de symptôme de défaillance se réduisent à l'identification d'un événement de Σ_f.
L'objectif de ces travaux est d'identifier l'occurrence d'un événement de Σ_f, s'il y a lieu, étant donné que, dans les traces générées par le système, seuls les événements de Σ_o sont observés.

L'architecture proposée (voir figure 5) est décentralisée et coordonnée. Deux sites locaux communiquent avec un coordinateur qui est responsable de la détection et du diagnostic de symptômes de défaillance se produisant dans le système. Chaque site local accède à un sous-ensemble des événements observables. Aussi, une projection P_i est associée au site i, où P_i est définie sur l'ensemble des événements observables Σ_{oi}. L'union de Σ_{o1} et de Σ_{o2} donne l'ensemble des événements observables Σ_o.

Au niveau de chaque site est mis en place un sous-système de détection-diagnostic défini également par un automate à états finis :

$$\Gamma_{di} = (\Theta_{di}, \Sigma_{oi}, \delta_{di}, q_{0i})$$

où Θ_{di}, Σ_{oi}, δ_{di} et q_{0i} sont respectivement l'espace d'état, l'ensemble d'événements, la fonction transition et l'état initial. La fonction de transition δ_{di} est construite de manière similaire à celle de Γ, mais inclut l'attachement des labels des symptômes de défaillance aux états et leur propagation d'un état à l'autre. L'ensemble des labels des symptômes de défaillance est défini par $\Delta_f = \{F_1, F_2, \ldots, F_j\}$. L'ensemble complet des labels possibles est $\Delta = \{N\} \cup 2^{\Delta f}$ (voir figure 7), N est interprété comme étant normal, tandis que F_i, $i \in \{1, 2, \ldots, j\}$ indique qu'un symptôme de défaillance de type F_i s'est produit. L'état initial q_{0i} d'un sous-système de détection diagnostic est défini par $\{(x_0, \{N\})\}$.
L'espace d'état Θ_{di} est composé des états atteignables à partir de q_{0i} moyennant la fonction δ_{di}. Un état q_{di} de Γ_{di} est étiqueté $q_{di} = \{(x_1, l_1), \ldots, (x_n, l_n)\}$, où $x_i \in X_0$ et $l_i \in \Delta$.

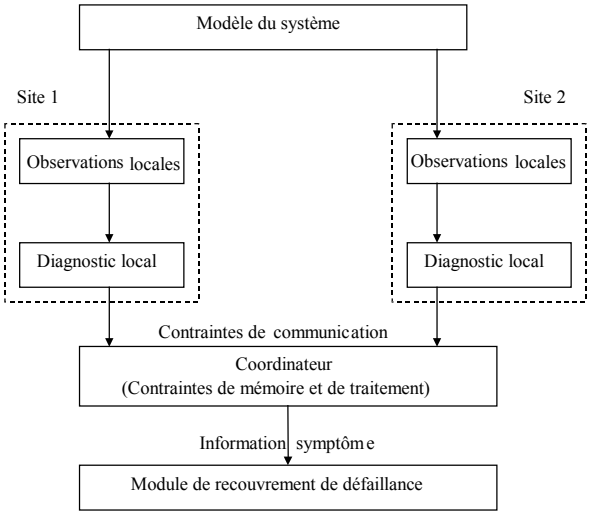


Figure 5. *Système de détection-diagnostic décentralisé et coordonné .*

Pour mieux comprendre cette approche, nous allons l'illustrer sur un exemple. Soit le système montré sur la figure 6. L'ensemble des événements est $\Sigma = \{a, b, c, d, e, \sigma\}$, où σ est le label associé au seul événement non observable et représentant l'occurrence d'un symptôme de défaillance. $\Sigma_{o1} = \{a, c, d, e\}$, $\Sigma_{o2} = \{b, d, e\}$, $\Sigma_f = \{\sigma\}$ et ici $\Sigma_f = \Sigma_{no}$. Les sous-systèmes de détection-diagnostic Γ_{d1} et Γ_{d2} associés aux projections P_1 et P_2 sont montrés sur la figure 7.

L'information de détection-diagnostic disponible au niveau de chaque site est donnée par l'état du sous-système de détection-diagnostic. Les deux sites communiquent une partie de leur information de détection et de diagnostic au coordinateur. Cette information peut être l'état du sous-système de détection-diagnostic. La tâche du coordinateur est de traiter, selon une règle de décision pré-décrite, les messages reçus des deux sites afin de déduire les occurrences des symptômes de défaillance. Si un symptôme de défaillance est détecté par le coordinateur, il est diffusé à un module de recouvrement de défaillance (voir figure 5).

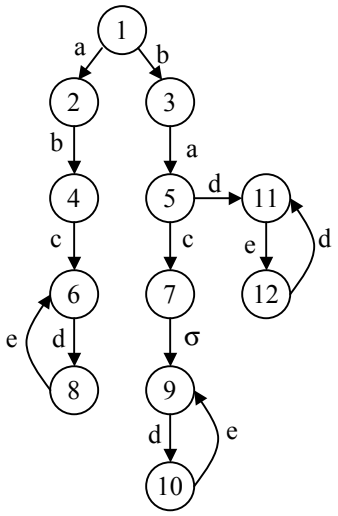


Figure 6. *Un exemple du système Γ.*

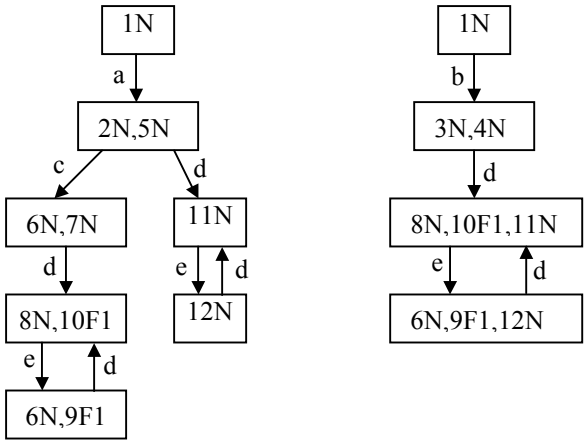


Figure 7. *Les sous-systèmes de détection-diagnostic Γ_{d1} (gauche) et Γ_{d2} (droite) relatifs à l'exemple.*

Afin de mieux définir les rôles alloués aux différents modules de l'architecture proposée, un protocole défini l'information de détection-diagnostic générée par les sites locaux, les règles utilisées par les sites locaux pour communiquer avec le coordinateur, et la règle de décision utilisée par le site coordinateur. Plusieurs protocoles ont été proposés (Debouk et al 1998) (Debouk et al 2000a).
Dans cette approche, supposer l'ensemble des événements représentant des symptômes de défaillance Σ_f connu, est restrictif. En effet, il est impossible d'énumérer exhaustivement tous les événements représentant des symptômes de défaillance. La solution est d'utiliser une approche plus générale se basant sur un modèle représentant les contraintes liées au procédé et qui est mis à jour en permanence par les évolutions provoquées par la commande et par les signaux émis par les capteurs (événements ou comptes rendus).

II.2.2.1.2. Architecture de surveillance décentralisée proposée par l'Université de Toronto

(Ravichandran 2002) a proposé une architecture décentralisée et coordonnée d'un système de détection-diagnostic, (figure 8). Le système de détection-diagnostic est composé d'observateurs locaux, d'estimateurs d'état local, d'un estimateur d'état global et d'une unité de décision.

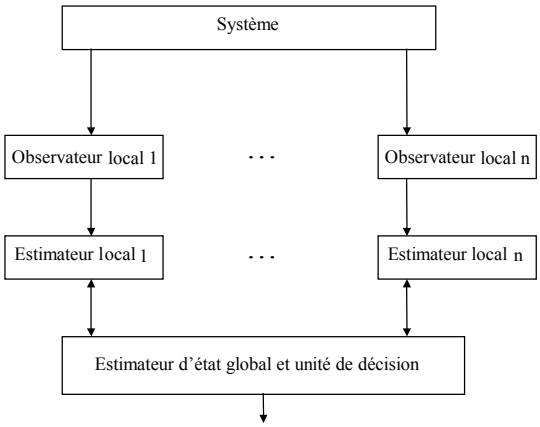


Figure 8. *Architecture du système de détection-diagnostic distribué (Ravichandran 2002)*

Le module « observateur local i » joue le même rôle que celui du module « observations locales » proposé précédemment dans la figure 4 et l'estimateur d'état local *i* a la même fonction que le module « diagnostic local ». Le rôle de l'estimateur d'état global et de l'unité de décision est le même que celui du coordinateur de l'architecture de la figure 4, à savoir l'estimation de l'état global du système surveillé moyennant la réception des estimations des états locaux des sous-systèmes de détection-diagnostic et la prise de décision concernant l'occurrence d'un symptôme de défaillance.
L'estimateur d'état global et l'unité de décision présentent les inconvénients de la structure centralisée (voir chapitre 1).

Après avoir considéré différentes approches de la surveillance distribuée, nous nous intéressons plus particulièrement à la prise en compte des aspects temporels et du comportement du système au niveau des mécanismes de détection.

II.2.2.2. Surveillance distribuée à base de modèle

La majorité des travaux portant sur la surveillance des systèmes à événements discrets a mis en évidence la nécessité de disposer d'un modèle représentant les différentes relations entre les événements générés par le procédé. Ces relations doivent être respectées pour garantir le bon fonctionnement du système. Ainsi toute violation de celles-ci doit être détectée et traitée au plus tôt afin que le système retrouve le plus rapidement possible son fonctionnement normal.

Nous distinguons deux types de modèles : *le modèle de comportement*, représentant des relations de séquencement entre des événements et *le modèle temporel*, représentant des relations temporelles (contraintes temporelles) entre les événements du système à surveiller.

II.2.2.2.1. Surveillance distribuée à base de modèle de comportement

Chaque capteur (respectivement actionneur) doit déterminer de manière autonome si son observation (respectivement son action) est en accord avec un sous-ensemble d'observations (respectivement actions) précédentes. Un modèle de comportement est pour cela défini et est associé à chaque élément de la commande (capteur, actionneur). Ce modèle consiste en une séquence d'événements, appelée *signature d'événements* et comprend un modèle répétitif de changements d'état d'un sous-ensemble d'éléments de commande (passage de l'état activé à l'état désactivé ou l'inverse) et représente ainsi un sous-ensemble de l'état du procédé (Chand 1993). Ce modèle répétitif représente une classe de modèle de comportement dans laquelle la séquence d'événements associée à un élément de commande se répète dans le temps, comme par exemple dans le cas des systèmes manufacturiers où le changement d'état des capteurs et des actionneurs se fait répétitivement à chaque entrée d'une pièce par exemple.
A titre d'illustration, considérons un système de remplissage de bouteilles représenté par la figure 9. Les bouteilles entrent sur un convoyeur et activent un capteur *BI*. Quand le capteur *BI* est activé, le convoyeur transporte les bouteilles vides jusqu'à un bouton poussoir *LS*. Quand le bouton poussoir est actionné, le moteur *(M1)* du convoyeur est arrêté. Le processus de remplissage peut ainsi commencer et le solénoïde *(SA)* est activé. Le solénoïde est désactivé lorsque le capteur d'une cellule photoélectrique *(PE)* détecte que la bouteille est remplie.

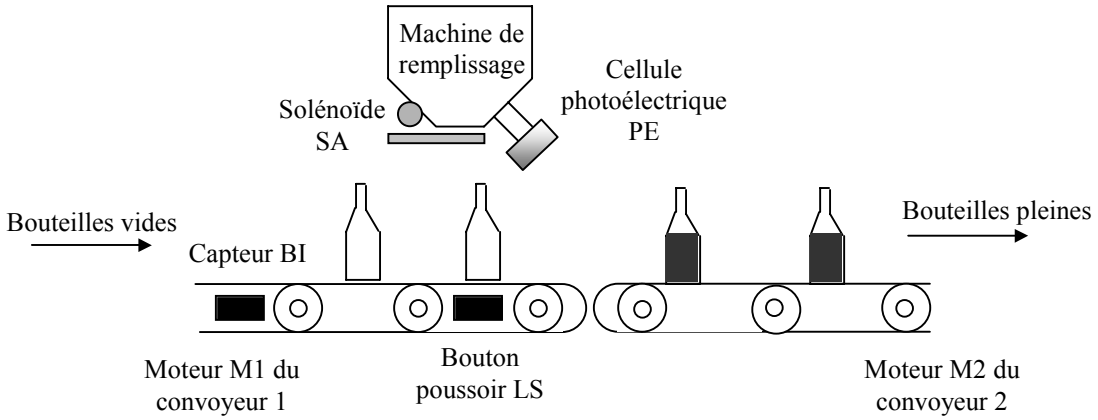


Figure 9. *Exemple d'une station de remplissage de bouteilles.*

Cette séquence est répétée à chaque fois qu'une nouvelle bouteille vide arrive dans le système. La signature est définie par LS↑ → M1↓ → SA↑→ PE↑. Le symbole ↑ indique le passage du capteur de l'état désactivé à l'état activé, alors que le symbole ↓ indique le contraire. Cette séquence exprime que l'activation de *PE* fait suite à l'activation de *LS* suivie par la désactivation de *M1* et par l'activation de *SA*.

Cette signature d'événements fournit à la cellule photoélectrique, un modèle de comportement qui définit une certaine évolution programmée du procédé. A la reconnaissance de cette signature est associée une fonction de probabilité qui prend des valeurs comprises entre 0 (signature non reconnue) et 1 (signature reconnue), et qui croit à chaque fois qu'un événement se produit (Chand 1993). Si cette séquence n'est pas reconnue par la cellule photoélectrique, un symptôme de défaillance est ainsi détecté et diagnostiqué. La non reconnaissance de la séquence peut avoir comme origine l'absence d'événements ou leur apparition dans un ordre différent de celui de la séquence.
Les événements de la signature appartiennent à différents éléments de la commande. Afin que l'élément de la commande auquel est associé l'événement *PE* (cellule photoélectrique) puisse reconnaître cette signature, il est nécessaire qu'il reçoive l'information concernant l'occurrence des autres événements de la séquence. Dans une telle approche, le problème de la prise en compte des délais de communication pour accéder à cette information n'est pas traité. Néanmoins, il est possible d'intégrer les délais de communication dans la formulation des signatures d'événements.
Cette approche s'apparente à celle que nous avons développé au chapitre III dans le cas où un seul site est responsable de la reconnaissance de la séquence et où les délais de communications sont nuls. Lorsque la séquence exprime des délais temporels entre événements en plus de leur séquencement, il devient nécessaire de prendre en compte ces aspects temporels dans la reconnaissance de la séquence.

II.2.2.2.2. Surveillance distribuée à base de modèle temporel

Un modèle temporel est défini par la donnée d'un ensemble de relations entre les dates d'occurrences d'événements en provenance du système surveillé. Généralement ce type de modèle est représenté par automate temporel (Ricker et al 2001), par automate temporisé (Pandalai et al 2000) ou par treillis temporel (Ghallab 1996). Les relations représentées par les modèles temporels sont des contraintes temporelles et de séquencement.

II.2.2.2.2.1. Automate temporel

Un automate temporel est défini par le 8-uplet $G = (T, \Sigma, L, L_0, L_f, T_r, C_T, C_E)$, où :

$T \subseteq \Re$ (ensemble des réels), il représente le temps,
Σ est l'ensemble des événements,
L est l'ensemble des états de l'automate,
$L_0 \subseteq L$ est l'ensemble des états initiaux,
$L_f \subseteq L$ est l'ensemble des états acceptables,
$T_r \subseteq L \times C_T \times \Sigma \times L$ est l'ensemble des transitions de l'automate,
C_T est l'ensemble des contraintes sur les dates d'occurrence des événements,
C_E est l'ensemble des dates d'occurrence des événements.

Prenons à titre d'illustration l'automate temporel suivant :

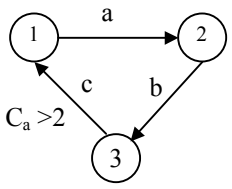


Figure 10. *Exemple d'un automate temporel*.

Dans la figure 10, la contrainte C_a >2 indique que l'événement *c* doit se produire deux unités de temps après l'occurrence de l'événement *a*.

II.2.2.2.2.2. Automate temporisé

Considérons que le modèle temporel est représenté par un automate à états temporisés. Un automate temporisé *G* est représenté par le 5-tuplet $G = (Q_G, \Sigma, A, d, e)$, où Q_G est un ensemble fini d'états, Σ est un ensemble fini de labels d'événement, $A \subseteq (Q_G \times Q_G)$ est l'ensemble des arcs, d est une fonction associant à chaque état un intervalle temporel de résidence, et $e : A \rightarrow \Sigma$ est une fonction assignant un label d'événement à chaque arc dans A. Le temps est dense, ainsi un événement peut se produire entre les bornes de l'intervalle de résidence de l'état. La figure 11 montre un exemple d'automate temporisé, où les nœuds du graphe représentent les états, et l'intervalle associé à chaque état est la durée de séjour dans cet état.

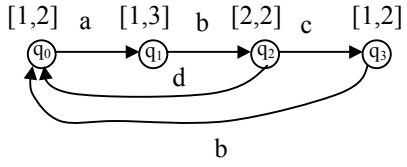


Figure 11. *Exemple d'un automate temporisé*.

II.2.2.2.2.3. Treillis temporel

Le treillis temporel, (figure 12), représente les différentes contraintes temporelles liant les événements du système surveillé. Les labels des événements sont associés aux nœuds du treillis et les contraintes, représentées sous forme d'intervalle [a, b], a, $b \in \Re^+ \cup \{+\infty\}$, sont associées aux arcs

de celui-ci. Un arc allant de e_i à e_j auquel est associé un intervalle [a, b], indique que l'événement e_j doit se produire dans la fenêtre temporelle [a, b] après l'occurrence de e_i.

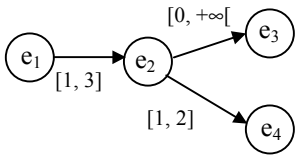


Figure 12. *Treillis temporel.*

Dans une architecture de surveillance distribuée, un modèle temporel est associé à chaque sous-système de surveillance et représente les relations entre les dates d'occurrence d'un sous-ensemble des événements générés par le procédé. L'ensemble de tous les modèles temporels représente toutes les relations existant entre les événements générés par le procédé.

Chaque sous-système de surveillance (superviseur) est généralement associé à un ordinateur ou processeur. De ce fait, une horloge est associée à chaque superviseur, dite *horloge locale*. Les horloges possédant une dérive différente les unes des autres. La datation de l'occurrence d'un événement est ainsi différente d'une horloge à l'autre (Ricker et al 2001). En plus, il peut y avoir des délais de communication entre les différents sous-systèmes de surveillance. La vérification d'un modèle temporel est conditionnée par la vérification des différentes contraintes le constituant en datant les événements par rapport à l'horloge locale.

La spécification de contraintes temporelles est un problème qui a été largement étudié dans la littérature. L'objectif à atteindre est l'obtention de contraintes temporelles représentant le plus « fidèlement » possible les durées séparant des dates d'occurrences d'événements.
Le choix d'un modèle ou d'un autre dépend des spécificités des relations temporelles. En effet, les délais séparant les occurrences des événements peuvent être constants ou bornés. En plus, le parallélisme structurel ne peut pas être exprimé par modèle d'automate temporisé, comme par exemple la vérification en parallèle de deux relations ou contraintes temporelles. Ce parallélisme est possible à exprimer à l'aide du treillis temporel en reliant un nœud du graphe à plusieurs autres nœuds moyennant des arcs. Ceci est illustré par la figure 12 dans laquelle les contraintes liant l'occurrence de e_2 aux occurrences de e_3 et de e_4 peuvent être vérifiées de manière indépendante.

II.2.2.3. Spécification de contraintes temporelles

Les occurrences des événements en provenance du procédé sont liées en fonction de la dynamique du procédé. Par exemple, un événement représentant le début du remplissage d'une bouteille doit être suivi, après un certain délai, par un événement indiquant que la bouteille est remplie. Les relations temporelles et de séquencement entre les changements (événements) au niveau des capteurs et des actionneurs peuvent alors être utilisées pour déterminer si un système fonctionne comme prévu. Afin de disposer de telles relations temporelles, il convient d'observer le fonctionnement du système et d'en extraire toute relation liant des occurrences d'événements, c'est la phase de *détermination* des contraintes temporelles représentant le fonctionnement normal du système. Les relations temporelles considérées sont représentées en terme d'intervalle de délai positif entre des paires d'événements. L'expression de ces contraintes a été abordée par plusieurs chercheurs utilisant des formalismes mathématiques différents. Les expressions de contraintes ainsi obtenues sont utilisées pour la détection d'occurrence de symptômes de défaillance.

Traditionnellement, les chiens de garde ont été utilisés pour détecter un fonctionnement anormal si la durée entre deux événements particuliers dépasse une limite.

II.2.2.3.1. Détermination des contraintes temporelles

(Sujit et al 2000) présente une méthode d'apprentissage de relations temporelles inter-événement utilisant des observations d'un système opérant correctement. Les statistiques de l'échantillon observé, caractéristiques du fonctionnement passé et correct du système, sont utilisées pour créer un espace de confiance des relations temporelles possibles entre les événements en provenance du système. Le délai entre deux événements spécifiques sous l'hypothèse d'un fonctionnement correct (normal) est distribué normalement avec une moyenne inconnue μ et une déviation standard inconnue σ.
Soit *U* l'espace temporel représentant l'ensemble de toutes les relations *(m, w)*, où m est le *centre* de l'intervalle caractérisant une relation temporelle entre deux événements et w sa *largeur*. Etant donnée une paire d'événements (e_1, e_2), une relation temporelle (m, w) indique que pour un système opérant normalement, si e_1 se produit à la date t_1, alors e_2 doit se produire avec un délai positif compris entre m – w/2 et m+ w/2. Si la moyenne μ et la déviation σ sont connues, une relation temporelle (m, w) a alors les valeurs $m = \mu$ et $w = v\sigma$, où *v* est un réel positif constant.

Comme les paramètres μ et σ du délai entre événements ne sont pas connus précisément, les paramètres *m* et *w* ne peuvent pas être déterminés directement. Seules la moyenne $\bar{x}$ et la déviation standard *s* d'un d'échantillon sont disponibles. Ces valeurs sont construites à partir des observations de *n*, ($n \in N^+$), délais x_i entre les événements e_1 et e_2 de l'échantillon relatives au fonctionnement normal du procédé. Le but est d'estimer la moyenne μ et la déviation standard σ du délai entre l'occurrence de e_1 et celle de e_2 en fonction de la moyenne $\bar{x}$ et de la déviation *s* de l'échantillon. Pour cela un niveau de confiance est fixé par un expert et est spécifié comme étant $(1-\alpha)$, où α est l'erreur pour la moyenne μ et la déviation standard σ.
Pour une déviation standard *s* de l'échantillon, la déviation standard du procédé σ, étant donné un niveau de confiance, appartient à un intervalle notée $\sigma_\alpha(n)$, et la moyenne μ à l'intervalle noté $\mu_\alpha(n, \sigma)$. Pour chaque $\sigma \in \sigma_\alpha(n)$ est donnée une plage $\mu_\alpha(n, \sigma)$ de moyennes possibles. Soit $D_v \subseteq U$ l'ensemble de toutes les relations temporelles (m, w) pour $m = \mu \in \mu_\alpha(n, \sigma)$, et $w = v\sigma$, $\sigma \in \sigma_\alpha(n)$.

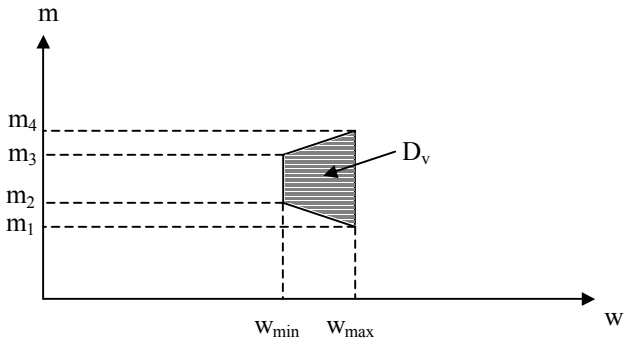


Figure 13. *Représentation de l'espace de confiance D_v.*

D_v représente donc l'ensemble des relations temporelles correspondant au procédé réel avec μ et σ inconnues, (figure 13) (Sujit et al 2000).

Dans le domaine de la surveillance existent deux types de défauts de surveillance : les *fausses alarmes*, quand les mécanismes de surveillance déclarent l'apparition d'un symptôme de défaillance lorsqu'aucune défaillance ne s'est produite dans le système, et *les détections manquées* quand un symptôme de défaillance n'est pas détecté lorsqu'une défaillance s'est produite dans le système. La détermination de relations temporelles inter-événements trop courtes mène à des fausses alarmes, par contre, si ces relations sont trop larges, ceci peut mener à des détections manquées.

II.2.2.3.2. Expression des contraintes temporelles

Après avoir mis en évidence les différentes contraintes liant les événements du système surveillé, il est nécessaire de pouvoir les exprimer avec un formalisme mathématique ou modèle. De façon générale, les contraintes temporelles peuvent être exprimées à partir de relations liant des dates d'occurrence et/ou des dates d'occurrence relatives d'instances d'événements. Une *occurrence* d'un événement définit un point dans un temps dense auquel l'événement se produit. Une occurrence d'un événement est donc une valeur temporelle appartenant à P^+. Dans (Mok et al 1997) est définie la notion d'événement *récurrent*, c'est à dire possédant plusieurs instances. Une fonction d'occurrence d'événement, notée Θ, est définie pour représenter la relation entre les occurrences d'événements et leurs dates d'occurrence durant le fonctionnement.

Définition : pour un événement *e* et un entier $i \in N^+$, la fonction Θ a été définie comme suit :

$\Theta(e, i)$ = la date d'occurrence de la ième instance de l'événement *e*. La fonction Θ est supposée monotone et croissante : $\forall$ *e* un événement, $\forall\ i \in N^+$, $\Theta(e, i+1) > \Theta(e, i)$.

Les contraintes temporelles sont donc des relations entre des paires de dates d'occurrence d'événements et peuvent être exprimées par des inégalités incluant les fonctions d'occurrence correspondantes en utilisant une spécification par langage basée sur la Logique Temps-Réel (LTR).
La contrainte $\Theta(e_1, i) + d \geq \Theta(e_2, j)$ avec $d > 0$, appelée *contrainte date limite* (Jahanian et al 1994), spécifie une date limite pour la jème occurrence de l'événement e_2 qui est de d unités de temps après la ième occurrence de l'événement e_1. La contrainte $\Theta(e_1, i) - d \geq \Theta(e_2, j)$ avec $d > 0$, appelée *contrainte de délai* (Jahanian et al 1994), nécessite la ième occurrence de e_1 à être au moins d unités de temps plus tard que la jème occurrence de e_2.

La spécification de contraintes temporelles peut aussi se faire en associant un intervalle temporel à la durée séparant l'occurrence de deux événements. Dans ce cas, une contrainte temporelle liant deux occurrences d'événements e_i et e_j est représentée sous forme d'intervalle [a, b], a, $b \in \Re^+ \cup \{+\infty\}$, c'est à dire que, $a \leq \Theta(e_j, m) - \Theta(e_i, n) \leq b$. En comparant ce type de contrainte avec les contraintes date limite et les contraintes de délai exprimées précédemment, il est possible d'écrire qu'une contrainte écrite sous forme d'intervalle est la conjonction d'une contrainte de date limite et d'une contrainte de délai puisque,

$$\Theta(e_i, n) + d \geq \Theta(e_j, m) \Leftrightarrow \Theta(e_j, m) - \Theta(e_i, n) \leq d$$
$$\text{et } \Theta(e_j, m) - d \geq O(e_i, n) \Leftrightarrow \Theta(e_j, m) - O(e_i, n) \geq d.$$

Le problème qui se pose maintenant et qui fait l'objet du paragraphe suivant est celui de la vérification ou satisfaction des contraintes temporelles exprimées.

II.2.2.4. Vérification de contraintes temporelles

Vu la croissance continue accordée à l'utilisation d'ordinateurs dans les systèmes temps-réel pour diverses applications comme l'avionique, la commande distribuée de procédé, le contrôle du trafic aérien, etc., le besoin d'avoir des systèmes fiables, délivrant les services attendus et à temps voulu, est devenu crucial. Les systèmes temps-réel interagissent avec l'environnement en réagissant aux événements externes et en produisant des résultats tout en respectant les contraintes temporelles imposées par le procédé. L'étude du problème de détection de symptômes de défaillance dans un système temps-réel distribué est motivé par l'imprévisibilité de l'environnement physique et l'incapacité à satisfaire les hypothèses de conception durant une phase de surcharge transitoire. Ceci peut causer des conditions inattendues ou la violation des contraintes du système en cours de fonctionnement.

Vérifier une contrainte temporelle dans un système distribué est d'autant plus compliqué que les occurrences des événements se font sur plusieurs processeurs (sites). La détection d'une violation le plus tôt possible est réalisée en évaluant les contraintes temporelles au cours du temps, tout en assurant l'échange de messages entre processeurs afin de propager les occurrences des événements. Dans un système temps-réel distribué, il peut y avoir en effet des contraintes temporelles qui portent sur des événements générés sur des processeurs différents.

Un autre problème dû à la distribution du système est celui de la synchronisation d'horloge. En effet, en l'absence d'horloges parfaitement synchronisées au niveau des processeurs (sous-systèmes de surveillance), la signification des contraintes temporelles basées sur des événements distribués doit être définie précisément. (Jahanian et al 1994) présente un algorithme de synchronisation d'horloge permettant de comparer les dates d'occurrence d'événements sur différents processeurs. Nous reviendrons sur ces aspects ultérieurement (§ II.3.1)

La surveillance du système repose sur le fait que le fonctionnement du système est vu comme une séquence d'occurrences d'événements datés. Les hypothèses de conception et les propriétés du système qui doivent être maintenues sont exprimées comme des relations invariantes entre différents événements qui sont surveillées durant le fonctionnement. Si la violation d'une relation invariante est détectée par le système de surveillance, des actions de recouvrement peuvent avoir lieu. Ces relations sont des contraintes temporelles distribuées sur plusieurs sous-systèmes de surveillance. Chaque sous-système de surveillance est associé à un processeur. Le rôle d'un sous-système de surveillance est de détecter les violations de contraintes temporelles au fur et à mesure que les événements d'une contrainte se produisent. Il renvoie aussi l'information concernant l'occurrence de certains événements vers d'autres sous-systèmes de surveillance distants. Dans (Holloway et 1996), le modèle de surveillance proposé utilise des ensembles de relations temporelles et de séquencement permettant de prédire quand les événements sont attendus.

A titre d'exemple, la figure 14 montre la structure d'un système de localisation d'avion. Le signal radar est reçu par un contrôleur de radar. Il est envoyé, par la suite, à un processeur qui effectue le calibrage et la localisation. Le signal obtenu est envoyé à l'hôte d'une interface qui effectue un pré-traitement et envoie le signal au processeur de la console qui réalise le filtrage. Finalement, le signal est envoyé à un processeur spécialisé qui réalise l'affichage de l'information de localisation approprié sur le moniteur. Il y a une durée limite de bout en bout de 2s depuis le moment où un signal est reçu par le contrôleur du radar jusqu'à ce qu'il soit affiché par le processeur d'affichage. En plus, il y a une durée limite intermédiaire de 0.5s sur les commandes d'affichage du processeur d'affichage depuis l'étape de pré-traitement sur le processeur de l'hôte de l'interface.

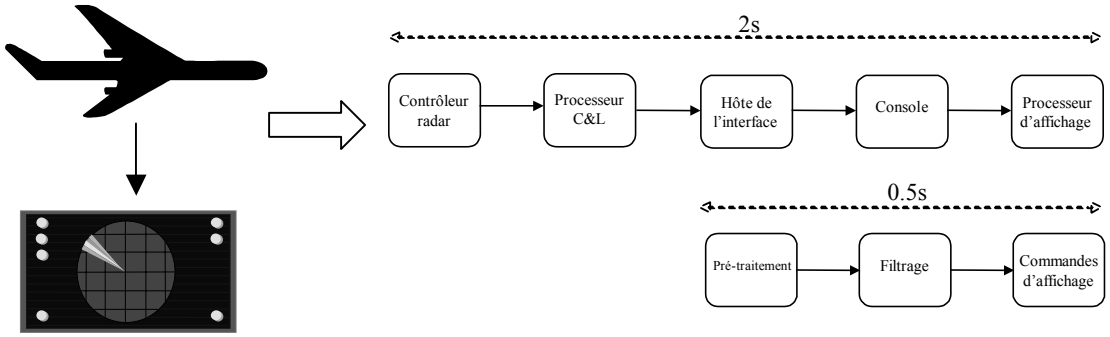


Figure 14. *Système de localisation d'avion.*

Dans un tel système, la surveillance est nécessaire afin de détecter les violations de contraintes inter-processeurs. Les violations de contraintes de délai doivent seulement être testées quand un événement se produit, et les violations de date limite ne doivent pas être testées avant l'expiration d'une certaine date limite.

Nous détaillons, dans la suite, la méthode de surveillance de contrainte temporelle développée par (Pandalai et al 2000). Cette méthode basée sur une prévision complète des dates d'occurrence des événements, est utilisée pour la surveillance des systèmes manufacturiers.

II.2.2.4.1. Prévision des dates d'occurrence des événements

La méthode de détection des violations de contraintes est fondée sur une prévision complète des dates possibles pour chaque événement attendu. Les contraintes temporelles surveillées seront exprimées sous forme d'intervalle.
L'approche de détection de symptôme de défaillance proposée consiste à utiliser un modèle spécifiant le comportement correct ou incorrect du système, et à analyser ensuite le fonctionnement observé du système afin de savoir s'il satisfait les relations spécifiées par le modèle. Les observations du système consistent en des séquences temporisées d'événements. Soit Σ l'ensemble d'événements que le système peut générer et $\Re$ l'ensemble des réels. Une trace temporelle $\rho \subseteq \Sigma \times \Re$, est un ensemble de paires événement-temps qui peut être infini. Une période finie $]0, T_f]$ et une trace ρ permettent de définir :

$$\rho(]0, t_f]) = \{(e,t) \in \rho \,/\, 0 < t \leq t_f\}$$

comme une restriction de ρ sur l'intervalle semi-fermé $]0,t_f]$. Dans ce qui suit, les traces sont seulement considérées sur des périodes de temps finies.

Le modèle de surveillance se base sur la notion de séquence d'événements et de relations temporelles entre événements. A un état du modèle est associé un ensemble de *prévisions* décrivant les événements futurs qui doivent être générés et les périodes relatives dans lesquelles ces événements sont attendus. L'ensemble des prévisions est mis à jour en ligne au fur et à mesure de l'apparition d'événements en ajoutant de nouvelles prévisions ou en supprimant les prévisions satisfaites ou violées.
Une *prévision* est de la forme (t, e, C, w), où $e \in \Sigma$ est un événement, $t \in \Re$ est la date d'occurrence de *e*, *C* est l'ensemble des conséquences, et w est une étiquette. Une *conséquence* est une paire (e',

τ), où e'$\in\Sigma$ est un événement et τ est un intervalle de délai qui peut être à bornes positives ou négatives.

Etant donné une prévision (t, e, C, w), une trace ρ, et une période]0, t_f], la satisfaction ou la violation d'une prévision suit les règles suivantes :

1) (t, e, C, w) est satisfaite sous ρ pour la période]0, t_f] s'il existe une conséquence (e', τ')$\in$C et t'$\in$ [t]$\oplus\tau$' tel que (e', t') $\in$ ρ(]0, t_f]), où $\oplus$ est la somme sur les intervalles.
2) (t, e, C, w) est violée sous ρ pour la période]0, t_f] si elle n'est pas satisfaite par ρ(]0, t_f]) et pour tout (e', τ')$\in$C, [t]$\oplus\tau'\subseteq$]0, t_f]. Lorsque la date globale t_f est atteinte, aucune des conséquences n'a pu être vérifiée alors qu'elles ne peuvent l'être qu'avant t_f. La prévision est ainsi violée.
3) (t, e, C, w) est ouverte sous ρ pour la période]0, t_f] si elle est n'est ni satisfaite ni violée. Dans ce cas la date globale t_f n'est pas encore atteinte et aucune des conséquences n'est vérifiée. La prévision est ouverte jusqu'à ce qu'elle soit vérifiée ou violée.

Par exemple, la figure 15 montre une prévision avec une conséquence unique et un intervalle de temps dans lequel l'événement conséquence est attendu.

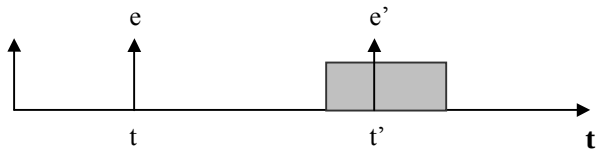


Figure 15. *La satisfaction de la prévision. La partie grisée indique la période τ durant laquelle l'événement e attend l'occurrence de l'événement e'.*

Considérons la prévision (5, e_1, {(e_2, [1, 2]), (e_3, [2, 4])}, w). La prévision est satisfaite par une occurrence de e_2 à la date t, $6 \le t \le 7$, ou par l'occurrence de e_3 à la date t, $7 \le t \le 9$. Pour n'importe quelle date $t_f < 9$, si la prévision n'est pas satisfaite sous ρ(]0, t_f]), alors elle est ouverte. Pour n'importe quelle date $t_f \ge 9$, si la prévision n'est pas satisfaite sous ρ(]0, t_f]), alors elle est violée.

Soit la séquence FERH d'événements F, E, R et H associée à un système avec certaines restrictions temporelles, impliquant par exemple un délai entre l'occurrence des deux événements F et E. Si une séquence FEFREH est observée, puisque cette séquence ne correspond pas à ce qui est attendu, elle entraîne l'apparition d'un symptôme de défaillance. Ce type de procédé est appelé procédé à *instance-unique.* Considérons des pièces sur un convoyeur. Chaque pièce génère la séquence EP (événements E et P) avec un certain délai borné entre E et P. S'il y a au même moment quatre pièces sur le convoyeur, alors la séquence d'événements observée EPEEPEPP avec des délais temporels corrects est parfaitement acceptable et correspond à l'entrelacement des quatre séquences des quatre pièces. Ce type de procédé est appelé procédé *à instance-multiple*. Les systèmes manufacturiers ont généralement des sous-systèmes qui ont les caractéristiques d'une *instance unique* et d'autres sous-systèmes avec les caractéristiques d'une *instance multiple*. Le modèle temporel proposé par (Pandalai et al 2000) permet de surveiller les deux types de sous-systèmes avec une technique unique. Les modèles de surveillance de procédé à instance-unique et à instance multiple sont générés à partir d'un modèle automate temporisé $\Gamma = (\Theta_\Gamma, \Sigma, A, d, \delta)$, (cf § II.2.2.2.2.3).

II.2.4.2.2 Modèle pour la surveillance de procédé à instance-unique

Un modèle temporel pour un procédé à instance-unique inclut :

- des *conséquences inverses interdites* $C_{ii}(q)$ = {(e,]-min(d(q)), 0[) | pour tout événement *e* d'entrée à q},
- *des conséquences postérieures* $C_p(q)$ = {(e, d(q)) | pour tout événement *e* de sortie à q}.

Lorsqu'un événement de sortie de l'état *q* se produit, l'ensemble des conséquences inverses interdites est l'ensemble des événements d'entrée à l'état *q* et des durées les séparant des événements de sortie, durant lesquelles ils ne devaient pas être générés. L'ensemble des conséquences postérieures est l'ensemble des événements de sortie de l'état *q* et des durées pendant lesquelles ils peuvent être générés par le procédé.

Le modèle de surveillance d'un procédé à instance unique est défini comme suit (Pandalai et al 2000) :

$$M_\Gamma = \{(e, \{(\{q\}, C_p(q'), q') / q, q' \in Q_\Gamma \text{ et } \delta(q, q') = e\}) / e \in \Sigma\}$$
$$\cup \{(e, \{(\{q'\}, C_{ii}(q'), prob) / q' \in Q_\Gamma \text{ et a l'événement e comme sortie}\}) / e \in \Sigma\}.$$

L'étiquette « prob » signifie « problème » et *e* est appelé l'événement « déclencheur ».

Reprenons l'exemple d'automate temporisé présenté § II.2.2.2.2.2. (figure 11). le modèle M_Γ pour la surveillance du procédé à instance unique consiste en les résultats suivants :

(a, { ({q_0} , {(b, [1, 3])}, q_1) }),
(b, { ({q_1} , {(c, [2, 2])}, q_2), ({q_3} , {(a, [1, 2])}, q_0) }),
(c, { ({q_2} , {(b, [1, 2])}, q_3) }),
(d, { ({q_2} , {(a, [1, 2])}, q_0) }),

(a, { ({q_0} , {(b,]-1, 0[)}, (d,]-1, 0[)}, prob) }),
(b, { ({q_1} , {(a,]-1, 0[)}, prob), ({q_3} , {(c,]-1, 0[)}, prob) }),
(c, { ({q_2} , {(b,]-2, 0[)}, prob) }),
(d, { ({q_2} , {(b,]-2, 0[)}, prob) }).

Le modèle de surveillance de procédé à instance unique permet de gérer la création de nouvelles prévisions quand un événement se produit, d'éliminer les prévisions quand elles sont satisfaites, et de déclarer l'apparition d'un symptôme de défaillance quand les prévisions sont violées. Lorsque une prévision à laquelle est associée l'étiquette « prob » est satisfaite ou qu'une prévision à laquelle est associée une étiquette de type état est violée, un symptôme de défaillance est détecté.

Ce modèle définit donc à chaque occurrence d'un événement des prévisions à satisfaire et des prévisions qui ne doivent pas être satisfaites. Il se base sur une représentation par automate temporisé des relations temporelles entre événements générés par un procédé à instance unique. Ce modèle impose un ordre prédéfini aux occurrences des événements et ne permet pas l'entrelacement des séquences d'événements. Pour permettre l'entrelacement des séquences, un modèle différent doit être défini.

II.2.2.4.3. Modèle pour la surveillance de procédé à instance-multiple

Le problème de la surveillance sur la base de comportements à instances-multiples à été introduit par (Holloway et Chand 1994, 1996) pour les systèmes automatisés de production. Chaque instance peut avoir un état courant différent. Il peut donc y avoir plusieurs instances qui ont généré un événement donné, et il est peut être impossible de déterminer à quelle instance l'événement appartient. Par conséquent, le modèle de surveillance doit considérer toutes les instances pouvant générer l'événement considéré. Pour cela, la notion d'indéterminisme pour l'observation (Pandalai et al 2000) doit être définie.

Définition : Une séquence d'états $\sigma = (q_0 q_1 \ldots q_n) \in Q^*_\Gamma$, avec Q^*_Γ l'ensemble des séquences d'états, est indéterminée pour l'observation pour Γ s'il existe une séquence σ' tel que $\sigma' \neq \sigma$, les événements associés aux deux séquences sont les mêmes et les durées associées aux états des deux séquences sont les mêmes sauf pour l'état de départ et de terminaison de chacune. Les séquences σ et σ' sont dites indéterminées pour l'observation entre-elles.

Définition : Soit $D \subseteq Q^*_\Gamma$ l'ensemble de toutes les séquences d'états σ tel que :

1) σ est l'état singleton ; ou
2) σ est indéterminée à l'observation, ne possède pas de cycle interne et il existe $q \in Q_\Gamma$ tel que $q\sigma$ ou σq n'est pas indéterminée à l'observation.

A titre d'exemple, dans l'automate de la figure 11, $D = \{(q_1, q_2), (q_3, q_0), (q_0), (q_1), (q_2), (q_3)\}$.

En plus des conséquences postérieures $C_p(q)$ définies dans le paragraphe précédent, ce modèle de surveillance inclut les conséquences inverses $C_i(q)$ qui sont l'ensemble des événements d'entrée à l'état q et des durées leur correspondant,

$$C_i(q) = \{(e, -d(q)) \mid \text{pour tout événement } e \text{ d'entrée à } q\}.$$

Le modèle de surveillance d'un procédé à instance multiple est défini comme suit :

$$M^*_\Gamma = \{(e, \{C_p(\sigma) \cup \{c \in C_p(q) \mid q \in Q_\Gamma, q \neq q_0, \text{ et } e \text{ est un événement d'entrée de } q\} \mid \sigma \in D \text{ et } e \text{ est un événement entrée de } \sigma\}$$
$$\cup \{C_i(\sigma) \cup \{c \in C_i(q) \mid q \in Q_\Gamma, q \neq q_0, \text{ et } e \text{ est un événement de sortie de } q\} \mid \sigma \in D \text{ et } e \text{ est un événement de sortie de } \sigma\}) \mid e \in \Sigma\}.$$

Comme exemple, pour l'automate de la figure 11, le modèle M^*_Γ est donné par la table suivante :

(a, { (b, [1, 3]),	(q_1)
(c, [3, 5]) }	(q_1, q_2)
∪ { ({ (b, [-2, -1]), (d, [-2, -1]) }),	(q_0)
(c, [-4, -2]) }	(q_3, q_0)
(b, { (a, [1, 2]),	(q_0)
(c, [2, 2]) }	(q_2)
∪ { (a, [-3, -1]),	(q_1)
(c, [-2, -1]) }	(q_3)
(c, { (b, [1, 2]),	(q_3)
(a, [2, 4]) }	(q_3, q_0)
∪ { (b, [-2, -2]),	(q_2)
(a, [-5, -3]) }	(q_1, q_2)
(d, { (a, [1, 2]), }	(q_0)
∪ { (b, [-2, -2]),	(q_2)
(a, [-5, -3]) }	(q_1, q_2)

Les prévisions données par la première colonne correspondent aux séquences de la deuxième colonne. Un symptôme de défaillance est détecté si les prévisions du modèle extrait sont violées.

Les modèles de surveillance de procédé à instance unique et à instance multiple peuvent être fusionnés permettant ainsi la création d'un modèle global de surveillance d'instances uniques et multiples s'exécutant concurrentiellement. Ce modèle se base sur un automate temporisé qui, comme nous l'avons dit, ne peut pas représenter tous les types de relations temporelles entre événements. En plus, disposer d'un modèle hors-ligne composé de prévisions peut être difficile dans le cas où il y a un grand nombre de conséquences relatives à l'occurrence des événements. Une solution, que nous développerons au chapitre III, consiste à utiliser des modèles connus qui sont les réseaux de Petri et qui permettent d'exprimer de telles conséquences et de vérifier le bon fonctionnement du procédé quand il est à instance unique ou multiple.

Cette étude est située dans le cas où les délais de communication entre sous-systèmes de surveillance sont négligeables par rapport aux durées exprimées par les contraintes. Ainsi les contraintes liant des événements se produisant au niveau de sous-systèmes différents sont surveillées par échange des occurrences d'événements, datées localement, entre ces sous-systèmes. Il est supposé que chaque sous-système de surveillance connaît toute conjonction de contraintes contenant un ou plusieurs événements, appelés événements locaux, se produisant sur son processeur local. Quand un événement local se produit, le sous-système de surveillance doit donc décider si l'événement doit être communiqué et si tel est le cas, vers quel sous-système renvoyer cet événement afin de détecter la violation d'un contrainte au plus tôt.

II.2.2.5. Détection au plus tôt de la violation de contraintes temporelles

La détection au plus tôt de la violation de contrainte temporelle est nécessaire pour pouvoir réaliser les actions de recouvrement à temps. Une détection tardive de violation de contrainte peut causer la propagation de la défaillance qui s'est déclarée au niveau du système surveillé. Mais la détection au plus tôt de la violation d'une contrainte n'est pas toujours synonyme de détection au plus d'un symptôme de défaillance. En effet, lorsqu'il s'agit d'un ensemble X de contraintes de délai et de date limite à surveiller, une détection de la violation d'une des contraintes de X peut être faite alors que le symptôme de défaillance pourrait être détecté encore plus tôt. Ceci est dû comme

nous allons le voir à ce que des contraintes appelées *contraintes implicites* ne sont pas représentées dans *X*. Une *contrainte implicite* est une contrainte qui n'apparaît pas dans *C* mais qui est logiquement impliquée par *X* (Mok et al 1997).

Soient les contraintes suivantes :

$$O(e_1) + 10 \geq O(e_2), (O(e_2) - 4) \geq O(e_3),$$

avec *O* est une fonction associant une date réelle positive à l'occurrence d'un événement.

La contrainte suivante $O(e_1) + 6 \geq O(e_3)$, est une contrainte implicite. Supposons que e_1 soit produit à la date 1. Si ni e_2 ni e_3 n'est produit à la date 7, alors à la date 7 cette contrainte implicite est violée mais aucune des deux contraintes initiales appelées contraintes *explicites* ne l'est encore. La vérification, dans ce cas, de la contrainte implicite est nécessaire afin de détecter la violation le plus tôt possible. Signalons que ce n'est pas toujours le cas, c'est à dire que les contraintes explicites peuvent être suffisantes à la détection au plus tôt de la contrainte.

Une contrainte peut être représentée par un arc orienté auquel est associé un poids. L'ensemble des contraintes peut être représenté par un graphe orienté appelé *graphe de contraintes*. Le graphe de contraintes de la figure 16 représente les contraintes suivantes :

$$O(e_1) - 1 \geq O(e_2), O(e_1) + 4 \geq O(e_3), O(e_1) \leq 10.$$

les termes des contraintes sont notés par des sommets. Le sommet *zéro* est introduit pour représenter la date *0*.

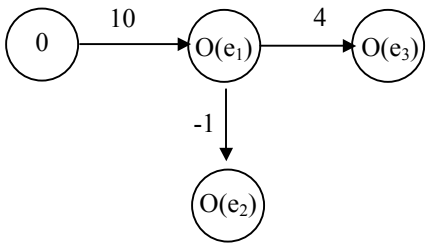


Figure 16. *Un graphe de contraintes.*

De manière générale, étant donné un graphe de contraintes G = (V, E), avec *V* l'ensemble des sommets et *E* l'ensemble des arcs, un chemin avec au moins un sommet intermédiaire représente une contrainte implicite. Dans la figure 16, le chemin du sommet zéro au sommet $O(e_2)$ de longueur 9 représente la contrainte implicite $O(e_2) \leq 9$. En ajoutant les arcs correspondant à toutes les contraintes implicites, le graphe de contraintes *complet* est obtenu. Si un cycle de longueur négative existe dans un graphe de contraintes, alors les contraintes ne peuvent pas être satisfaites (Mok et al 1997).

Dans le cadre de cette thèse, nous supposons disposer d'un ensemble de contraintes permettant la détection au plus tôt d'une quelconque violation. Ceci permet à la fonction recouvrement de s'exécuter au plus tôt et d'éviter ainsi toute propagation des conséquences de la violation.

Après avoir étudié les aspects liés à la spécification et à l'expression de contraintes qui sont les éléments à la base de la surveillance par modèle temporel, nous allons considérer dans le paragraphe suivant les problèmes qui sont apparus suite à la distribution des mécanismes de la surveillance-commande.

II.3. Problèmes liés à la distribution du système de surveillance-commande

Ces problèmes concernent particulièrement des aspects liés à la distribution des mécanismes de datation des occurrences des événements, à la prise en compte des délais de communication et à la minimisation des coûts de communication et de calcul relatif.

II.3.1. Synchronisation d'horloge

Le système de surveillance distribué consiste en un ensemble de sous-systèmes de surveillance ou superviseurs coopérants, chacun associé à un processeur. Chaque sous-système de surveillance maintient le sous-ensemble de contraintes à surveiller, enregistre les événements en provenance du procédé ou d'applications, vérifie les violations de contraintes et transmet des occurrences datées d'événements aux sous-systèmes de surveillance qui ont en besoin pour vérifier leurs contraintes.
Il est supposé (Jahanian et al 1994) que la date d'occurrence d'un événement correspond à la date locale de son occurrence au niveau du processeur sur lequel il se produit. L'occurrence d'un événement est ainsi datée par un et un seul processeur. Le problème est que les horloges des différents processeurs ne sont pas identiques et dérivent l'une de l'autre. La vérification des graphes de contraintes doit donc tenir compte de ces dérives d'horloge.
Si la dérive entre les horloges n'est pas bornée par une valeur connue, la vérification des graphes de contraintes devient une tâche très difficile. Un algorithme (Lundelius et al 1984) a été utilisé afin de maintenir une dérive bornée entre les horloges. Le principe est qu'un processus d'horloge sur chaque processeur associé à un site se synchronise avec un processus d'horloge d'un processeur *maître* en échangeant des messages.
Soit ε la dérive maximum entre les horloges. Supposons une contrainte de délai de longueur P ($P>0$) entre e_i et e_j, c'est à dire que e_j doit se produire au moins P unités de temps après l'occurrence de e_i. Ceci s'exprime par :

$$O(e_i) + P \leq O(e_j)$$

Rappelons que la valeur de $O(e_i)$ est connue seulement par le site datant e_i et celle de $O(e_j)$ par le site datant e_j.
Supposons en plus que l'horloge locale datant l'occurrence de e_j ait ε unités de temps d'avance par rapport à l'horloge locale datant l'occurrence de e_i. Une violation a lieu si e_j se produit sur le site (le processeur) le gérant après l'occurrence de e_i et avant $P + \varepsilon$ unités de temps. Ainsi, la contrainte entre les événements doit être ajustée comme suit (Jahanian et al 1994) :

$$O(e_i) + (P + \varepsilon) \leq O(e_j).$$

De la même manière, une contrainte de date limite de longueur P ($P>0$) entre e_i et e_j impose que e_j se produit avant P unités de temps après l'occurrence de e_i. Ceci veut dire que :

$$O(e_j) \leq O(e_i) + P$$

Dans le cas où l'horloge locale datant l'occurrence de e_j est en retard de ε par rapport à l'horloge locale datant l'occurrence de e_i. Une violation a lieu si e_j se produit après P - ε après l'occurrence de e_i. De ce fait, la contrainte entre les deux événements doit être ajustée à :

$$O(e_j) \leq O(e_i) + (P - \varepsilon).$$

Cette approche traite le problème de la dérive d'horloge en supposant l'existence d'une dérive maximum. La prise en compte de la dérive maximum dans l'expression des contraintes peut influencer la vérification de celles-ci et amener à des fausses détections de violations ou à des violations non détectées. Ceci est dû à la grande précision engendrée par le choix de la valeur maximum de la dérivée qui n'est qu'une estimation grossière de la dérive réelle à l'instant où la contrainte est vérifiée. Au chapitre III, nous développons une approche basée sur la prise en compte des délais de communication dans la vérification des contraintes. Ce délai est dans notre cas borné donnant ainsi plus de précision à la vérification distribuée des contraintes.

II.3.2. Prise en compte des délais de communication

Jusqu'à maintenant, les délais de communication entre les différents sous-systèmes de surveillance étaient considérés comme négligeables. En pratique, les délais de transmission (communication) des événements entre différents sites (sous-systèmes) existent. La non prise en compte de ces délais peut causer le non respect des spécifications du système. (Yeddes et al 1999) propose d'introduire des modèles de communication dans le modèle de la commande. Ceci permettrait de retarder la production de quelques événements dans le procédé afin d'absorber les délais de communication.

A cause des délais de communication entre les différents sites supportant les superviseurs, un superviseur S_i ne peut pas interdire l'occurrence d'événements de Φ_i, rappelons que Φ_i est la liste des événements interdits associée au contrôleur *i*. En effet, l'information permettant au superviseur S_i d'interdire certains événements n'est disponible qu'après un certain délai.
Ce problème est typiquement celui de la commande d'une section critique pour le trafic ferroviaire, figure 17.

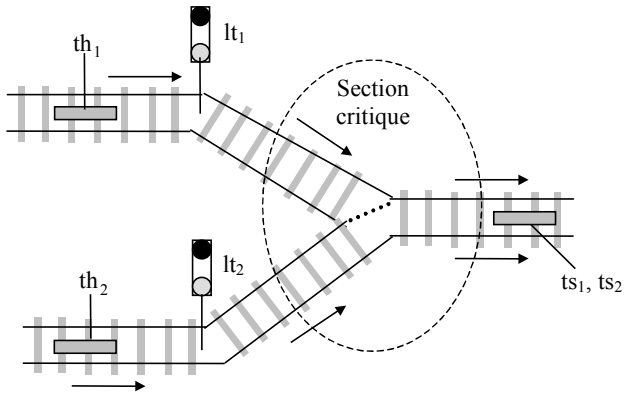


Figure 17. *Trafic ferroviaire*.

Le système physique est composé de deux rails avec deux capteurs (th_i, i=1,2) indiquant la présence ou l'absence d'un train des deux côtés de la section critique, et de deux feux de trafic (lt1 et lt2) permettant d'autoriser le passage d'un train dans la section critique (feu vert). Les capteurs (événements ts_i, i=1,2) indiquent la sortie du train i de la section critique.

Il est supposé que seuls les événements lt_i sont contrôlables. Rappelons qu'un événement contrôlable est un événement qui peut être interdit ou autorisé par un contrôleur. La spécification à respecter par le système est qu'il n'y ait pas deux trains en même temps dans la section critique afin bien évidemment d'éviter les collisions. Les deux feux de trafic fonctionnent indépendamment l'un de l'autre. Les deux superviseurs locaux sont S_1 et S_2. Le superviseur S_1 contrôlant le feu de trafic 1 interdit l'événement contrôlable lt_1 quand le feu de trafic 2 est vert et vis vers ça. Intuitivement il est compréhensible que, si la durée écoulée entre les arrivées des deux trains est inférieure au délai de communication, les deux feux de trafic seront verts en même temps. Dans ce cas, les deux trains seront dans la section critique en même temps. Afin d'éviter une telle situation, il faut tenir compte des délais de communication dans le modèle des superviseurs et retarder le passage au vert des deux feux de trafic d'un certain temps retardant ainsi les dates d'occurrence des événements contrôlables lt_1 et lt_2. Ce retard peut être intégré au modèle de commande sous forme de transition temporelle précédant l'événement contrôlable (Yeddes et al 1999).

II.3.3. Rétablissement de l'ordre global des messages échangés

Une autre conséquence de l'existence des délais de communication dans les systèmes de commande-surveillance distribués est que l'ordre dans lequel des messages sont envoyés d'un site à un autre n'est pas le même que celui dans lequel ces messages sont reçus. Dans un système décentralisé constitué d'un ensemble de superviseurs (sites) et d'un coordinateur, le coordinateur ne peut à cause des délais de communication, ordonner les messages qu'il reçoit. Afin de résoudre ce problème, deux approches peuvent être adoptées :

- La première approche consiste à associer des horloges aux sites de surveillance-commande locaux et à estampiller les messages que ces sites envoient. Les horloges peuvent être synchronisées moyennant des mécanismes connus (Schmid 1997). Les messages peuvent ainsi être ordonnés au niveau du coordinateur grâce à leurs estampilles. Malheureusement ceci n'est pas toujours possible. (Stark 1999) a montré en effet que le mécanisme de synchronisation d'horloge est très contraignant dans le cas de réseaux de communication mobiles.
- La seconde approche, repose sur des algorithmes basés sur des modèles à événements discrets permettant d'ordonner les messages reçus par le coordinateur (Debouk et al 2000b).

Vue que la première approche montre des limites pour certains types de systèmes, la deuxième approche, bien que plus compliquée, offre un cadre plus général à l'application des mécanismes d'ordonnancement de messages en provenance de plusieurs sites. Ainsi, (Debouk et al 2000b) propose de stocker tous les messages arrivant des sites (superviseurs) locaux jusqu'à ce qu'il soit possible de les ordonner. Ceci est réalisé sous l'hypothèse qu'au maximum un seul message est à l'origine du problème d'ordre. La recherche du bon ordre d'envoi des messages se fait à chaque fois pour trois messages consécutifs reçus par le coordinateur. Le choix d'un nombre plus grand de messages à ordonner implique l'extraction d'un grand nombre d'ordres possibles au niveau du coordinateur. La figure 18 montre l'arrivée de trois nouveaux messages au site coordinateur. Un message $\in \{x, y, z\}$ avec l'indice $i \in \{1, 2\}$ indique qu'il a été envoyé par le superviseur (site) i. Il est supposé qu'il existe seulement deux superviseurs 1 et 2.

Messages	Ordres possibles	Ordres extraits
$x_1y_2z_2$	$x_1y_2z_2$ ou $y_2x_1z_2$	x_1y_2 ou y_2x_1
$x_2y_1z_1$	$x_2y_1z_1$ ou $y_1x_2z_1$	x_2y_1 ou y_1x_2
$x_1y_2z_1$	$x_1y_2z_1$ ou $y_2x_1z_1$ ou $x_1z_1y_2$	x_1y_2 ou y_2x_1 ou $x_1z_1y_2$
$x_2y_1z_2$	$x_2y_1z_2$ ou $y_1x_2z_2$ ou $x_2z_2y_1$	x_2y_1 ou y_1x_2 ou $x_2z_2y_1$

Figure 18. *Extraction des ordres possibles au niveau du site coordinateur.*

La colonne de gauche décrit l'ordre dans lequel les messages sont reçus par le coordinateur. La colonne du milieu décrit tous les ordres possibles pouvant résulter de la réception des messages par le coordinateur. La colonne de droite décrit les ordres extraits.
La procédure consiste à attendre trois messages, à trouver tous les ordres possibles des deux premiers messages arrivés, tout en gardant le troisième message dans le même ordre jusqu'à l'arrivée de nouveaux messages, (figure 18) (Debouk et al 2000b).
L'approche proposée impose que les ensembles des événements observables par les deux sites soient disjoints. Si ce n'est pas le cas, les événements reçus en commun peuvent servir comme message de synchronisation entre les deux sites et permettre ainsi de faciliter la tâche du coordinateur. Dans le chapitre III, nous supposons le délai d'envoi des messages borné. Ceci permettra à un site de surveillance d'ordonner les messages lui arrivant moyennant une incertitude due à l'imprécision sur le délai.

II.3.4. Minimisation du coût de communication et de calcul

La communication entre les différents sous-systèmes de surveillance a un coût relatif à l'utilisation des canaux de communication et à l'effort additionnel dû au traitement de l'information envoyée aux autres sous-systèmes. Le délai de communication doit être pris en compte dans cette évaluation. Les coûts de communication et de calcul imposent un compromis entre l'objectif de la surveillance et celui de minimiser ses coûts.
Nous avons vu précédemment que les messages échangés entre les processeurs afin de pouvoir vérifier les contraintes temporelles devaient contenir une information sur l'occurrence d'événements. Pour les raisons de coût, le nombre des messages échangés doit être minimal.
Quand un événement local se produit, le sous-système de surveillance doit décider si l'événement doit être communiqué aux autres sous-systèmes de surveillance.
Etant donné un graphe de contraintes *G* et un sommet e_i, la liste des sous-systèmes de surveillance auxquels l'occurrence de l'événement e_i doit être communiquée, peut être construite en exécutant *l'algorithme du plus court chemin* sur *G* (Jahanian 1994). Le graphe résultant est appelé *le graphe du plus court chemin* et il est l'extension de *G* en rajoutant des arcs représentant les contraintes implicites. Les messages doivent aussi être envoyés en suivant ces arcs additionnels. Comme nous l'avons déjà expliqué, afin de détecter les violations le plus tôt possible, les contraintes implicites doivent être considérées.

Considérons l'exemple de la figure 19 où un graphe de contraintes *G* et le graphe du plus court chemin dérivé sont représentés.

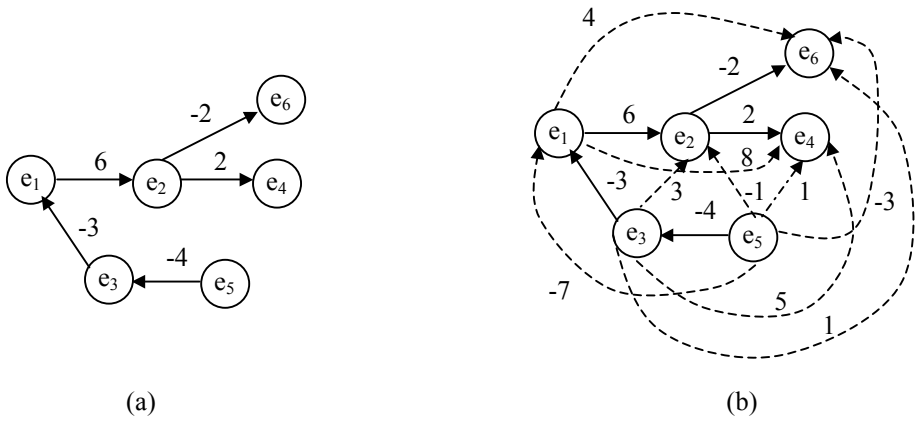


Figure 19. *(a) graphe de contrainte,(b) graphe de contrainte à plus court chemin*.

D'après le graphe du plus court chemin, si un événement e_i se produit, il doit être communiqué à tous les sous-systèmes de surveillance d'un sommet associé à e_j, s'il existe un arc de longueur positive de e_i à e_j ou un arc de longueur négative de e_j à e_i (Jahanian 1994). En effet, s'il y a un sommet e_j tel qu'il existe un chemin de longueur négative de e_j à e_i dans le graphe de contrainte alors il existe une contrainte de délai entre e_i et e_j, e_i précède e_j. La date d'occurrence de e_i doit être envoyée au sous-système de surveillance de e_j. De la même manière, s'il existe un sommet e_j tel qu'il y a un arc de poids positif de e_i à e_j, alors une existe une contrainte date limite entre e_i et e_j. Dans l'exemple de la figure 19 (b), l'occurrence de e_1 doit être envoyée aux sous-systèmes de surveillance de e_3 et de e_5, mais également aux sous-systèmes de surveillance de e_2, de e_4 et de e_6.

Il est souvent possible d'éliminer des messages redondants ou pouvant être combinés en un seul message. Dans l'exemple de la figure 19(b), l'occurrence de e_1 doit être communiquée au système de surveillance de e_4 et de e_2, et l'occurrence de l'événement e_2 doit être communiquée au sous-système de surveillance de e_4. Le poids de l'arc de e_1 à e_4 est le même que la longueur du chemin $e_1e_2e_4$. Le message du sous-système de surveillance de e_1 au sous-système de surveillance de e_4 peut être éliminé.
Le problème de minimisation du nombre de message pour un graphe de contraintes reste néanmoins difficile à traiter. En effet, l'élimination d'un nombre maximum de messages redondants est un problème NP-complet pour les graphes de contraintes possédant seulement des arcs à poids positif.

Jusqu'à présent nous nous sommes essentiellement intéressés aux contraintes de type délai et de type date limite. Or, dans la pratique ce sont des contraintes temporelles de type intervalle qui doivent être considérées dans la mesure où la durée écoulée entre deux occurrences d'événements est souvent bornée. Comme nous l'avons déjà dit, ce type de contrainte représente la conjonction (*ET*) d'une contrainte de délai et de date limite. Aussi, nous allons maintenant présenter les chroniques, qui en tant qu'ensembles d'événements de contraintes de type intervalle liant ces événements, apparaissent être un outil adéquat pour l'expression de contraintes temporelles.

II.4. Les chroniques

Une chronique est un ensemble d'événements et de contraintes temporelles liant ces événements. Etant donné un flot d'événements datés comme entrée du système de reconnaissance de chronique, celui-ci suit et reconnaît des schémas d'évolution ou modèles de chroniques spécifiés hors-ligne (Ghallab 1996, 1998). Dans les chroniques, l'aspect temporel est pris en compte à travers la spécification de contraintes temporelles entre les événements en provenance du système surveillé.

Il est nécessaire de vérifier hors-ligne la cohérence des chroniques écrites par l'utilisateur afin de permettre une reconnaissance rapide. Cette vérification consiste à compiler les chroniques en structures de données efficaces (Dousson et al 1994). Des contraintes temporelles résultant d'une propagation peuvent être rajoutées par le compilateur. La propagation des contraintes temporelles est locale à chaque chronique et se fait par un algorithme de consistance de chemins dérivé de (Dechter et 1991). Cet algorithme fournit, pour chaque chronique, le graphe complet des contraintes minimales équivalent à l'ensemble des contraintes de l'utilisateur et celles ajoutées par le compilateur.

Contraintes de l'utilisateur Graphe complet des contraintes minimales

Figure 20. *Treillis de contraintes de l'utilisateur et graphe complet des contraintes minimales.*

La figure 20 donne le treillis (graphe) des contraintes spécifiées par l'utilisateur et le graphe complet des contraintes minimales correspondant.

Dans un modèle de chronique, toutes les contraintes temporelles sont relatives. Dès qu'une date d'occurrence est connue pour un événement, tous les autres événements attendus dans la chronique deviennent contraints de façon absolue dans le temps. En plus, le système de reconnaissance de chronique maintient à jour les ensembles de dates encore possibles (compatibles avec les contraintes) pour les événements attendus, ensemble appelé *fenêtre admissible* notée *D(i)* pour un nœud i (label d'événement) (Dousson et al 1994). Au fur et à mesure du déroulement de la reconnaissance, le processus restreint les fenêtres admissibles pour les événements restants. Si les dates possibles d'un nœud i sont restreintes à celles contenues dans *D(i)*, alors cette information va être propagée dans le graphe complet.

Supposons que dans l'exemple précédant e_1 se produise à la date 2'. La chronique attend alors les événements e_2 et e_3 chacun dans des fenêtres admissibles conformes aux contraintes (figure 21).

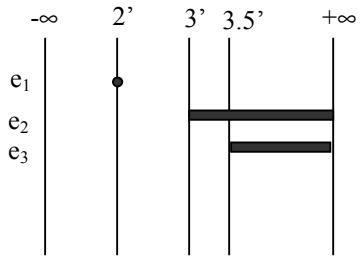


Figure 21. *Propagation de l'occurrence de e_1 sur les fenêtres des autres événements.*

Supposons maintenant que e_2 se produise à la date 4, la figure 22 est obtenue.

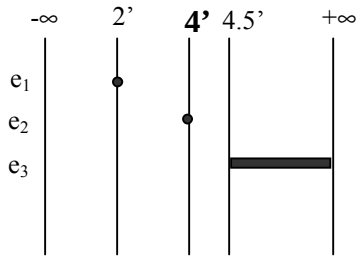


Figure 22. *Propagation de l'occurrence de e_2 sur les fenêtres des autres événements.*

A l'aide des fenêtres admissibles *D(e)*, une date limite de validité de la chronique en cours, appelée *deadline*, peut être définie. Un deadline correspond à la plus petite borne supérieure des fenêtres admissibles des événements attendus. Si cette date est dépassée, il est certain que la chronique ne pourra jamais être reconnue (Dousson et al 1994). Un symptôme de défaillance est ainsi détecté.
Nous utilisons dans la suite de la thèse l'appellation « chronique » pour désigner tout ensemble de contraintes et d'événements les constituant. Les chroniques étudiées peuvent être des chroniques représentant le comportement normal ou anormal du système surveillé.

II.5. Conclusion

Ce chapitre présente les principaux aspects relatifs à la distribution de la commande et de la surveillance de systèmes distribués à événements discrets. La commande, distribuée sur plusieurs sous-systèmes de commande, gère l'échange de messages entre les différents sous-systèmes afin de réaliser les objectifs de la commande. La surveillance, distribuée en plusieurs sous-systèmes de surveillance, introduit la notion d'observations locales comme étant le sous-ensemble d'événements du procédé observé par un sous-système de surveillance. Nous avons vu que les tâches de surveillance ne pouvaient être réalisées sans que les différents sous-systèmes s'échangent un certain nombre d'information.

Pour la fonction détection, les aspects temporels liés au système surveillé sont essentiels et doivent être intégrés dans la définition des mécanismes de détection, mais nous avons montré que cette distribution pose des problèmes liés particulièrement à la vérification de contraintes temporelles mettant en jeu des événements se produisant sur des sites différents. En fait, afin de surveiller une telle contrainte distribuée, de nouveaux mécanismes doivent être élaborés comme par exemple la synchronisation d'horloge.

De manière générale, la surveillance et la commande distribuées ont fait apparaître des problèmes notamment de réordonnancement des messages reçus, de commande d'une section

critique en présence de délai de communication et de minimisation du coût de communication et de calcul.
Pour finir, nous avons rapidement présenté les chroniques et les mécanismes qui leurs sont associés dans l'idée de les utiliser pour la résolution des problèmes liés à la surveillance distribuée.

A l'issue de ce chapitre, il apparaît nécessaire de s'intéresser à la décomposition d'une architecture de surveillance centralisée afin d'obtenir une architecture distribuée. Ceci englobe la décomposition du système surveillé et la distribution des mécanismes de surveillance. Pour cela, il faut définir précisément les limites de chacun des sous-systèmes surveillés, issus de la décomposition du système global, en définissant les composants (ressources capteurs, etc.) faisant partie de chaque sous-système et le sous-ensemble d'événements qu'ils génèrent.

Une fois la décomposition du système surveillé réalisée, il convient de s'intéresser à la distribution de la chronique représentant les différentes contraintes temporelles liant les événements générés par le système surveillé. Ceci est réalisé en distribuant les contraintes temporelles sur les sous-systèmes surveillés en tenant compte des événements observables par chacun. Cette distribution donne lieu à la notion de sous-chronique. La vérification de toutes les contraintes temporelles constituant une sous-chronique implique la vérification de celle-ci et la vérification de toutes les sous-chroniques implique la vérification de la chronique.

La vérification des contraintes impose aux différents sous-systèmes de surveillance de communiquer lorsque l'information nécessaire n'est pas totalement disponible localement. Aussi un protocole de communication impliquant ces sous-systèmes doit être défini. L'expéditeur, le destinataire et les messages échangés doivent par conséquent être explicités. L'aspect coopération entre superviseurs devrait aussi être étudié dans le sens où plusieurs sous-systèmes de surveillance peuvent être amenés à coopérer afin de vérifier une contrainte. Dans ce contexte, il est nécessaire de prendre en compte les délais de communication, supposés négligeables jusque là. En effet, les délais de communication engendrent une imprécision sur la vérification des contraintes temporelles due à l'incertitude sur la valeur de ce délai. Ainsi les mécanismes de vérification des contraintes temporelles doivent être adaptés afin de supporter un délai de communication variable et borné. Par ailleurs, il semble pertinent de traiter le cas où certains événements sont non observables par les sous-systèmes de surveillance alors qu'ils font partie des contraintes temporelles à surveiller. L'utilisation d'événements intermédiaires qui sont observables et permettent de vérifier une contrainte incluant un événement non observable peut être une solution.

Ces différents points font l'objet du troisième chapitre de notre mémoire, que nous présentons maintenant.

Chapitre III

La fonction détection distribuée : Prise en compte des délais

III.1. Introduction

La plupart des travaux développés récemment ont adopté une architecture distribuée pour la surveillance. La surveillance distribuée est une solution aux nombreux problèmes posés par les architectures classiques (centralisée, hiérarchique…), d'un point de vue complexité, tolérance aux fautes, réduction des coûts, etc. L'adoption d'une structure distribuée pour la surveillance a engendré la distribution des différentes fonctions participant à la réalisation de la tâche de surveillance. La fonction détection distribuée a pour rôle de détecter toute violation des spécifications de bon fonctionnement du système surveillé de manière locale ou de manière distribuée par communication et coopération entre différents superviseurs. Le diagnostic distribué qui s'intéresse à la localisation et à l'identification de l'origine de la défaillance se trouve facilité grâce à l'observation partielle du système surveillé engendrée par l'architecture distribuée, d'autant plus qu'en général, si un composant de la fonction détection distribuée détecte un symptôme de défaillance au niveau de la partie du procédé qu'il surveille, l'origine de la défaillance se situe au niveau de ce sous-procédé. Les fonctions de suivi, de pronostic, de décision et de reprise doivent aussi être distribuées afin de s'intégrer à l'architecture de surveillance adoptée. Jusqu'à nos jours, peu de travaux ont traité la distribution et le développement de mécanismes distribués pour ces fonctions. Parmi ces travaux nous citons les travaux de (Da Silveira et al 2003) dont le but est proposer une procédure systématique pour obtenir un modèle distribué avec redondance partielle à partir d'un modèle centralisé spécifié par réseaux de Petri. Le résultat de la distribution est un ensemble de sous-modèles qui décrivent le comportement d'une (ou de plusieurs) ressource(s) et toutes ses interfaces avec les autres ressources. Ces modèles font partie d'un module composé aussi par les fonctions de surveillance-commande.

Nous nous intéressons, plus particulièrement dans cette thèse, à la distribution de la fonction détection avec prise en compte des délais. Ces délais peuvent être des délais de communication entre sous-systèmes de détection ou des délais représentant des durées opératoires. La décomposition physique du système à surveiller est naturelle dans le sens où un procédé est composé de plusieurs éléments (ressources, processus, opérateur, etc.). En se basant sur cette décomposition physique, le modèle de comportement temporel, que nous avons appelé chronique,

est distribué sur plusieurs sous-systèmes de détection. La méthode de distribution d'une chronique, les protocoles de communication et de coopération entre les sous-systèmes, et les mécanismes de détection doivent alors être spécifiés.

III.2. Définitions

Nous reprenons ici quelques définitions utiles à la compréhension de la suite de notre travail.

Evénement : un événement correspond à un stimulus auquel le système doit réagir en changeant d'état. Il peut survenir suite à un message du procédé ou à la terminaison d'un processus.

Evénement reçu : un événement *e* appartenant à l'ensemble Σ des événements du système surveillé est dit reçu par un site de surveillance s'il peut être daté par celui-ci.

Date d'occurrence : c'est la mesure à l'aide d'une horloge du système de supervision de la coordonnée temporelle d'un événement reçu. Dans un système distribué constitué de différents sites de surveillance (superviseurs) S_1, S_2,..S_n, une horloge, dite *horloge locale*, est associée à chacun des sites.
Un événement est donc *daté* dans le repère temporel local, lié au site de surveillance S_i le recevant, par la fonction *"Occurrence"* notée *O* et définie ci-dessous:

$$O : \Sigma \rightarrow Q^+$$
$$e_i \rightarrow O(e_i)$$

Chaque événement reçu étant daté dans le repère temporel du site de surveillance S_i qui le reçoit. Nous distinguons Σ_{di} la liste des événements datés par un site *i* et Σ_{ndi} la liste des événements non datés par ce même site. Initialement, la liste Σ_{di} est vide pour tout site *i* et la liste Σ_{ndi} contient tous les événements de Σ. Si un événement $e \in \Sigma_{ndi}$ est reçu par un site alors cet événement devient daté par ce site et sera inséré dans liste Σ_{di} et supprimé de la liste Σ_{ndi}. Il peut s'agir d'un événement *e* en provenance du système surveillé ou d'un autre site de surveillance.

Contraintes entre événements : une contrainte est une relation impliquant une durée liant des événements. Une contrainte n'est donc jamais liée à une référence temporelle (horloge locale) ni à l'architecture de mise en œuvre. Il s'agit d'une relation traduisant une propriété intrinsèque de l'application considérée. Une contrainte peut par exemple, traduire une durée de transport entre deux sites d'une réalisation. Ces contraintes peuvent être des relations binaires ou n-aires.

Relation binaire : deux types de contraintes peuvent être distingués,

- *la contrainte de précédence* : elle est définie par $O(e_i) < O(e_j)$ et exprime que l'événement e_j doit arriver après l'événement e_i.
- *la contrainte de type intervalle* : elle est définie par $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$ notée C_{ji} avec $d_{j,i}$ et $f_{j,i} \in Q^+$ et exprime que l'événement e_j doit arriver après l'événement e_i dans l'intervalle $[d_{j,i}, f_{j,i}]$.

Relation n-aire : il s'agit de contrainte de type *fenêtre d'admissibilité* définie par:

$d_i \leq \min_j(O(e_i)-O(e_j)) \leq f_i$, notée D_i, avec $d_i, f_i \in Q^+$;

{e_j} est l'ensemble de tous les événements liés avec e_i par une contrainte du type intervalle ou de précédence. Ce type de contrainte exprime que l'événement e_i doit arriver, après l'occurrence du dernier événement le précédent, dans l'intervalle $[d_i, f_i]$.

Chronique : une chronique est composée de l'ensemble des événements Σ et d'un ensemble de contraintes liant ces événements.

Nous allons maintenant présenter en détails la méthode de distribution du système de surveillance et notamment de la fonction détection.

III.3. Décomposition de l'architecture de surveillance

La surveillance distribuée utilise les avantages de l'intelligence distribuée (Dilts et al 1991) (Duffie et al 1996) et repose sur une architecture divisant les responsabilités de surveillance sur plusieurs sites ou sous-systèmes de surveillance ou superviseurs S_i coopérants (figure 1).

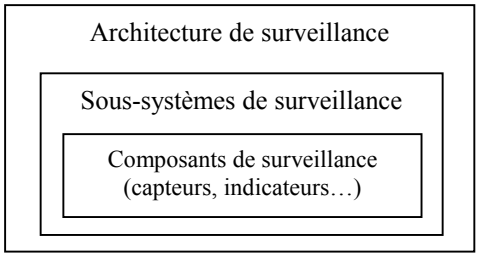


Figure 1. *Décomposition de l'architecture de surveillance.*

Chaque site gère une *zone physique* définie par un ensemble d'éléments du système surveillé tel que capteurs, indicateurs et ressources. Les éléments d'une zone fonctionnent de manière quasi-indépendante par rapport à d'autres éléments appartenant à des zones différentes toutefois avec une faible interaction. Cette interaction est fonctionnelle dans le sens où la valeur produite par les capteurs dépend de celles précédemment produites par d'autres capteurs appartenant à des zones différentes. De même, une ressource agissant sur un produit, doit attendre que d'autres ressources, dans des zones différentes, effectuent leurs tâches respectives sur le produit considéré. D'autres formes d'interaction entre sites de surveillance peuvent être considérées notamment pour la synchronisation entre ressources. La synchronisation de ressources est un rendez-vous entre plusieurs ressources, permettant à une opération de s'exécuter lorsque celles-ci sont disponibles.

Dans le cadre de la surveillance distribuée nous nous intéressons exclusivement aux capteurs qui sont à l'origine de l'information nécessaire aux mécanismes de surveillance. Les zones obtenues peuvent se chevaucher facilitant ainsi la résolution des problèmes de la localisation et du diagnostic d'un symptôme de défaillance (figure 2). Ceci implique qu'un capteur peut appartenir à plus d'une zone à la fois. L'état de ces capteurs « en commun » est ainsi communiqué à plusieurs sous-systèmes de surveillance créant ainsi une redondance pour des mécanismes de surveillance incluant ces capteurs. Si un sous-système de surveillance est dans un état d'échec suite à une défaillance matérielle, les autres sous-systèmes de surveillance gérant les capteurs en commun peuvent assurer la continuité du service de surveillance.
Chaque sous-système de surveillance se compose de six fonctions : la détection, le diagnostic, le pronostic, le suivi, le recouvrement et l'urgence (Zamai 1997). Le rôle de ces fonctions a été détaillé dans le chapitre I. Dans le cadre de ce travail, nous nous limitons à la fonction détection.

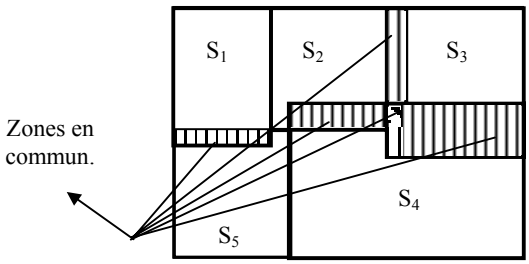


Figure 2. *Subdivision du système surveillé.*

Nous appelons dans la suite, *superviseur*, un sous-système de détection et les mécanismes qui lui sont associés. Un superviseur S_i, i = 1..n, réalise les fonctionnalités de la détection à savoir l'identification d'un symptôme de défaillance dans la zone considérée au moyen de la réception d'un sous-ensemble d'événements $\Sigma_i \subset \Sigma$, c'est à dire les événements directement issus des capteurs de la zone. Nous supposons nuls, les délais de transmission dus à l'envoi des occurrences des événements de la zone vers le superviseur dédié. Si le sous-modèle de bon fonctionnement associé à la zone considérée est non respecté, le superviseur détecte l'apparition d'un symptôme de défaillance. La notion de sous-modèle de bon fonctionnement va être étudiée dans la suite de ce chapitre.

Exemple d'application:

Reprenons l'exemple d'une station de remplissage de bouteilles présenté à la figure 3.

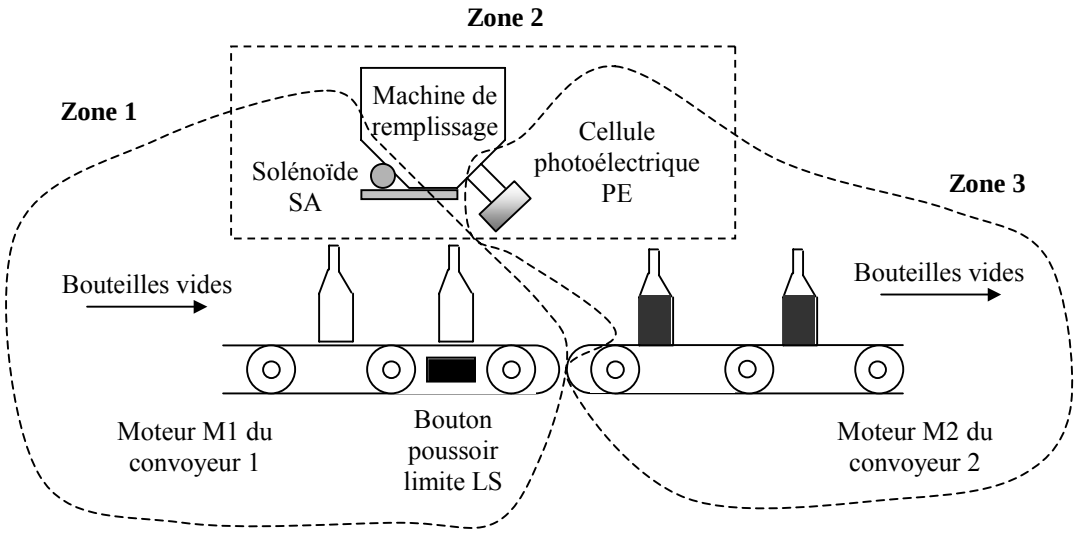


Figure 3. *Exemple de décomposition d'un système de remplissage de bouteilles en zones.*

Une décomposition naturelle de ce système de remplissage de bouteilles conduit à considérer indépendamment les éléments du système qui sont : le convoyeur 1 auquel sont associés l'actionneur *M1* et le capteur *LS*, le convoyeur 2 auquel est associé l'actionneur *M2* et la machine de remplissage à laquelle sont associés le capteur *PE* et l'actionneur *SA*. Les deux convoyeurs et la machine de remplissage sont associés chacun à une zone à surveiller délimitée par des traits en

pointillés sur la figure 3. Des chevauchements entres ces zones ont été introduits. En effet, l'actionneur *SA* appartient aux zones 1 et 2, et le capteur *PE* appartient aux zones 2 et 3.
Comme nous l'avons vu au paragraphe II.2.2.2., il existe des contraintes ou relations entre les différents événements générés par le système surveillé, que sont : PE activée, LS activé, M1 arrêté, SA actionné, etc. Ces contraintes sont exprimées à l'aide d'un modèle de comportement ou d'un modèle temporel. Ce modèle doit être distribué en sous-modèles sur les différents superviseurs. Chaque superviseur est associé à une zone à surveiller, (figure 4).

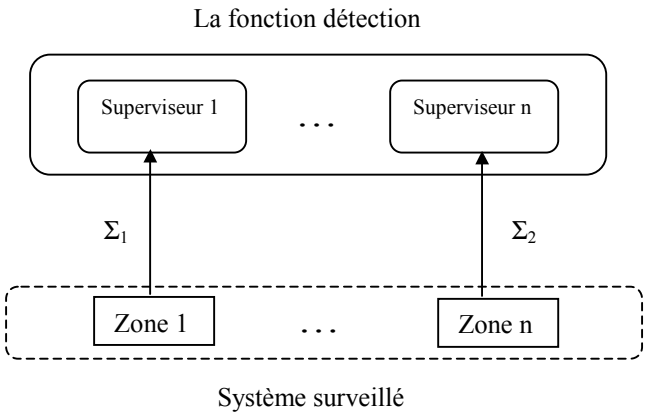


Figure 4. *Le couplage zone$_i$ - superviseur$_i$.*

En considérant le modèle temporel, la distribution des contraintes le composant met en évidence deux types de contraintes temporelles : *les contraintes locales* et *les contraintes globales*.

III.3.1. Contraintes locales, contraintes globales

La fonction détection a pour rôle de reconnaître des évolutions du système surveillé à travers la vérification des différentes contraintes temporelles liant des événements du système surveillé. Compte tenu de la distribution de la fonction détection, il apparaît deux familles de contraintes :

- *les contraintes locales* qui lient des événements de l'ensemble Σ_i associé à un superviseur *i*.
- *les contraintes globales* sont les contraintes qui ne sont pas locales, (figure 5).

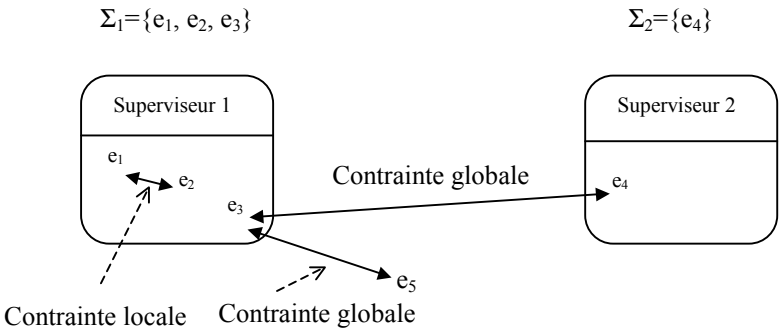


Figure 5. *Contraintes locales et contraintes globales.*

Dans la figure 5, considérons le cas où l'événement e_4 est daté par le superviseur 2. Si l'occurrence de l'événement e_4 est renvoyée par le superviseur 2 au superviseur 1 sans délai de communication, alors celui-ci est reçu par le superviseur 1 qui peut ainsi dater son occurrence et vérifier la

contrainte le liant à e_3. Dans le cas où un délai de communication existe, le superviseur 1 reçoit du superviseur 2 une image de l'événement e_4. En datant l'occurrence de l'image reçue et moyennant des mécanismes que nous développerons plus tard, le superviseur peut ainsi vérifier la contrainte considérée. Notons enfin que l'événement $e_5 \notin \Sigma_1 \cup \Sigma_2$ ne peut pas être reçu par les deux superviseurs et par conséquent aucun de ces superviseurs ne peut dater son occurrence.

III.3.2. Notion de sous-chronique et de reconnaissance de chronique

Comme nous l'avons introduit précédemment, une chronique est un ensemble d'événements et de contraintes liant ces événements. La distribution du modèle temporel représenté par une chronique engendre la distribution de la chronique en plusieurs *sous-chroniques* associées chacune à un superviseur, (figure 6).

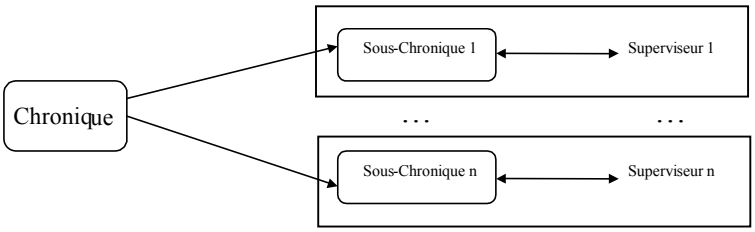


Figure 6. *Décomposition de la chronique en sous-chroniques.*

Nous avons dit précédemment que l'architecture de détection distribuée, constituée de différents superviseurs S_1, S_2,…S_n, crée à partir de l'ensemble des événements Σ des listes de sous-ensembles non nécessairement disjoints d'événements $\Sigma_{d1},\Sigma_{nd1},\Sigma_{d2},\Sigma_{nd2},..,\Sigma_{dn},\Sigma_{ndn}$. Par contre, la distribution d'une chronique partitionne l'ensemble des contraintes la constituant sur les *n* superviseurs.
Une sous-chronique associée à un superviseur S_i est de ce fait constituée des contraintes locales liant des événements de Σ_i et de contraintes globales impliquant au moins un événement de Σ_i.

A noter que pour une architecture de détection et une chronique données, la distribution d'une contrainte globale peut être faite de différentes manières. En effet, une contrainte globale peut être rattachée à tout superviseur S_i qui a un des événements la constituant dans son ensemble Σ_i. Nous reviendrons sur ce point dans le paragraphe suivant.
A partir de là, nous introduisons la notion de reconnaissance de chronique. Une chronique est dite *reconnue* si toutes les contraintes qui la composent sont vérifiées compte tenu des durées séparant les occurrences des événements la constituant. La reconnaissance d'une sous-chronique se définit de la même manière. Quand une chronique est décomposée en *n* sous-chroniques, la reconnaissance des *n* sous-chroniques assure que la chronique est aussi reconnue.

III.3.3. Distribution des contraintes temporelles

Soit ζ l'ensemble des contraintes d'une chronique. La distribution de la chronique en plusieurs sous-chroniques implique la décomposition de l'ensemble ζ en plusieurs sous-ensembles ζ_i, i=1..n, associés chacun à une sous-chronique telle que $\cup_i \zeta_i = \zeta$. Cette distribution des contraintes fait apparaître, comme nous l'avons déjà évoqué, deux types de contraintes, les contraintes locales et les contraintes globales. Une contrainte locale est rattachée au superviseur recevant de la zone surveillée les occurrences des événements qui la composent. Une contrainte globale peut être rattachée à n'importe quel superviseur datant un des événements la constituant. Si

elle est rattachée à tous les superviseurs datant un de ses événements, la contrainte globale serait ainsi surveillée par plusieurs superviseurs. Cette solution présente un avantage certain lié à l'amélioration de la tolérance aux fautes dû à la duplication de la contrainte à surveiller sur plusieurs superviseurs. L'inconvénient de cette solution est d'être coûteuse en nombre de messages échangés entre superviseurs. Une deuxième solution, vers laquelle nous nous sommes orientés, consiste à rattacher une contrainte globale à un seul superviseur.

Nous proposons de distribuer une contrainte globale selon le schéma suivant :

- Pour une contrainte de précédence, $O(e_i) < O(e_j)$, c'est le superviseur S_j recevant l'occurrence de e_j en provenance du procédé, c'est à dire $e_j \in \Sigma_j$, qui surveille la contrainte. Si $e_j \notin \bigcup_h \Sigma_h$, h=1..n (l'occurrence de e_j n'est renvoyée par aucun des capteurs du système), c'est le superviseur S_i qui reçoit l'occurrence de e_i du procédé qui surveille la contrainte,

- Pour une contrainte de type intervalle, $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$, c'est le superviseur recevant e_j en provenance du procédé qui surveille la contrainte. De même, si $e_j \notin \bigcup_h \Sigma_h$, h=1..n, c'est le superviseur S_i recevant l'occurrence de e_i du procédé qui surveille la contrainte.

- Pour une contrainte de type fenêtre d'admissibilité, $d_j \leq \min_{i,\, i=1..n}(O(e_j)-O(e_i)) \leq f_j$, c'est le superviseur recevant e_j en provenance du procédé qui surveille la contrainte. Si $e_j \notin \bigcup_h \Sigma_h$, h=1..n, ce sont les superviseurs S_i, i=1..n, recevant les occurrences des e_i en provenance du procédé qui surveillent la contrainte.

Comme nous l'avons déjà dit, un superviseur a besoin de dater tous les événements de la contrainte pour pouvoir la vérifier en calculant les durées la définissant. Dans le cas d'une contrainte globale, le superviseur a besoin de recevoir l'occurrence de tous les événements non datés constituant la contrainte pour pouvoir la vérifier. Lorsque un événement $e \notin \bigcup_h \Sigma_h$, h=1..n, il ne peut pas être reçu par les superviseurs et son occurrence ne peut pas par conséquent être datée par aucun d'entre eux. L'occurrence d'un *événement intermédiaire* $e_k \in \Sigma_k$ lié avec *e* par une durée bornée connue peut être utilisée pour vérifier la contrainte impliquant *e*. Lorsque cette durée connue représente une durée de communication, l'événement intermédiaire s'appelle image de e. Nous revenons sur ce point dans la suite du chapitre.
Nous venons de voir que la vérification d'une chronique suppose la vérification des sous-chroniques associées et de ce fait la vérification des contraintes temporelles. Dans le cas de contraintes temporelles globales, cette vérification suppose une communication entre les superviseurs que nous allons maintenant étudier.

III.3.4. Le protocole de communication entre superviseurs

Les contraintes faisant partie d'une sous-chronique pouvant être globales, la communication entre différents superviseurs afin de les vérifier est donc nécessaire. En effet, les superviseurs doivent s'échanger les occurrences de certains événements afin de pouvoir vérifier des contraintes globales. Les questions qui se posent alors sont : quel superviseur communique et avec qui ? qu'est ce qu'un superviseur communique à un autre ? quand est ce qu'un superviseur communique avec un autre ?

La communication consiste au renvoi par un superviseur d'occurrences d'événements reçus en provenance du système surveillé, c'est à dire de Σ_i, (figure 7). Un superviseur doit donc garder une liste de superviseurs pour chaque événement reçu du système surveillé. Cette liste représente l'ensemble des superviseurs à qui l'occurrence de cet événement sera renvoyée.

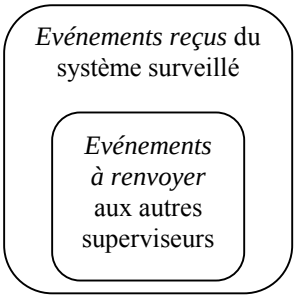


Figure 7. *Evénements reçus et événements renvoyés.*

A un événement reçu doit donc être associé le(s) superviseur(s) destinataire(s). Formellement, soit *e* un événement reçu par S_i et *S* l'ensemble des superviseurs. Notons $S^* \cup \{\varnothing\}$ l'ensemble de tous les sous-ensembles de superviseurs. Soit $D \in S^* \cup \{\varnothing\}$, *D* est un sous-ensemble de superviseurs. Une *liste de renvoi* L_r est un ensemble de couples (e, D) tel que *D* représente l'ensemble des superviseurs à qui l'événement *e* sera renvoyé. Si un événement reçu n'est pas à renvoyer par un superviseur alors l'ensemble $D = \varnothing$ lui est associé.

Signalons enfin qu'un superviseur doit renvoyer l'occurrence d'un événement le plus tôt possible. Par la suite, nous traitons le cas où il existe, un délai entre le renvoi et la réception de l'occurrence d'un événement reçu, ou un délai (durée) entre un événement $e \notin \bigcup_h \Sigma_h$, h=1..n, et un événement intermédiaire.

III.3.5. Notion de délai

Comme les occurrences des événements doivent être échangées entre les différents superviseurs, les délais (Debouk et al 2000) (Tripakis 2002) sont une donnée à prendre en compte pour la vérification des contraintes temporelles. Il existe deux types de délai, *le délai de communication* entre superviseurs et *le délai opératoire*.

III.3.5.1. Délai de communication

Le délai de communication se manifeste par le fait que les messages envoyés d'un superviseur à l'autre prennent un certain temps avant d'arriver au superviseur destinataire (kandasamy et al 2002). De ce fait, un superviseur reçoit l'occurrence d'un événement en provenance d'un autre superviseur avec un certain délai. Dans la figure 8, e_k est l'image de e_i reçue par le superviseur 2.

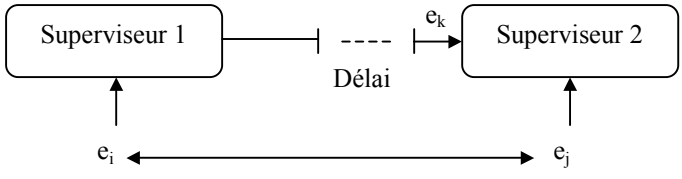


Figure 8. *Le délai de communication entre superviseurs.*

La vérification de la contrainte liant e_i et e_j doit donc tenir compte du délai de communication associé à l'envoi de l'événement e_i par le superviseur 1 au superviseur 2. Ceci est réalisé par la mesure de la durée qui s'est écoulée entre la réception de e_k et de celle de e_j par le superviseur 2. L'événement e_k est dans ce cas l'événement intermédiaire.

III.3.5.2. Délai opératoire

Le délai opératoire est une durée de fonctionnement ou de transfert de ressource. Cette durée sépare l'occurrence d'un événement qui ne peut pas être reçu par les superviseurs et de celle d'un événement intermédiaire qui est reçu par un superviseur. Supposons le cas d'une contrainte de type intervalle, $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$, à vérifier par le superviseur 2 recevant l'occurrence de l'événement e_j, (figure 9).

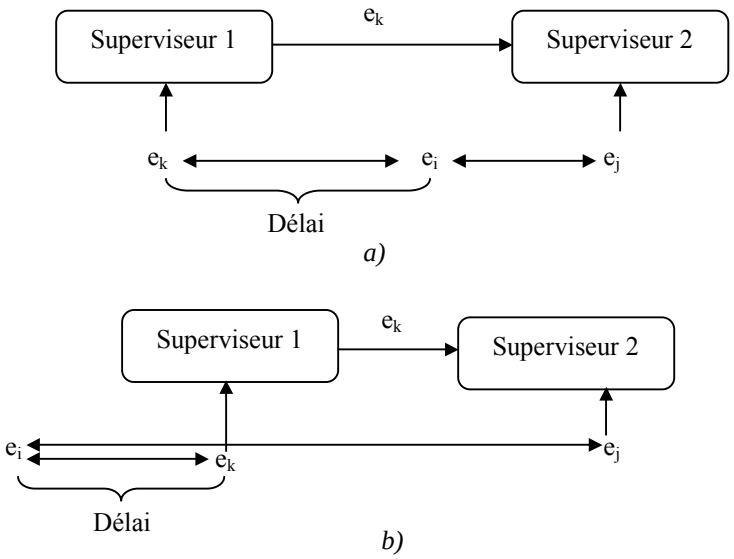


Figure 9. *Le délai opératoire.*

La vérification de la contrainte globale de type intervalle C_{ji} nécessite la réception par le superviseur 2 de l'occurrence de l'événement e_i laquelle ne peut pas être reçue par aucun des superviseurs. Si l'occurrence de e_i est reçue par un superviseur *p*, p≠2, le superviseur *p* renvoie l'occurrence de e_i au superviseur 2 et celui-ci devient capable de vérifier la contrainte C_{ji}.

Quand l'occurrence de e_i ne peut pas être renvoyée au superviseur 2, parce que e_i ne peut être reçu par aucun superviseur, le vérification de la contrainte C_{ji} sera faite à partir d'un événement intermédiaire e_k dont l'occurrence est liée à celle de e_i par un délai appartenant à un intervalle à bornes positifs *a* et *b*, (figure 9). La contrainte liant les événements e_j et e_i est la contrainte à vérifier sachant un délai opératoire entre les occurrences des événements e_k et e_i.
Signalons que lorsque l'occurrence de e_j n'est reçue par aucun des superviseurs, c'est le superviseur 1 recevant l'occurrence de e_i qui vérifie la contrainte. Dans ce cas, les mécanismes permettant de vérifier la contrainte sont symétriques à ceux détaillés au dessus.

La prise en compte des délais opératoires intervient dans le cas de la surveillance des chaînes logistiques de l'entreprise étendue (Cohen et al 97), des systèmes de transport et des systèmes manufacturiers (surveillance des opérations d'usinage, surveillance du transport de pièces via des convoyeurs ou des robots mobiles).

Nous abordons par la suite l'étude de la vérification de contraintes temporelles appartenant à des sous-chroniques en prenant en compte les délais. Les techniques présentées sont valables pour tout système dans lequel l'incertitude (Lehner et al 1996) sur les délais de communication ou opératoires et les tolérances sur les contraintes temporelles sont du même ordre. Quand les délais sont négligeables par rapport aux durées des contraintes, ceci revient à appliquer les mécanismes classiques de la vérification de contraintes sans tenir compte d'un délai quelconque.

III.4. Reconnaissance d'une sous-chronique avec prise en compte des délais

La reconnaissance d'une évolution du système surveillé implique la reconnaissance des sous-chroniques associées à cette évolution. Nous allons étudier dans ce paragraphe, la vérification d'une contrainte temporelle (Jahanian et al 94) globale au travers d'un test d'appartenance d'une durée mesurée à un ensemble de plages de valeurs. L'étude présentée aborde la vérification des trois types de contraintes introduites précédemment.

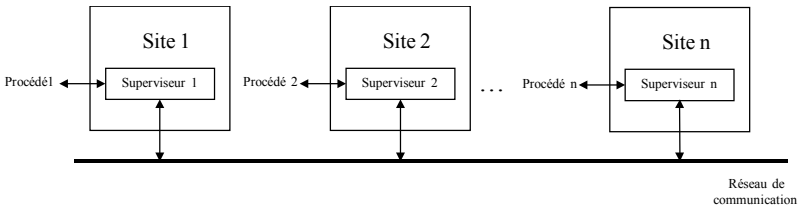


Figure 10. *Architecture de surveillance distribuée.*

La figure 10 représente l'architecture de surveillance distribuée considérée. Nous noterons $\Delta \in [\delta_m, \delta_M]$; $\delta_m, \delta_M \in Q^+$, le délai de communication existant entre les différents sites ou le délai opératoire.

Nous allons maintenant nous intéresser à la prise en compte de ce délai Δ au niveau de la vérification des différentes contraintes. Le cas d'une contrainte de type intervalle sera détaillé en premier dans la mesure où les deux autres contraintes (fenêtre d'admissibilité et précédence) peuvent être ramenées à une contrainte de ce type, comme nous le verrons ultérieurement.

III.4.1. Prise en compte du délai de communication dans la vérification des contraintes

Dans ce cas, nous nous intéressons à la prise en compte du délai de communication dans la vérification d'une contrainte globale.

III.4.1.1. Cas de contrainte de type intervalle

Considérons le cas d'un système à deux superviseurs S_1 et S_2. Supposons que l'événement e_j soit reçu par le superviseur S_1 et que e_i soit reçu par le superviseur S_2, (figure 11).

Nous étudions seulement la surveillance de la contrainte globale C_{ji} liant $O(e_j)$ et $O(e_i)$ par le superviseur 1. Soit e_k un événement envoyé par le superviseur 2 et reçu par le superviseur 1. Les événements e_j et e_k sont tous deux reçus par le superviseur 1 et donc datés sur ce même site. En plus, S_1 possède la connaissance des bornes *min* et *max* de la durée Δ qui s'est écoulée entre l'événement e_k qu'il a daté et e_i daté par S_2.

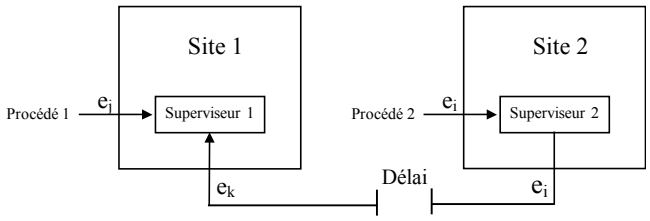


Figure 11. *Echange de données entre les deux superviseurs.*

Connaissant la durée entre les dates d'occurrences $O(e_j)$ et $O(e_k)$, notre objectif est de vérifier la contrainte C_{ji} : $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$ (1) en tenant compte de la durée Δ.

Nous pouvons écrire que :

$$O(e_j) - O(e_i) = (O(e_j) - O(e_k)) + \Delta = (O(e_j) - O(e_k)) + (O(e_k) - O(e_i)),$$

Soit encore puisque $\delta_m \leq \Delta \leq \delta_M$ (2)

$$O(e_j) - O(e_k) + \delta_m \leq O(e_j) - O(e_i) \leq O(e_j) - O(e_k) + \delta_M \text{ (3)}$$

et $O(e_j) - O(e_k) \in]-\infty, +\infty[$.

L'inégalité (3) est l'expression de la contrainte à vérifier, mais vue par le superviseur S_1.
La surveillance de la contrainte de type intervalle C_{ji} consiste alors à chercher à partir des durées mesurables $O(e_j) - O(e_k)$, les durées $O(e_j) - O(e_i)$ qui vérifient à la fois :

$$\begin{cases} d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i} \text{ (1)} \\ O(e_j) - O(e_k) + \delta_m \leq O(e_j) - O(e_i) \leq O(e_j) - O(e_k) + \delta_M \text{ (3)} \end{cases}$$

Une représentation graphique des deux contraintes dans le même plan nous conduit à la figure 3. Dans le plan $O(e_j) - O(e_k)$, $O(e_j) - O(e_i)$ les inégalités (1) et (3) définissent chacune une bande ou région. Les durées recherchées appartiennent à l'intersection de ces deux bandes qui définit un polygone noté PO.

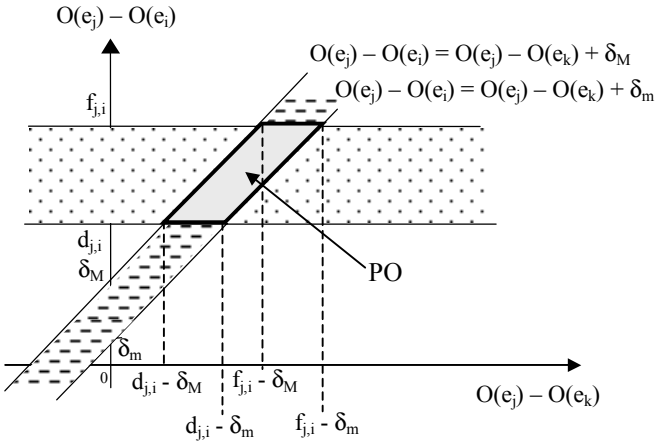


Figure 12. *Représentation graphique des contraintes (1) et (3).*

La position de *PO* dépend de la durée $O(e_j) - O(e_k)$ mais également des positions des bornes de la durée Δ par rapport aux bornes $d_{j,i}$ et $f_{j,i.}$. Un cas quelconque est présenté figure 12.

Dans la mesure où il existe une incertitude sur la durée de communication Δ d'un superviseur à un autre, à une date d'occurrence $O(e_k)$ enregistrée par S_1 correspond toute une plage de valeurs possibles et équiprobables (car nous n'avons pas d'informations préalables sur les plages d'occurrence de e_i), pour la date d'occurrence $O(e_i)$ au niveau de S_2. Donc à une mesure de la durée $O(e_j) - O(e_k)$ correspond une plage de valeurs possibles pour la durée $O(e_j) - O(e_i)$ de longueur égale à β, (figure 13). Parmi ces durées possibles seules celles appartenant à *PO* permettent de vérifier C_{ji}.

Notre objectif est alors de quantifier parmi l'ensemble des durées possibles $O(e_j) - O(e_k)$ celles qui permettent de vérifier la contrainte $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$. Cette quantification permet de définir une mesure de la possibilité (Cardoso 1990) avec laquelle la contrainte $C_{ji,}$ est vérifiée.

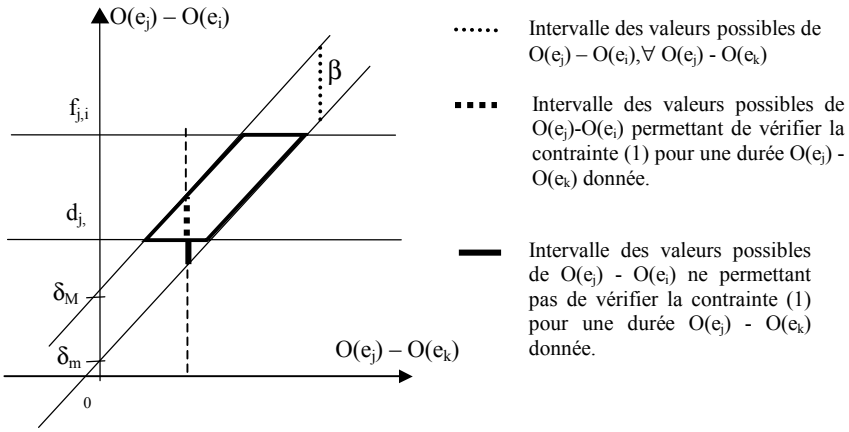


Figure 13. *Les valeurs possibles pour la durée $O(e_j) - O(e_i)$.*

Selon la durée $O(e_j) - O(e_k)$ mesurée, trois cas sont à considérer : l'inclusion, l'intersection et la disjonction, figure 14.

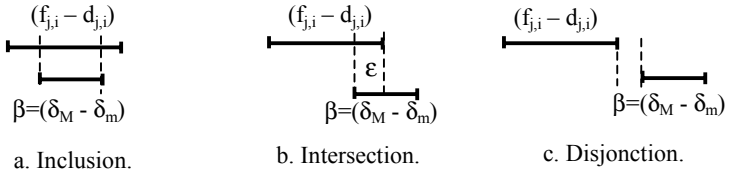
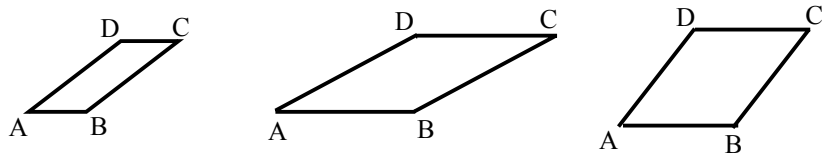


Figure 14. *Types d'intersections entre deux intervalles.*

Dans le cas a) de la figure 14, l'intersection entre les deux intervalles est de longueur $\min(\delta_M - \delta_m, f_{j,i} - d_{j,i})$. Dans le cas b), l'intersection entre les deux intervalles est de longueur $\varepsilon \in [0, \min(\delta_M - \delta_m, f_{j,i} - d_{j,i})[$. Dans le cas c), cette intersection est de longueur nulle.

La mesure de la possibilité s'exprime à partir du rapport de la quantité de durées $O(e_j) - O(e_i)$ acceptables vérifiant la contrainte $C_{ji,}$ sur l'ensemble des durées $O(e_j) - O(e_i)$ possibles (de longueur β, avec $\beta = (\delta_M - \delta_m)$), figure 13.



$\delta_M - \delta_m < f_{j,i} - d_{j,i}$
$x_B < x_D$
Type 1

$\delta_M - \delta_m = f_{j,i} - d_{j,i}$
$x_B = x_D$
Type 2

$\delta_M - \delta_m > f_{j,i} - d_{j,i}$
$x_B > x_D$
Type 3

Figure 15. *Représentation graphique du polygone PO.*

Pour une durée Δ quelconque, $\delta_m \leq \Delta \leq \delta_M$, PO est défini par les quatre points A, B, C et D de coordonnées respectives $(d_{j,i}- \delta_M , d_{j,i})$, $(d_{j,i}- \delta_m, d_{j,i})$, $(f_{j,i} - \delta_m, f_{j,i})$ et $(f_{j,i} - \delta_M, f_{j,i})$. Nous noterons par la suite x_i l'abscisse du point i dans le plan de notre étude. La figure 15 montre les différentes formes du polygone PO selon la comparaison entre les durées $(f_{j,i} – d_{j,i})$ et $(\delta_M - \delta_m)$.

A partir des résultats précédents (figure 15) nous concluons alors que :

- le cas a) de la figure 14 correspond aux valeurs de $O(e_j) – O(e_k)$ appartenant à $[x_B,x_D]$ (respectivement $[x_D,x_B]$) si le polygone PO est de type 1 ou 2 (respectivement de types 2 ou 3). Soit plus généralement $O(e_j) – O(e_k)$ appartient à $[\min(x_B,x_D), \max(x_B,x_D)]$.
- le cas b) correspond aux valeurs de $O(e_j) - O(e_k)$ appartenant à $[x_A, \min(x_B,x_D)] \cup [\max (x_B,x_D), x_C]$.
- le cas c) de la figure 14 correspond aux valeurs de $O(e_j) – O(e_k)$ appartenant à $]-\infty, x_A [\cup] x_C, +\infty[$.

Pour caractériser ce problème, supposons $O(e_j) – O(e_k) \in P$, où P est l'univers du discours. Soit *A* l'ensemble flou des valeurs de P permettant de vérifier la contrainte C_{ji}. $\mu(O(e_j) – O(e_k))$ est la fonction d'appartenance de $O(e_j) – O(e_k)$ à A. La valeur de μ est donnée par le rapport de la longueur de l'intervalle des valeurs de $O(e_j) – O(e_i)$ permettant de vérifier la contrainte (1), pour cette durée $O(e_j) – O(e_k)$ particulière, sur la longueur de l'intervalle des durées $O(e_j) – O(e_i)$ vérifiant la contrainte (3), qui est $(\delta_M - \delta_m)$.

Reprenons les trois cas de la figure 14 :

- Dans le cas a), $\mu(O(e_j) – O(e_k)) = \min(\frac{f_{j,i} - d_{j,i}}{\delta_M – \delta_m} , 1)$.

  La valeur 1 correspond au cas du polygone PO du type 1 ou 2.

- Dans le cas b), $\mu(O(e_j) – O(e_k)) = \frac{\varepsilon}{\delta_M – \delta_m}$, $\varepsilon \in [0, \min(\delta_M - \delta_m, f_{j,i} – d_{j,i})]$.

- Dans le cas c), $\mu(O(e_j) – O(e_k)) = 0$.

A partir de l'analyse précédente nous concluons que :

- si $O(e_j) – O(e_k) \in [\min(x_B,x_D), \max(x_B,x_D)]$,

$$\mu(O(e_j) – O(e_k)) = \min(\frac{f_{j,i} - d_{j,i}}{\delta_M – \delta_m} , 1).$$

- si $O(e_j) - O(e_k) \in [x_A, \min(x_B, x_D)]$,

$$\mu(O(e_j) - O(e_k)) = \frac{O(e_j) - O(e_k) - d_{j,i} + \delta_M}{\delta_M - \delta_m}.$$

- si $O(e_j) - O(e_k) \in [\max(x_B, x_D), x_C]$,

$$\mu(O(e_j) - O(e_k)) = \frac{O(e_k) - O(e_j) + f_{j,i} - \delta_m}{\delta_M - \delta_m}.$$

- si $O(e_j) - O(e_k) \notin [x_A, x_C]$, $\mu(O(e_j) - O(e_k)) = 0$, la contrainte (1) ne peut pas être vérifiée, figure 16.

La figure 16 donne la valeur de μ en fonction de $O(e_j) - O(e_k)$. Elle est relative au cas de la figure 12 précédemment étudiée.

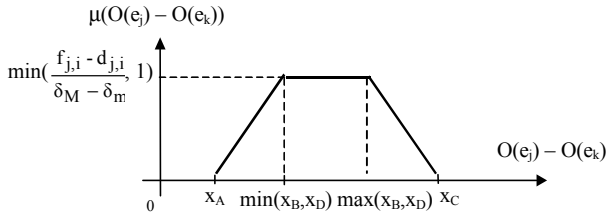


Figure 16. *La fonction d'appartenance à l'ensemble flou A, cas de la contrainte (1), $x_A \geq 0$, $x_B \geq 0$, $x_C \geq 0$, $x_D \geq 0$.*

III.4.1.2. Cas de contrainte de précédence

Une contrainte de précédence $O(e_i) < O(e_j)$ peut s'écrire comme une contrainte de type intervalle de la forme :

$0 < O(e_j) - O(e_i) < +\infty$. C'est le cas où $d_{j,i} = 0$ et $f_{j,i} = +\infty$.

La surveillance de la contrainte de précédence par S_1 consiste alors à chercher à partir des durées mesurables $O(e_j) - O(e_k)$, les durées $O(e_j) - O(e_i)$ qui vérifient à la fois :

$$\begin{cases} 0 \leq O(e_j) - O(e_i) < +\infty \\ O(e_j) - O(e_k) + \delta_m \leq O(e_j) - O(e_i) \leq O(e_j) - O(e_k) + \delta_M \end{cases}$$

Un raisonnement analogue à celui du § III.4.1.1. nous donne les résultats suivants dans le cas d'un système à deux superviseurs, sachant que le polygone *PO* est cette fois défini par $x_A = -\delta_M$, $x_B = -\delta_m$, $x_D = +\infty$, $x_C = +\infty$:

- si $O(e_j) - O(e_k) \in [-\delta_m, +\infty]$, $\mu(O(e_j) - O(e_k)) = 1$.
- si $O(e_j) - O(e_k) \in [-\delta_M, -\delta_m]$ alors $\mu(O(e_j) - O(e_k)) = \frac{O(e_j) - O(e_k) + \delta_M}{\delta_M - \delta_m}$.

- si $O(e_j) - O(e_k) < -\delta_M$ alors $\mu(O(e_j) - O(e_k)) = 0$, la contrainte de précédence ne peut pas être vérifiée, (figure 17).

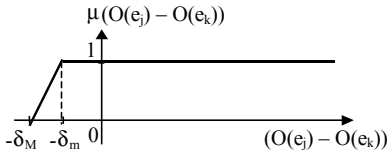


Figure 17. *La fonction d'appartenance à l'ensemble flou A ,cas de la contrainte de précédence.*

III.4.1.3. Cas de contrainte de type fenêtre d'admissibilité

Cette contrainte est exprimée par : $d_j \leq \min_{p=1..n,\ n\in N.}(O(e_j) - O(e_p)) \leq f_j : D_j$.

Elle peut être exprimée comme une conjonction de plusieurs contraintes de type intervalle comme suit:

$$\wedge_{p=1..n,\ n\in N.}\ d_{j,p} \leq O(e_j) - O(e_p) \leq f_{j,p}, \text{ avec } d_{j,p} = d_j + \pi_p \text{ et } f_{j,p} = f_j + \pi_p,\ \pi_p \in [0, +\infty[.$$

π_p représente la durée séparant l'occurrence d'un événement e_p de celle de l'événement e_j. Nous obtenons la contrainte liant e_j et e_p (p=1..n) suivante :

$$d_j + \pi_p \leq O(e_j) - O(e_p) \leq f_j + \pi_p \text{ notée } C_{jp}, \text{ avec } \pi_p \in [0, +\infty[.$$

Pour obtenir une expression simple de la contrainte C_{jp}, nous choisissons des valeurs particulières de π_p (0 et $+\infty$). Ces valeurs permettent d'avoir les bornes les plus larges possibles de la contrainte C_{jp} afin d'englober tous les cas fonction de π_p.
Nous concluons que $d_j \leq O(e_j) - O(e_p) < +\infty$, p=1..n.

Afin de distribuer cette contrainte, nous supposons qu'un seul événement e_p est reçu par le superviseur S_2 et que l'événement e_j est reçu par S_1. Les événements e_k, k=1..n, k≠p, sont reçus localement par S_1, qui est comme nous l'avons déjà dit, celui qui surveille cette contrainte.

Soit e_k l'événement envoyé par le superviseur 2 et reçu par le superviseur 1.
En identifiant $d_{j,i}$ à d_j et $f_{j,i}$ à $+\infty$, un raisonnement analogue à celui du § III.4.1.1. nous donne les résultats suivants dans le cas d'un système à deux superviseurs, sachant que le polygone PO est maintenant tel que $x_A = d_j - \delta_{Mp}$, $x_B = d_j - \delta_{mp}$, $x_D = +\infty$, $x_C = +\infty$:

- si $O(e_j) - O(e_k) \in [d_j - \delta_{mp}, +\infty]$, $\mu(O(e_j) - O(e_k)) = 1$.
- si $O(e_j) - O(e_k) \in [d_j - \delta_{Mp}, d_j - \delta_{mp}]$ alors

$$\mu(O(e_j) - O(e_k)) = \frac{O(e_j) - O(e_k) - d_j + \delta_{Mp}}{\delta_{Mp} - \delta_{mp}}$$

- si $O(e_j) - O(e_k) < d_j - \delta_{Mp}$ alors $\mu(O(e_j) - O(e_k)) = 0$, la contrainte C_{jp} ne peut pas être vérifiée, (figure 18).

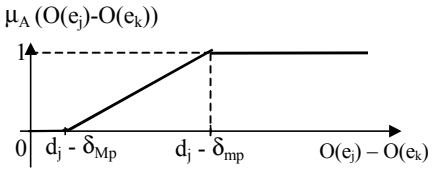


Figure 18. *La fonction d'appartenance à l'ensemble flou A, cas de la contrainte de type fenêtre d'admissibilité.*

Dans le cas général, s'il y a deux ou plusieurs événements e_p, p=1..n, p≠1, reçus par des superviseurs S_p, alors l'étude est sensiblement la même que dans le cas d'un seul événement e_p, puisque ces événements sont considérés un à un dans le calcul. S_1 reçoit les événements e_k, k=1..n, k≠1, des différents superviseurs S_p.

III.4.2. Prise en compte du délai opératoire dans la vérification des contraintes

Dans ce cas, il faut tenir compte du délai opératoire dans la vérification d'une contrainte globale. Ce type de délai représente, comme nous l'avons déjà dit, dans les chaînes logistiques ou les systèmes de transport une durée opératoire de longueur bornée. Le calcul à faire pour la vérification d'une contrainte de type intervalle C_{ji} avec prise en compte du délai opératoire est sensiblement le même que dans le cas de la prise en compte d'un délai de communication. Par la suite nous donnons les résultats de notre étude relatifs à ce cas.

Reprenons le cas a) de la figure 9 et connaissant la durée séparant les occurrences de e_j et de e_k, notre objectif est de surveiller la contrainte C_{ji} en tenant compte de la durée Δ introduite précédemment.

Nous pouvons alors écrire que :

$$O(e_j) – O(e_i) = (O(e_j) – O(e_k)) - \Delta = (O(e_j) – O(e_k)) + (O(e_k) – O(e_i)),$$

Soit encore puisque $\delta_m \leq \Delta \leq \delta_M$ (2)

$$O(e_j) – O(e_k) - \delta_M \leq O(e_j) – O(e_i) \leq O(e_j) – O(e_k) - \delta_m$$

et $O(e_j) – O(e_k) \in]-\infty, +\infty[$.

Un même raisonnement que celui présenté au § III.4.1.1. nous permet de conclure, pour $x_A = d_{j,i} + \delta_m$, $x_B = d_{j,i} + \delta_M$, $x_D = f_{j,i} + \delta_m$, $x_C = f_{j,i} + \delta_M$, que :

- si $O(e_j) – O(e_k) \in [\min(x_B,x_D), \max(x_B,x_D)]$,

$$\mu(O(e_j) – O(e_k)) = \min\left(\frac{f_{j,i} - d_{j,i}}{\delta_M - \delta_m}, 1\right).$$

- si $O(e_j) – O(e_k) \in [x_A, \min(x_B,x_D)]$,

$$\mu(O(e_j) – O(e_k)) = \frac{O(e_j) – O(e_k) – d_{j,i} – \delta_m}{\delta_M - \delta_m}.$$

- si $O(e_j) - O(e_k) \in [\max(x_B, x_D), x_C]$,

$$\mu_A(O(e_j) - O(e_k)) = \frac{O(e_k) - O(e_j) + f_{j,i} + \delta_M}{\delta_M - \delta_m}.$$

- si $O(e_j) - O(e_k) \notin [x_A, x_C]$, $\mu(O(e_j) - O(e_k)) = 0$, la contrainte (1) ne peut pas être vérifiée.

Pour le cas b) de la figure 9, le calcul est le même que le § III.4.1.1.

Après avoir étudié la vérification d'une contrainte temporelle en prenant en compte un délai opératoire, nous nous intéressons maintenant au cas général.

III.4.3. Prise en compte des délais, cas général

Il s'agit de surveiller une contrainte comme étant une relation impliquant une séquence de durées liant des événements, par opposition au cas simple où une contrainte impliquait une seule durée liant des événements. Nous considérons que la durée globale surveillée séparant les occurrences de e_i et de e_j est la somme des durées constituant la séquence.
Dans le cas d'un délai de communication, ce délai n'est constitué que d'une seule durée qui sépare les occurrences de e_i et de e_k, (figure 19).

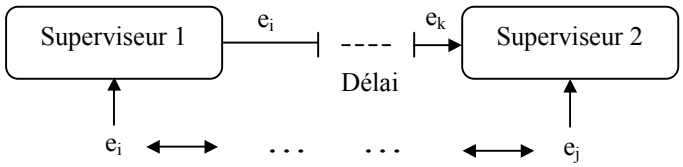


Figure 19. *Le délai de communication entre superviseurs, cas général.*

Dans le cas d'un délai opératoire, celui-ci peut être constitué d'une séquence de plusieurs durées opératoires séparant l'occurrence de e_i de celle de l'événement intermédiaire e_k, (figure 20). L'événement e_j est daté par un superviseur 2, l'événement e_k est daté par un superviseur 1 et l'événement e_i n'est reçu par aucun des superviseurs.

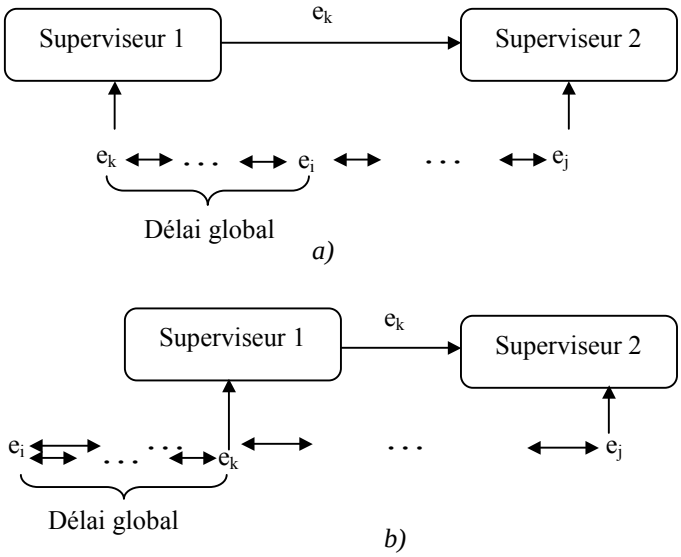


Figure 20. *Le délai opératoire, cas général.*

Les mécanismes de vérification de la contrainte liant les occurrences des événements e_j et e_i sont les mêmes que ceux détaillés au § III.3.4.2. En effet, la durée globale surveillée séparant les occurrences de e_i et de e_j est la somme des durées constituant la séquence et le délai global est la somme des durées connues de la séquence séparant les occurrences des événements e_i et e_k.

Nous présentons maintenant quelques exemples d'application afin d'illustrer notre approche de la prise en compte des délais dans la vérification de contraintes temporelles.

III.4.4. Exemples d'application

Nous considérons tout d'abord un problème de délai de communication puis des problèmes de délais opératoires.

III.4.4.1. Délai de communication entre superviseurs

Supposons un système composé de *n* processeurs qui exécutent des instructions d'un même programme, (figure 21). Le superviseur *n* associé au processeur n surveille l'exécution des différentes instructions. Le superviseur 1 associé au processeur 1 reçoit e_1, l'événement début de la première instruction, et le superviseur n reçoit e_n, l'événement fin de la dernière instruction. L'exécution des instructions par les processeurs se fait en parallèle ou en séquence en fonction de la dépendance qu'a l'exécution d'une instruction du résultat de l'exécution d'une autre instruction. Ainsi les différents processeurs se synchronisent afin d'assurer une exécution distribuée des instructions, c'est à dire sur différents processeurs.

Sachant une durée totale, comprise entre 10ms et 20ms pour l'exécution de toutes les instructions et un délai de communication entre le superviseur 1 et le superviseur n compris entre 5ms et 8ms. Nous allons surveiller le déroulement de l'exécution des différentes instructions par l'intermédiaire du superviseur *n*. Pour cela, le superviseur *n* reçoit l'image de l'événement e_1 qui est e_k.

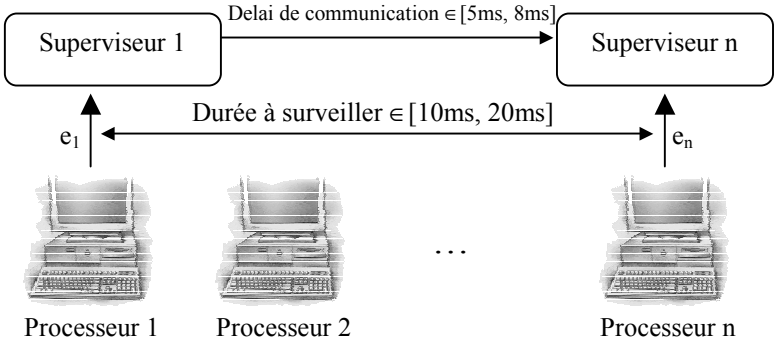


Figure 21. *Exemple de prise en compte du délai de communication.*

La formulation du problème donne la contrainte à vérifier suivante :

$$10ms \leq O(e_n) - O(e_1) \leq 20ms$$

sachant un délai de communication $\Delta = O(e_k) - O(e_1) \in [5ms, 8ms]$.

A partir des résultats obtenus dans le § III.4.1.1., nous obtenons la figure suivante :

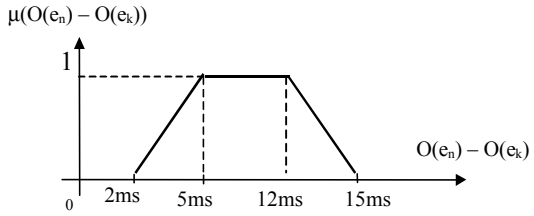


Figure 22. *La fonction d'appartenance relative à la contrainte à vérifier.*

D'après la figure 22, si la durée séparant les occurrences de e_n et de e_k est dans l'intervalle [5ms, 12ms], l'exécution des instructions s'est déroulée normalement. Si cette durée est en dehors de cet intervalle, la possibilité de vérifier la contrainte représentant la durée totale d'exécution augmente chaque fois que la durée entre les occurrences de e_n et de e_k s'approche de l'intervalle [5ms, 12ms]. En effet, si cette possibilité est inférieure à 1, l'exécution des instructions peut avoir pris du retard ou les mécanismes de synchronisation peuvent être défaillants. Nous détaillerons plus tard quand est ce que ceci amène à la détection d'un symptôme de défaillance.

III.4.4.2. Délai opératoire ou de transport

Nous nous intéressons à la surveillance de l'exécution d'un plan de d'usinage de pièces dans un atelier. Les pièces, acheminées par tapis roulants, subissent des opérations dont une opération de fraisage. Cette opération est effectuée par une « fraiseuse » , (figure 23).

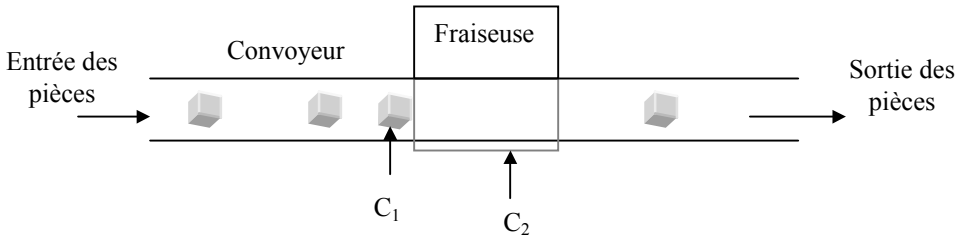


Figure 23. *Exemple de prise en compte du délai opératoire.*

Nous disposons de capteurs C_1 et C_2. Lorsque le capteur C_1 est activé, un événement e_k : « pièce arrivée sur la fraiseuse » est généré et sera reçu par un superviseur 1. Lorsque le capteur C_2 est activé, un événement e_j : « opération de fraisage terminée » est généré et sera reçu par un superviseur 2. Le problème que nous nous posons est de surveiller la durée séparant l'entrée d'une pièce sur le tapis et la fin de l'opération de fraisage qui lui est associée. Signalons que l'événement e_i : « entrée sur tapis » ne peux pas être reçu par les superviseurs puisqu'il n'existe pas de capteur qui lui est dédié.
Supposons avoir la contrainte temporelle à surveiller suivante :

$$3 \leq O(e_j) - O(e_i) \leq 10 \quad : C_{ji}$$

Les hypothèses sur lesquelles se base la surveillance de cette contrainte sont :

- la durée séparant l'entrée d'une pièce sur le tapis et le début de l'opération d'usinage est bornée. Cette durée est le délai opératoire $\Delta = O(e_k) - O(e_1)$;

- la tolérance associée à la contrainte à surveiller est du même ordre que l'incertitude sur le délai opératoire.

Comme nous l'avons dit au § III.3.3., c'est le superviseur 2 qui vérifie la contrainte C_{ji} à partir de la durée opératoire connue $\Delta = O(e_k)-O(e_i)$, $\Delta \in [1, 3]$, et de la réception de l'événement e_j du capteur C_2 et de l'événement e_k renvoyé par le superviseur 1. En datant les occurrences des événements e_k et e_i, le superviseur 2 peut ainsi calculer la possibilité de vérification de la contrainte n utilisant le calcul développé au § III.4.1.1. Nous obtenons les résultats suivants :

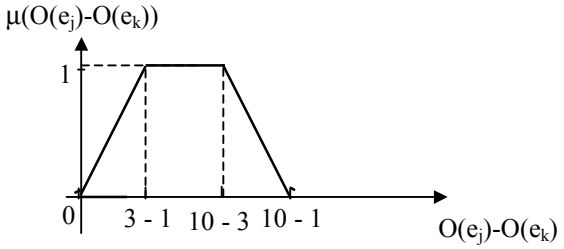


Figure 24. *La fonction d'appartenance relative à la contrainte C_{ji}.*

Prenons, par exemple, le cas de fonctionnement où $O(e_j) - O(e_k) = 1$. Le superviseur 2 déduit que la possibilité pour que l'opération de fraisage se soit bien déroulée est de 0.5, (figure 24). Ceci peut être dû à ce que l'opération de fraisage se soit déroulée plus rapidement que prévu et se soit donc mal exécutée.
Si $O(e_j) - O(e_k) = 8$, le superviseur déduit, avec une possibilité de 0.5, que l'opération de fraisage a pris plus de temps que prévu. Ce retard pourrait influencer l'exécution d'opérations futures pour la pièce considérée et perturber ainsi le plan d'exécution qui lui est associé. Dans les deux cas, le superviseur 2 détecte qu'un mauvais fonctionnement a pu avoir lieu lors de l'opération de fraisage.

A ce stade, il est nécessaire de clarifier la notion de flou qu'associent les mécanismes proposés à la vérification d'une contrainte globale.

III.4.5. Interprétation de la notion de flou

Si l'incertitude sur le délai n'existait pas, la possibilité de vérifier une contrainte serait de 1 ou de 0. Quand le l'incertitude sur le délai est prise en compte, la possibilité de vérifier une contrainte appartient à [0, 1]. La vérification d'une contrainte devient donc *floue* (Cardoso et al 1991, 1996).

Prenons le cas d'une contrainte de type intervalle $d_{j,i} \leq O(e_j) - O(e_i) \leq f_{j,i}$ notée C_{ji} avec $d_{j,i}$ et $f_{j,i} \in Q^+$. Supposons que cette contrainte soit vérifiée avec une possibilité de 0.75. Ceci veut dire qu'après l'occurrence de e_i, la possibilité pour que l'occurrence de e_j soit réellement dans l'intervalle $[d_{j,i}, f_{j,i}]$ est de 0.75, (figure 25).

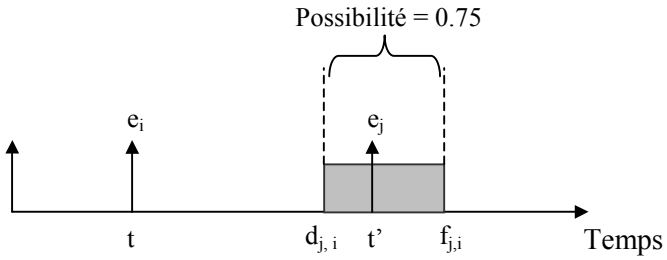


Figure 25. *Possibilité d'occurrence.*

A ce stade de la surveillance, le superviseur concerné par la surveillance de la contrainte ne possède que la connaissance de cette valeur floue de possibilité. Réellement, l'occurrence de e_j peut avoir eu lieu dans l'intervalle $[d_{j,i}, f_{j,i}]$ ou en dehors de cet intervalle. L'interprétation de cette notion de flou est la suivante :

- l'occurrence de e_j a eu lieu dans l'intervalle $[d_{j,i}, f_{j,i}]$. A cause de l'incertitude sur le délai, le superviseur ne peut être sûr de la vérification de la contrainte ;
- l'occurrence de e_j a eu lieu en dehors de l'intervalle $[d_{j,i}, f_{j,i}]$. Dans ce cas, l'occurrence de e_j s'est produite plus tôt que $d_{j,i}$ ou plus tard que $f_{j,i}$ relativement à l'occurrence de e_i, mais à cause de l'incertitude sur le délai de communication, le superviseur ne peut être sûr de la non vérification de la contrainte.

Trois approches peuvent être utilisées permettant d'avoir une certitude sur la vérification de la contrainte :

- La première consiste à fixer un seuil de vérification θ d'une contrainte globale. Si la possibilité de vérifier la contrainte est supérieure à ce seuil alors la contrainte est considérée vérifiée par le superviseur, sinon elle est considérée non vérifiée. Ce type d'approche peut être utilisé dans des systèmes où le fonctionnement autorise un risque quant au déroulement des opérations. Dans les systèmes manufacturiers existent des paramètres variables et non contrôlables comme la charge de travail d'une machine donnée. En effet, les délais d'attente des pièces au niveau d'une machine peuvent varier en fonction du nombre de pièces en attente de la même opération. Ceci peut influer la vérification de contraintes temporelles. Comme cette situation est généralement transitoire, il est très probable que la contrainte globale soit vérifiée pour une prochaine configuration. Le problème qui reste posé est la manière de fixer le seuil θ pour toute contrainte globale. Ceci peut être résolu en se basant sur le fonctionnement passé du système surveillé. Ainsi un opérateur peut, dans un premier temps, établir pour chaque possibilité de vérification de la contrainte si celle-ci a été réellement vérifiée ou pas en observant le fonctionnement du système. Le seuil θ peut être exprimé comme étant la plus petite valeur de possibilité à partir de laquelle la contrainte est réellement vérifiée ;

- La deuxième approche concerne les systèmes dans lesquels les opérations à surveiller présentent un danger ou un risque vital. Dans ce cas, il est nécessaire d'avoir une information certaine sur la vérification ou non d'une contrainte associée. La détermination d'un seuil de vérification n'est pas envisageable dans ce cas. Une solution consisterait à charger un opérateur d'aller vérifier l'état du procédé dont les événements sont impliqués dans la contrainte vérifiée avec une possibilité inférieure à 1 ;

- La troisième approche, moins coûteuse que les deux précédentes, consiste à faire coopérer des superviseurs particuliers permettant ainsi d'affiner la vérification de la contrainte globale et d'acquérir dans certains cas une certitude quant à la vérification de la contrainte. Une combinaison de ces trois approches peut aussi être utilisée.

Nous développons l'approche de coopération dans le paragraphe suivant.

III.4.6. Coopération entre superviseurs

La coopération est une étape qui suit immédiatement la vérification avec une possibilité appartenant à]0, 1[d'une contrainte globale. Elle permet à d'autres superviseurs de re-vérifier la même contrainte globale mais sachant un délai diffèrent. Comme la re-vérification de la contrainte nécessite un certain temps additionnel, le superviseur susceptible d'initier les mécanismes de coopération peut ou non lancer une coopération. En effet, une détection tardive de la violation de la contrainte temporelle peut impliquer la propagation du symptôme de défaillance et impliquer des conséquences catastrophiques sur le fonctionnement du système, alors qu'une détection immédiate de la violation peut amener à une fausse alarme. La multiplication des fausses alarmes peut créer chez l'opérateur un sentiment de non confiance vis à vis du système de surveillance causant ainsi l'ignorance d'alarmes « sérieuses » ou réelles par l'opérateur. Suivant l'une ou l'autre des conséquences engendrées par une coopération, un superviseur peut lancer ou pas une coopération.

Un affinement par coopération de la vérification d'une contrainte globale peut avoir lieu lorsqu'un même événement est commun à au moins une contrainte globale autre que celle vérifiée avec une possibilité $\in$]0, 1[. Nous étudions et sans perte de généralité, le cas de deux contraintes globales ayant un événement en commun comme le montre la figure 26.

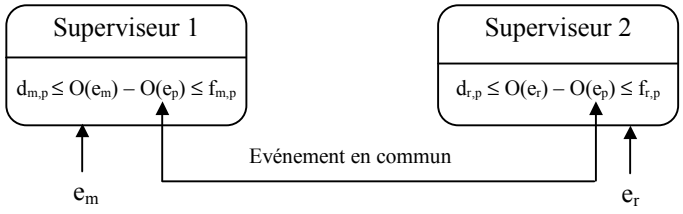


Figure 26. *Evénement en commun.*

Le cas de la figure 26 est le seul cas possible dans lequel un affinement de la vérification d'une des contraintes peut être fait. En effet, tous les autres cas de figure peuvent être résolus de manière locale par un superviseur.
L'événement apparaissant dans le « membre droit » d'au moins deux contraintes globales est appelé *événement en commun*. Par exemple dans la figure 26, l'événement e_p est l'événement en commun. Des mécanismes additionnels doivent être développés afin d'exploiter cette donnée.

Dans la figure précédente, les superviseurs 1 et 2 surveillent deux contraintes globales incluant chacune le même événement e_p. Si une de ces deux contraintes est vérifiée avec une possibilité $\in$]0, 1[par un des superviseurs, celui-ci peut amorcer un processus de coopération en envoyant l'événement non en commun (« membre gauche de la contrainte ») à l'autre superviseur. Le superviseur destinataire re-vérifie la contrainte qui a été vérifiée avec une possibilité $\in$]0, 1[. Dans le meilleur cas, la contrainte sera vérifiée avec une possibilité égale à 1 ou à 0. Ceci permettra au superviseur expéditeur d'avoir une certitude quant à la vérification ou non de la contrainte considérée.

Nous explicitons, par la suite, ces différents mécanismes de coopération dans le cas d'un délai opératoire et d'un délai de communication. Le cas d'une contrainte de type intervalle sera le seul cas détaillé dans la mesure où les deux autres contraintes (fenêtre d'admissibilité et précédence) peuvent être ramenées à une contrainte de ce type, §III.4.1.

III.4.6.1. Cas d'un délai opératoire

Supposons que pour vérifier une contrainte de type intervalle C_{mp} où l'événement e_p n'est reçu par aucun superviseur, le superviseur 1 doive recevoir l'occurrence d'un événement intermédiaire e_{k1} d'un superviseur n, n≠1 et n≠2, avec un délai opératoire connu appartenant à [a, b], a, b∈ Q^+. De même, pour vérifier la contrainte C_{rp}, le superviseur 2 doit recevoir l'occurrence d'un événement intermédiaire e_{k2} d'un superviseur i, i≠1 et i≠2 avec un délai opératoire connu appartenant à [c, d], c, d∈ Q^+, figure 27.

Les treillis temporels correspondant à la situation décrite sont les suivants :

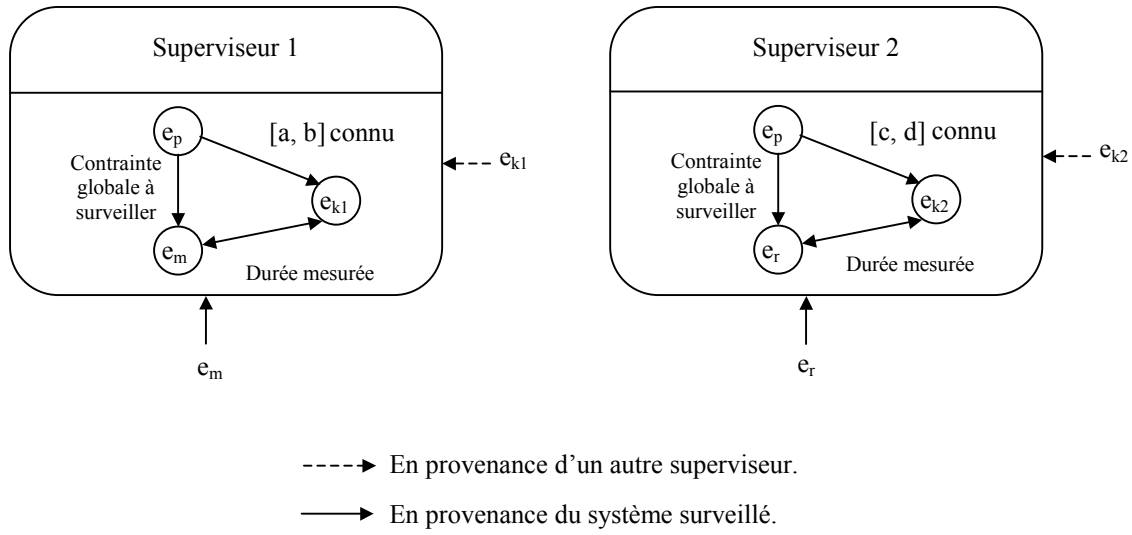


Figure 27. *Treillis temporels.*

Supposons aussi que la contrainte C_{mp} soit vérifiée avec une possibilité appartenant à]0, 1[.

Le superviseur 1 doit donc envoyer l'occurrence de l'événement e_m au superviseur 2. Nous obtenons un nouveau treillis au niveau du superviseur 2 qui est le suivant :

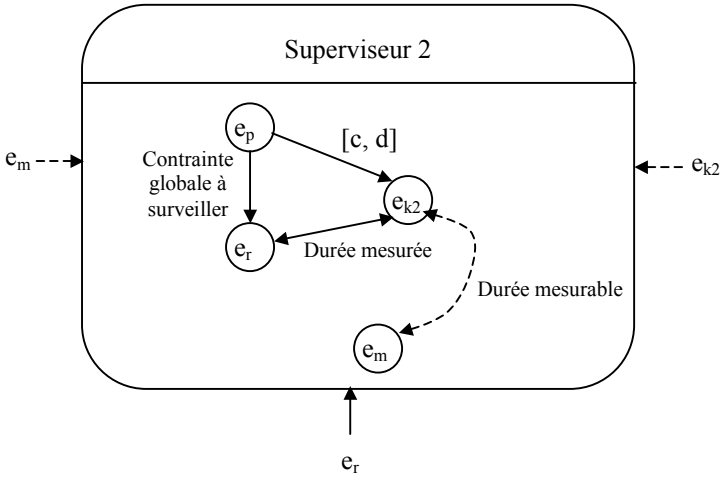


Figure 28. *Nouveau treillis au niveau du superviseur 2.*

A niveau du superviseur 2, nous pouvons maintenant écrire :

$$O(e_m)\text{-}O(e_p) = (O(e_m) - O(e_{k2})) + (O(e_{k2}) - O(e_p)) \quad (4).$$

Afin de vérifier la contrainte C_{mp} par le superviseur 2 sachant (4), celui-ci se base sur les résultats du § III.4.1.1. Le délai est dans ce cas $O(e_{k2})\text{-}O(e_p) \in [c, d]$, alors que la durée à mesurer pour vérifier la contrainte C_{mp} est $O(e_m) - O(e_{k2})$. Si la contrainte C_{mp} est vérifiée, le résultat de la possibilité de sa vérification est envoyé au superviseur 1. Trois cas se présentent :

1) le résultat de la possibilité envoyé est de 1, le superviseur 1 conclut qu'aucun symptôme de défaillance n'est à détecter ;
2) le résultat de la possibilité envoyé est de 0, la sous-chronique est non reconnue et le superviseur 1 détecte un symptôme de défaillance ;
3) le résultat de la possibilité envoyé appartient à]0, 1[, alors cette information servira à calculer une possibilité moyenne de vérification de la contrainte considérée. La valeur de la possibilité moyenne représente la nouvelle valeur de possibilité de vérification de la contrainte. Dans ce cas, le système de surveillance distribué peut faire appel à la première et à la deuxième approche présentées dans le § III.4.5.

Afin de pouvoir initier les mécanismes de coopération, un superviseur associe localement à chaque événement daté, un ensemble de superviseurs à qui envoyer un message de coopération. La liste des couples (événement, ensemble de superviseurs) est appelée *liste de coopération notée* L_c et a la même forme que celle utilisée pour la communication entre superviseurs que nous avons notée L_r. Une liste L_c est donc associée à chaque superviseur S_i.

La forme standard d'un message échangé entre superviseurs est un triplet (e, e', ω) où e est un événement daté, e' est un événement en commun, e et e' sont liés par une contrainte et ω est une valeur de possibilité. Le message est de la forme (e, ∅, ∅) lorsqu'il s'agit d'un message qui informe d'une occurrence d'événement. Il est de la forme (e, e', ∅) lorsqu'il s'agit d'un message de

sollicitation de coopération et a la forme (e, e', P), avec P est une valeur de possibilité, lorsqu'il s'agit d'un message de résultat de coopération.

Soit e un événement reçu par S_i et e' un événement en commun lié à e par une contrainte $C \in \zeta_i$ avec ζ_i l'ensemble des contraintes associées au superviseur S_i. Nous présentons, par la suite, les mécanismes de coopération entre superviseurs.

Mécanismes de coopération :

> *Si (C est vérifiée par S_i avec une possibilité$\in$]0, 1[) alors envoyer (e, e', ∅) à tout $S_j \in L$, j=1..n, avec (e, L)$\in L_c$,* avec *L* un sous-ensemble de superviseurs et L_c la liste de coopération associée au superviseur S_i.
>
> *Si (e, e', ∅) est reçu par S_j alors re-vérifier au niveau de S_j la contrainte C.*
>
> *Si la contrainte C est vérifiée alors envoyer (e, e', P(C)) à S_i,* P(C) la nouvelle possibilité de vérification de la contrainte C et que nous noterons plus tard par P_t.

Le superviseur S_i reste en attente des messages de résultats de possibilité des différents superviseurs S_j jusqu'à ce qu'un message contenant un résultat de possibilité égale à 1 ou 0, ou que tous les messages de résultats de coopération attendus lui arrivent. Si tous les messages attendus ont été reçus par S_i et qu'ils contiennent des valeurs de possibilité P_t appartenant à]0, 1[, alors ce superviseur peut calculer la moyenne Ψ, choisir le maximum de ces valeurs, ou en choisir le minimum comme étant la possibilité de vérification de la contrainte. Le choix du maximum (respectivement du minimum) crée un erreur maximale égale à la différence entre la valeur maximale et la valeur minimale de la possibilité. Le choix de la moyenne permet de réduire cette erreur à la moitié de cette différence. Nous choisissons donc la moyenne des possibilités comme étant la possibilité de vérification de la contrainte. Nous obtenons :

$$\Psi = \frac{\sum_{t=1..n} P_t}{n},$$

avec *n* est le nombre de superviseurs ayant coopéré avec S_i et P_t la possibilité associée à la vérification de la contrainte C par un superviseur S_t.
Ce choix n'étant pas général et dépend de la nature du système surveillé comme nous l'avons détaillé précédemment. Ce processus peut se répéter pour chaque contrainte globale. Rappelons que si une contrainte ne peut pas être vérifiée alors la sous-chronique correspondante est non reconnue. Quand toutes les contraintes d'une sous-chronique sont vérifiées avec une possibilité strictement positive, la sous-chronique est vérifiée avec une possibilité qu'est le minimum de toutes les possibilités de vérification des contraintes la composant.

Exemple d'application :

Soient les contraintes à vérifier suivantes :

$$3 \leq O(e_2) - O(e_1) \leq 5 \quad : C_{2\,1}$$
$$3 \leq O(e_4) - O(e_1) \leq 7 \quad : C_{4\,1}$$

La décomposition du système en zones donne lieu à quatre superviseurs S_1, S_2, S_3 et S_4. Le superviseur S_1 recevant en provenance du procédé seulement l'événement e_2 vérifie la contrainte C_{21} sachant un délai représenté par la durée connue $O(e_3) - O(e_1)$ et le superviseur S_2 recevant en provenance du procédé seulement l'événement e_4 vérifie la contrainte C_{41} sachant un délai représenté par la durée connue $O(e_5) - O(e_1)$. Le superviseur S_3 reçoit en provenance du procédé l'événement e_3 et le superviseur S_4 reçoit en provenance du procédé l'événement e_4. L'événement e_1 n'est reçu par aucun des superviseurs.
Nous représentons ci-dessous les informations dont dispose chacun des superviseurs pour vérifier les contraintes, (figure 29).

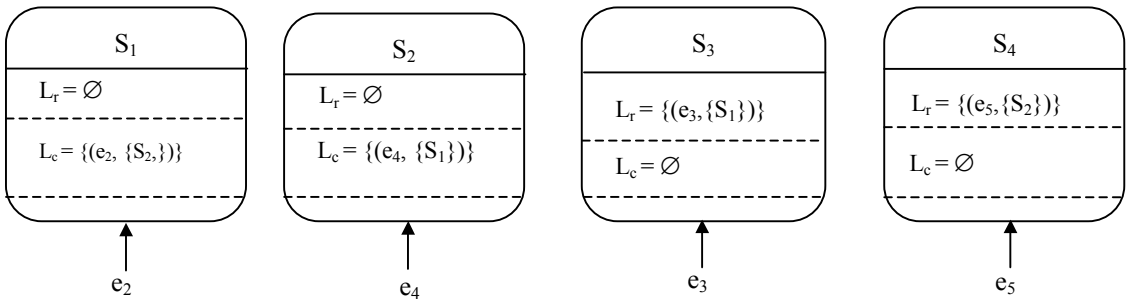


Figure 29. *Représentation des informations relatives aux superviseurs.*

Dès qu'un superviseur reçoit un événement en provenance du système surveillé, il l'envoie aussitôt, en consultant la liste de renvoi L_r, aux superviseurs concernés afin qu'ils puissent vérifier leurs propres contraintes, figure 30.

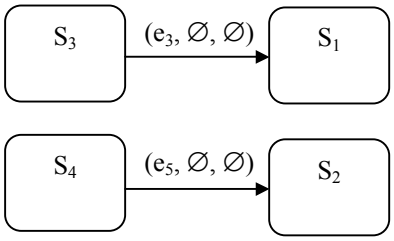


Figure 30. *Communication des événements à partir de la liste de renvoi.*

Lorsqu'un superviseur S_i, i=1,2, vérifie une contrainte avec une possibilité$\in$]0, 1[, il commence un processus de coopération en consultant la liste L_c. Par exemple, si la contrainte C_{21} est vérifiée avec une possibilité de 0.4, le superviseur S_1 peut coopérer avec le superviseur S_2 vérifiant C_{41} puisqu'il y a un événement en commun qui est e_1, en envoyant l'occurrence de e_2, (figure 31).

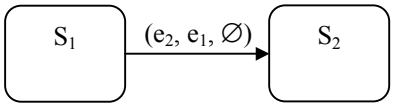


Figure 31. *Envoi d'un message de coopération.*

En mesurant la durée entre l'occurrence de e_2 et celle de e_5, le superviseur S_2 associe une nouvelle possibilité à la vérification de la contrainte C_{21}. Supposons que cette possibilité vaut 0. Le superviseur S_2 envoie un message de résultat de coopération à S_1, (figure 32).

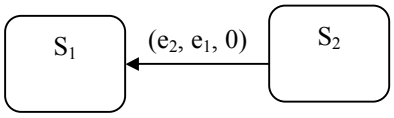


Figure 32. *Envoi d'un message de résultat de coopération.*

Le superviseur S_1 détecte alors une violation de la contrainte C_{21}.

III.4.6.2. Cas d'un délai de communication

Prenons le cas d'étude de la figure 27. Pour vérifier la contrainte C_{mp}, le superviseur 1 doit recevoir l'occurrence de l'événement e_p d'un superviseur n, $n \neq 1$ et $n \neq 2$, avec un délai de communication connu appartenant à [a, b], a, $b \in Q^+$. L'événement image de e_p reçu par le superviseur 1 est noté e_{k1}. De même, pour vérifier la contrainte C_{rp}, le superviseur 2 doit recevoir l'occurrence de l'événement e_p du même superviseur *n* avec un délai de communication connu appartenant à [c, d], c, $d \in Q^+$. L'événement image de e_p reçu par le superviseur 2 est noté e_{k2}.

Supposons que la contrainte C_{mp} soit vérifiée avec une possibilité appartenant à]0, 1[. Le superviseur 1 doit donc envoyer l'occurrence de l'événement e_m au superviseur 2 avec un délai de communication connu appartenant à [e, f], e, $f \in Q^+$. L'événement image de e_m reçu par le superviseur 2 est noté e_{k3}, figure 33. Nous obtenons un nouveau treillis au niveau du superviseur 2 qui est le suivant :

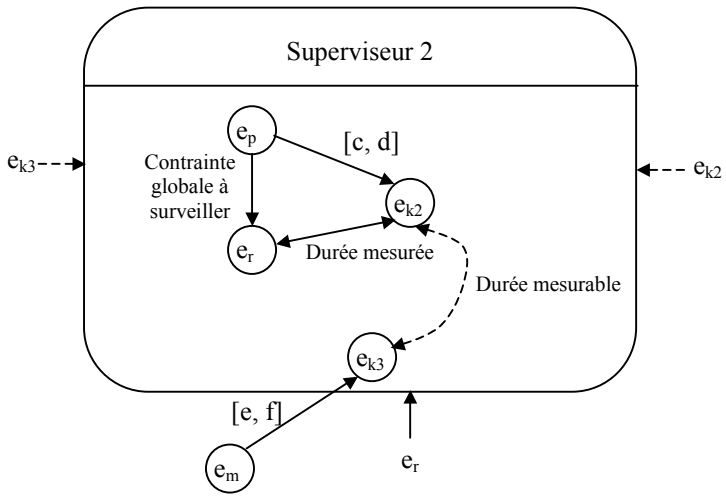


Figure 33. *Nouveau treillis au niveau du superviseur 2.*

A niveau du superviseur 2, nous pouvons maintenant écrire :

$$O(e_m)\text{-}O(e_p) = (O(e_m) - O(e_{k3})) + (O(e_{k3})\text{-}O(e_{k2})) + (O(e_{k2})\text{-}O(e_p)) \quad (5).$$

Afin de vérifier la contrainte C_{mp} par le superviseur 2 sachant (5), celui-ci se base sur les résultats du § III.4.1.1. Le délai appartient, dans ce cas, à [c-f, d-e] alors que la durée à mesurer pour vérifier la contrainte C_{mp} est $O(e_{k3}) - O(e_{k2})$. Si la contrainte C_{mp} est vérifiée, le résultat de la possibilité de sa vérification est envoyé au superviseur 1.

En supposant qu'un superviseur connaisse les bornes du délai de communication entre lui et les autres superviseurs, les mécanismes de coopération présentés au § III.4.6.1 restent valables dans ce cas.

Après avoir étudié la vérification distribuée de contraintes temporelles avec prise en compte des délais, nous nous intéressons par la suite à la prise en compte d'une date d'occurrence imprécise d'un événement lors de l'intégration de celui-ci dans une chronique ou sous-chronique. Ce problème se rapproche de la problématique traitée dans ce chapitre dans la mesure où l'incertitude associée au délai est assimilée à l'incertitude associée à la date d'occurrence d'un événement faisant partie de la contrainte et où la possibilité vérification d'une contrainte globale correspond à la possibilité de l'intégration d'un événement dans le processus de reconnaissance en cours d'une chronique ou d'une sous-chronique.

III.5. Date d'occurrence imprécise

(Dousson et al 1994) a introduit la prise en compte d'une date d'occurrence imprécise d'un événement lors de son intégration dans une chronique. C'est le cas où le système ne peut pas dater avec précision les dates d'occurrence des événements. Il est supposé que les dates possibles d'occurrence d'un événement fournies par une horloge locale à un superviseur sont équi-distribuées dans un intervalle *d(e)*. Afin de résoudre ce problème, la possibilité $\pi(e)$ d'intégrer un événement dans une chronique ou sous-chronique a été définie dans (Dousson et al 1994) par :

$$\pi(e) = \frac{\lambda(d(e) \cap D(e))}{\lambda(d(e))}$$

avec λ donnant la longueur d'un intervalle et D(e) l'ensemble des dates d'occurrence de *e* compatibles avec les contraintes. Cette expression prend en compte les résultats fournis par la figure 14 ainsi que des calculs faits au § III.4.1.1. La possibilité pour que la chronique ou sous-chronique reconnue se soient réellement reconnues, correspond au minimum des possibilités π(e) des événements. Des extensions possibles peuvent être envisagées précisant des seuils minimum de possibilité pour chacun des événements à intégrer.

III.6. Conclusion

Il existe dans les systèmes de surveillance distribués des délais qui représentent des délais de communication ou des délais liés à des durées opératoires. La vérification de contraintes temporelles liant des événements qui ne sont pas datés par le même superviseur, doit tenir compte de ces délais. Nous avons développé dans ce chapitre, pour chaque type de délai, les mécanismes permettant de vérifier de telles contraintes. Le résultat obtenu est une possibilité de vérification de la contrainte. Comme la possibilité de vérification d'une contrainte est une valeur floue, une approche permettant l'affinement de cette vérification est définie comme étant une coopération pouvant, sous certaines conditions, avoir lieu entre les différents superviseurs. La procédure de coopération consiste à re-vérifier la contrainte concernée par des superviseurs différents de celui l'initiant. Lorsque une contrainte est vérifiée avec une possibilité nulle, une violation de la contrainte s'est produite et un symptôme de défaillance est détecté. Dans le cas où cette possibilité appartient à]0, 1[, l'utilisation d'un seuil de vérification de la contrainte peut être une solution permettant de trancher quant à la vérification ou non de la contrainte. Une sous-chronique est reconnue avec une possibilité qu'est le minimum de toutes les possibilités de vérification des contraintes la composant.

Les mécanismes de la détection distribuée de violations de contraintes temporelles qui sont développés doivent être modélisés par un outil adéquat permettant la représentation des aspects statiques d'expression des contraintes à vérifier et dynamiques de vérification de celles-ci. Ces modèles doivent aussi pouvoir exprimer les différentes interactions entre les sous-systèmes de surveillance. Le chapitre suivant est dédié aux aspects modélisation.

Chapitre IV

Modélisation par réseaux de Petri des mécanismes de la fonction détection distribuée

IV.1. Introduction

Les réseaux de Petri sont reconnus comme étant un outil bien adapté à la modélisation des systèmes asynchrones grâce à leur représentation graphique simple et l'outil mathématique d'analyse qu'ils offrent. Ils sont appropriés à la représentation de contraintes temporelles dont la conjonction forme une chronique. Pour la détection distribuée, un réseau de Petri représente une sous-chronique. Le problème qui se pose est celui de la modélisation de ces réseaux représentant les sous-chroniques et les mécanismes nécessaires à leurs reconnaissance, ainsi que les interactions possibles entre eux.
Nous présentons brièvement dans la première section de ce chapitre les notions de base des réseaux de Petri. Dans la deuxième section, les principes de modélisation d'une chronique sont décrits. Les mécanismes de la détection distribuée présentés au chapitre III sont modélisés dans la troisième section. Nous verrons que les réseaux de Petri ont l'avantage de permettre de représenter de façon naturelle un sous-ensemble de sous-chroniques communicantes et coopérantes. Pour cela, nous proposons un modèle de sous-chroniques composé d'un modèle de contraintes locales et d'un modèle de contraintes globales.

IV.2. Rappels sur les réseaux de Petri

Nous présentons ici quelques définitions de base susceptibles d'unifier les notations dans la suite de ce chapitre.

IV.2.1. Réseau de Petri et concepts de base

Définition 1 : Un réseau de Petri (Peterson 1977), est un quadruplet R=< P, T, Pre, Post > où :

- P est ensemble fini de places,
- T est ensemble fini de transitions,

- Pre : P×T → N est l'application incidence *avant*, reliant des transitions à des places les précédant,
- Post : P×T → N est l'application incidence *arrière*, reliant des transitions à des places les suivant.

Définition 2 : Soir R un Réseau de Petri (RdP). Un marquage du réseau de Petri R est une fonction *M* de l'ensemble des places *P* dans *N*. L'ensemble des marquages d'un réseau de Petri R est donné par M(R). Un réseau marqué est le couple <R, M> avec M_0 le marquage *initial.*

Définition 3 : Soit *R* un réseau de Petri, *M* est un marquage, et *t* une transition. La transition t est sensibilisée (franchissable) pour le marquage M, noté M ≥ Pre(., t) ou M(t> ou ${}^{t}M \rightarrow$, si et seulement si :

$$\forall p \in P, M(p) \geq Pre(p, t).$$

Définition 4 : Soit *R* un réseau de Petri, *M* est un marquage, et *t* une transition sensibilisée pour M. Le tir (franchissement) de t donne la marquage M' tel que :

$$\forall p \in P, M'(p) = M(p) - Pre(p, t) + Post(p, t).$$

Définition 5 : Soit *R* un réseau de Petri, *M* est un marquage, et t_1, t_2 deux transitions. Les transitions t_1 et t_2 sont en conflit structurel, si et seulement si, il existe une place p∈ P d'entrée telle que :

$$Pre(p, t_1) . Pre(p, t_2) \neq 0, \text{ où } . \text{ dénote la multiplication d'entier naturels.}$$

Les transitions t_1 et t_2 sont en conflit effectif pour un marquage M, si elles sont en conflit structurel et sont sensibilisées par M.

IV.2.2. Réseau de Petri temporel

Les réseaux de Petri temporels ou t-temporels ont été introduits par (Merlin 1974). Ils se basent sur le principe de l'association d'une durée de sensibilisation bornée $\theta_s(t)$ à chaque transition *t*. Ceci permet de laisser disponibles les jetons dans la place d'entrée de la transition *t* jusqu'à son tir. D'autres transitions en conflit avec *t* peuvent ainsi consommer ces jetons lors de leur franchissement. Dans ces réseaux, un intervalle de temps [a, b] est associé à chaque transition. Si une transition est restée sensibilisée pendant *a* unités de temps, alors elle *peut* être tirée. Aussi, si une transition est restée sensibilisée pendant b unités de temps, elle *doit* être tirée. L'intervalle [a, b] associe donc une incertitude à la date de tir de la transition qui va être tirée entre *a* et *b* unités de temps relativement à sa date de sensibilisation (Boyer 2001)(Valette 1994).

Toutes les valeurs de $\theta_s(t)$ telles que :

$$a \leq \theta_s(t) \leq b$$

sont des durées de sensibilisation possibles pour t. Un événement (franchissement de la transition t par exemple) est contraint d'arriver à au moins *a* unités de temps et au plus à *b* unités de temps après la sensibilisation de t.

Définition 6 : Un réseau de Petri temporel est une paire < R, I > où :

- R est un réseau de Petri < P, T, Pre, Post > auquel est associé un marquage initial M_0,

- I est la fonction d'intervalle initial qui associe à chaque transition un intervalle fermé rationnel, I(t) = [a, b] décrit une durée de sensibilisation de la transition *t*.

Grâce à leur structure, les réseaux de Petri temporels ont l'avantage de représenter naturellement les chiens de garde.

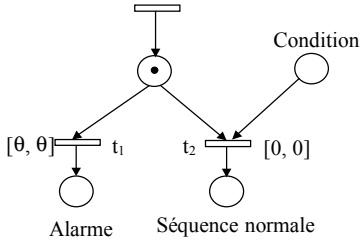


Figure 1. *Représentation d'un chien de garde*.

Dans la figure 1, le marquage de la place « Condition » signifie l'occurrence d'un événement. La transition t_2 est, de ce fait, tirée. Si au bout de θ unités de temps l'événement ne s'est pas produit, alors la transition t_1 est tirée et une alarme est déclenchée.

IV.2.3. Réseau de Petri p-temporel

Les réseaux de Petri p-temporels ont été introduits par (Khansa 1997). Dans ce type de réseaux un intervalle temporel est associé aux places. Un jeton dans une place *p* à laquelle est associé un intervalle [a, b] doit être consommé au plus tôt *a* unités de temps et au plus tard *b* unités temps après le marquage de la place *p*. Si un jeton n'est pas consommé avant *b* unités de temps, alors il devient un jeton « mort ».

Définition 7 : Un réseau de Petri p-temporel est une paire < R, I > où :

- R est un réseau de Petri < P, T, Pre, Post > auquel est associé un marquage initial M_0,
- I est la fonction d'intervalle initial qui associe à chaque place un intervalle fermé rationnel, I(p) = [a, b] décrit une durée de marquage de la place *p*.

En général, ce type de réseau représente des durées séparant des occurrences d'événements. Ces durées peuvent, par exemple, être des durées opératoires séparant un événement *début* et un événement *fin* d'une même opération *Op* sur une pièce. L'exécution de *Op* est modélisée par un jeton dans une place. Dans ce cas, si le jeton correspondant à l'opération *Op* en cours devient un jeton « mort », alors ceci permet de déduire que la pièce devant subir cette opération est restée trop longtemps dans cet état. D'un point de vue de la surveillance, ceci permet la détection d'un symptôme de défaillance lié à l'opération.

IV.2.4. Comparaison des deux modèles

L'avantage des réseaux de Petri temporels par rapport aux réseaux de Petri p-temporels est la modélisation des chiens de garde. Il est en effet impossible de modéliser un chien de garde avec le modèle p-temporel (Boyer 2001). Il est possible de représenter un réseau de Petri p-temporel par un réseau de Petri temporel, (figure 2).

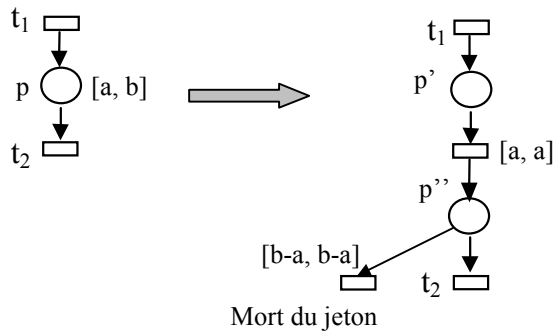


Figure 2. Traduction d'un modèle en l'autre.

Dans la figure 2, le franchissement de la transition t_1 induit le marquage de place *p'* dont le jeton ne peut être consommé qu'après *a* unités de temps. Lorsque la place *p''* est marquée, la transition t_2 doit être tirée avant *b-a* unités de temps.

L'utilisation des deux types de réseaux dans la modélisation de chronique relève de la simplification du modèle global de celle-ci par rapport au cas où un seul type de réseau de Petri (temporel ou p-temporel) est utilisé pour la modélisation.

IV.3. Modélisation de chronique et principes de base

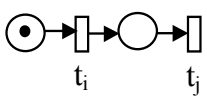
Pour modéliser les chroniques (Ghallab 1998), nous nous basons sur les réseaux de Petri temporels et p-temporels. Les contraintes de précédence sont implicitement modélisées par le séquencement de franchissement des différentes transitions. Rappelons que le franchissement d'une transition est associé à l'occurrence d'un événement. Les contraintes de type fenêtre d'admissibilité sont associées aux transitions comme dans le cas des réseaux de Petri temporels. Les contraintes de type intervalle sont associées, quant à elles, aux places comme dans le cas des réseaux de Petri p-temporels. En combinant les réseaux de Petri temporels et p-temporels, nous obtenons les réseaux de Petri p-t-temporels. Nous présentons, par la suite, les aspects dynamiques et statiques liés à la modélisation de chroniques utilisant les réseaux de Petri p-t-temporels.

IV.3.1. Etude des aspects statiques de la modélisation

Sachant qu'une chronique est un ensemble d'événements et de contraintes liant ces événements, sa modélisation doit être précédée par la modélisation des contraintes la composant. Une fois cette étape achevée, la modélisation de la chronique est faite en combinant les différents modèles de contraintes obtenant ainsi le modèle réseau de Petri p-t-temporel de la chronique.

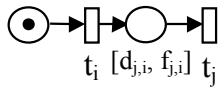
IV.3.1.1. Modélisation des contraintes

La modélisation des trois types de contraintes définies précédemment est faite par Réseaux de Petri p-temporels et t-temporels (Boufaied et al 2002). L'occurrence d'un événement en provenance du système surveillé est associée au franchissement d'une transition. Dans ces conditions, toute transition sensibilisée doit être tirée dans l'intervalle de temps qui lui est associé et tout jeton doit être consommé dans l'intervalle de temps qui lui est associé. Nous obtenons les différentes modélisations suivantes :

- La contrainte de précédence est modélisée par : 

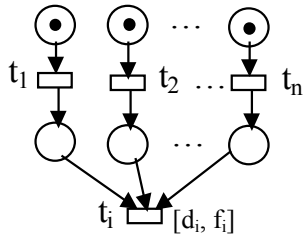
Dans ce modèle représentant une contrainte de précédence, la transition t_j associée à l'occurrence de l'événement e_j est tirée après la transition t_i associée à l'occurrence de l'événement e_i. Ceci modélise la relation de précédence existant entre les deux événements.

- La contrainte de type intervalle est modélisée par :



Après le tir de la transition t_i, le jeton produit ne peut être consommé par le tir de la transition t_j que s'il a séjourné, dans la place, une durée comprise entre les bornes de la contrainte C_{ji}, c'est à dire appartenant à l'intervalle $[d_{j,i}, f_{j,i}]$.

- La contrainte de type fenêtre d'admissibilité est modélisée par :



Le tir de la transition t_i ne peut avoir lieu qu'après le tir de toutes les transitions t_j, j = 1..n. Ce tir s'effectuera après une durée de sensibilisation de t_i comprise entre les bornes de la contrainte D_i, c'est à dire appartenant à $[d_i, f_i]$.

Signalons enfin qu'à une place ou à une transition est associé par défaut l'intervalle $[0, +\infty[$.

Abordons maintenant la modélisation d'une chronique au travers d'un exemple.

IV.3.1.2. Modélisation d'une chronique

Considérons une chronique constituée de l'ensemble des événements $\{e_1, e_2, e_3\}$ liés par les contraintes temporelles suivantes :

$$0 \le O(e_3) - O(e_1) \le 1 \quad : C_{3\,1}$$
$$1 \le \min_{p}(O(e_3) - O(e_p)) \le 3 \quad : D_3$$
$$O(e_2) < O(e_3)$$

En se basant sur les principes de modélisation des contraintes donnés dans le paragraphe précédent, le regroupement des modèles de contraintes permet d'obtenir le modèle réseau de Petri de la chronique, figure 3. Le regroupement consiste à fusionner les modèles des contraintes en un seul modèle représentant la chronique. Cette fusion concerne toutes les transitions associées à l'occurrence d'un même événement et toutes les places $p_1, p_2, \ldots, p_n$ telles que pour tout i =1..n :

- $\exists$ t, t'$\in$T, Pre $(p_i, t) \neq 0$ et Post $(p_i, t') \neq 0$, toutes les places succédant aux transitions associées à l'occurrence d'un même événement, et précédant des transitions associées à l'occurrence d'un même événement sont fusionnées. Sur la figure 3, un exemple correspond à la fusion des places p_1 et p_2. Ainsi, quand il existe une contrainte de type intervalle ou une contrainte de précédence en plus d'une contrainte de type fenêtre d'admissibilité entre deux événements, cette fusion nous permet d'obtenir un seul modèle représentant les deux contraintes.

- ou $\exists$ t$\in$T, Pre (p_i, t) $\neq$ 0 et Post (p_i, t') = 0 pour tout t'$\in$T, toutes les places précédant des transitions associées à l'occurrence d'un même événement et ne succédant aucune transition sont fusionnées. Sur la figure 3, un exemple correspond à la fusion des places p_3 et p_4. Ceci permet d'exprimer l'attente commune à plusieurs contraintes de l'occurrence d'un même événement.
- ou $\exists$ t$\in$T, Post (p_i, t) $\neq$ 0 et Pre (p_i, t') = 0 pour tout t'$\in$T, toutes les places succédant des transitions associées à l'occurrence d'un même événement et ne précédant aucune transition sont fusionnées. Sur la figure 3, un exemple correspond à la fusion des places p_5 et p_6. Ceci permet d'exprimer la reconnaissance par plusieurs contraintes de l'occurrence d'un même événement.

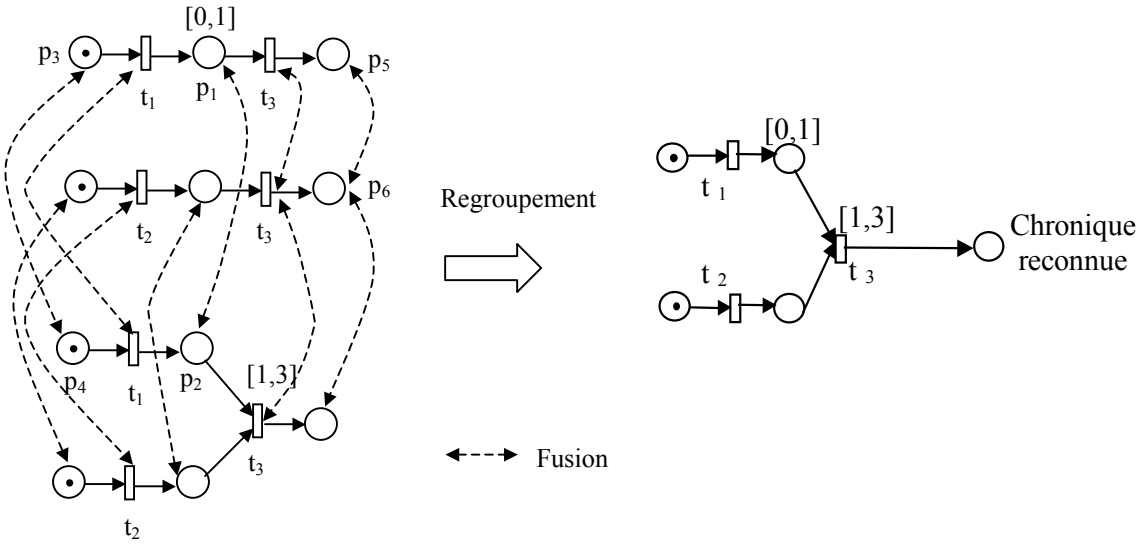


Figure 3. *Modélisation de la chronique.*

L'intervalle associé à une place (respectivement à une transition) résultant de la fusion est l'intersection de tous les intervalles associés aux places (respectivement aux transitions) fusionnées. Si la place « chronique reconnue » est marquée, cela indique que la chronique a été reconnue et que toutes les contraintes la constituant sont vérifiées.
Signalons que les configurations représentées par la figure 4 ne peuvent pas avoir lieu dans notre modèle réseau de Petri de la chronique.

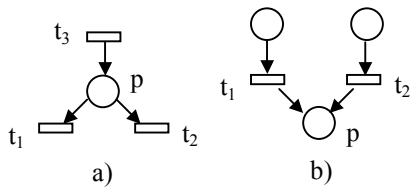


Figure 4. Configurations inexistantes.

Ceci est dû au fait que ces configurations n'existent pas dans les modèles représentant les contraintes et qu'elles ne peuvent pas être générées suite au regroupement. En effet dans le cas de la figure 4,a), il est impossible d'avoir la place *p* comme une fusion de deux places d'entrée aux transitions t_1 et t_2. De même, la place *p* de la figure 4,b) ne peut pas être obtenue pour des raisons similaires. L'impossibilité associée au cas a) se traduit au niveau des contraintes par le fait qu'elles sont vérifiées indépendamment l'une de l'autre. En d'autres termes, la vérification par exemple de la contrainte liant les événements e_3 (associé à l'occurrence de t_3) et e_1 (associé à l'occurrence de t_1)

ne doit pas influencer la vérification de la contrainte liant les événements e_3 et e_2 (associé à l'occurrence de t_2). De même, le cas b) ne peut pas avoir lieu car ceci traduit le fait que les occurrences des événements e_1 et e_2 vont être confondues alors qu'ils se sont produits à des dates différentes. Une contrainte mettant en jeu l'occurrence de e_1 ou de e_2 et devant être vérifiée, ne peut l'être à cause de la confusion engendrée sur les occurrences de e_1 et de e_2.

Cette approche de regroupement, bien que traitée sur un exemple, peut être appliquée dans un cas général où plusieurs contraintes des trois types définis coexistent. En effet, les règles présentées dans ce paragraphe peuvent être appliquées sur *m* modèles de contraintes. Nous obtenons ainsi un modèle global, que nous avons appelé chronique, permettant de représenter toutes les contraintes modélisées et mettant en place une bijection entre une transition du réseau et l'occurrence d'un événement.

Après avoir détaillé l'aspect statique de la modélisation d'une chronique, nous formalisons, dans le paragraphe suivant, les principes relevant de l'aspect dynamique liés à la modélisation et associés à la reconnaissance de chronique.

IV.3.2. Etude des aspects dynamiques liés à la modélisation

L'évolution du marquage du réseau de Petri modélise la progression du processus de reconnaissance de la chronique en reconnaissant de nouvelles occurrences des événements. A partir de là, nous définissons la notion *d'instance de chronique en cours* comme étant une séquence des événements reconnus. Une chronique est dite *reconnue* si une instance en cours contient tous les événements figurant dans les contraintes et si les contraintes sont vérifiées. Une instance en cours de la chronique « meurt » si une contrainte n'est pas vérifiée ou ne peut plus être vérifiée. Dans le modèle réseau de Petri, l'évolution de l'instance en cours de la chronique est conditionnée par le franchissement de transitions associées à l'occurrence d'événements.

Afin de décrire ces aspects dynamiques, nous définissons formellement l'évolution du marquage des réseaux de Petri p-t-temporels modélisant une chronique.

IV.3.2.1. Les réseaux de Petri p-t-temporels

Nous donnons dans ce paragraphe des définition formelles des réseaux de Petri p-t-temporels utilisés pour modéliser une chronique ainsi que les mécanismes d'évolution de leur marquage dans le cas où un événement possède une occurrence unique. Un événement à occurrence unique est un événement se produisant une seule fois dans le procédé.

Définition 8 Un réseau de Petri p-t-temporel est un 7-uplet $<P,T,Pre,Post,M_0,IST,ISP>$ dans lequel $<P,T,Pre,Post,M_0>$ est un réseau de Petri marqué, $IST : T \rightarrow Q^+ \times (Q^+ \cup \{+\infty\})$ est la fonction intervalle initial sur les transitions, et $ISP : P \rightarrow Q^+ \times (Q^+ \cup \{+\infty\})$ est la fonction intervalle initial sur les places.
L'application IST associe à chaque transition *t* du réseau un intervalle à bornes rationnelles IST(t) = [mint, maxt] avec $0 \leq mint \leq maxt$ et maxt pouvant être $+\infty$. Quant à l'application ISP, elle associe à chaque place *p* du réseau un intervalle à bornes rationnelles ISP(P) = [minp, maxp] avec $0 \leq minp \leq maxp$ et maxp pouvant être $+\infty$.

Définition 9 Un état d'un réseau p-t-temporel est un triplet E = (M,I,J) dans lequel M est un marquage, I est l'application associant un intervalle temporel dynamique à chaque transition et J est l'application associant un intervalle temporel dynamique à chaque jeton.

L'état initial du réseau de Petri p-t-temporel est constitué du marquage initial M_0, de l'application I_0 qui associe à chaque transition sensibilisée *k* son intervalle dynamique et l'application J_0 qui associe à chaque jeton p^i (i est un entier naturel), d'une place p et du marquage initial, son intervalle dynamique (Khansa 1997) (Berthomieu 2001). Soit E_0 l'état initial du réseau de Petri p-t-temporel. Nous supposons qu'initialement pour toute place *p* du réseau de Petri M(p) ≤ 1, nous avons donc :

$$E_0 = (M_0, I_0, J_0), \text{ avec } I_0(k) = IST(k) \text{ si } M_0 \geq Pre(., k)$$
$$= \varnothing \quad \text{sinon.}$$
$$\text{et } J_0(p^i) = ISP(p) \text{ si } M_0(p) = 1$$
$$= \varnothing \quad \text{sinon.}$$

a) Transition franchissable

Rappelons que toute transition sensibilisée doit être tirée dans l'intervalle de temps qui lui est associé. Cet intervalle est relatif à la date de sensibilisation de la transition. De la même manière, chaque jeton doit être consommé dans l'intervalle de temps qui lui est associé. Cet intervalle est relatif à la date d'apparition du jeton dans la place.

Dans ces conditions, franchir *t* (prendre en compte l'occurrence d'un événement *e*), à une date θ relative à la sensibilisation de *t*, depuis un état E = (M,I,J), est permis si et seulement si :

1- $M \geq Pre(., t) \wedge \theta \in I(t) \wedge \forall k \neq t, M \geq Pre(., k) \Rightarrow \theta \leq Max(I(k))$, l'occurrence de l'événement *e* est associée à la transition *t* et celle de l'événement *e'* est associée à la transition *k*.
2- l'intervalle $[a_l, b_l]$ d'intersection de tous les intervalles associés aux jetons dans les places d'entrée de *t* doit être non vide (Khansa 1997),
3- $O(e) \in [a_l, b_l]$. L'événement *e* doit se produire dans l'intervalle $[a_l, b_l]$.

b) Franchissement d'une transition

Le nouvel état E'= (M',I',J') atteint depuis E par le tir de *t* est déterminé par :

- M' = M – Pre(t) + Post(t) (comme dans les réseaux de Petri marqués).
- Pour chaque transition k :
  si k est non sensibilisée par M', alors I'(k) = ∅.
  si k est distincte de t, sensibilisée par M et par M', alors

$$I'(k) = [max(0,Min(I(k))-\theta), Max(I(k))-\theta],$$

avec θ est la durée séparant la sensibilisation de k et le tir de la dernière transition franchie t.

sinon, I'(k) = IST(k).

c) Consommation des jetons

Pour chaque place p contenant un jeton p^i,
si p est non marquée dans M', alors J'(p) = ∅.
si p est marquée dans M et M', alors

$$J'(p^i) = [max(0,Min(J(p^i))-\theta), Max(J(p^i))-\theta]$$

avec θ est la durée séparant le marquage de *p* dans M et le marquage de *p* dans M'.

sinon, $J'(p^i) = ISP(p)$.

Tout jeton doit être consommé dans l'intervalle de temps qui lui est associé, son échéance est la borne maximale de cet intervalle. Soit *p* la place contenant un jeton p^i. Ce jeton sera retiré du réseau de Petri (utilisation de chiens de garde) ou considéré comme « mort » si et seulement si :

- ce jeton n'est pas consommé dans $J(p^i)$ c'est à dire qu'il atteint son échéance sans être consommé.
- la transition *t* tel que $M(p) \geq Pre(p,t)$, sensibilisée par p^i, ne peut plus être tirée dans l'intervalle de tir qui lui est associé.

Un événement est attendu par une instance de la chronique si la transition associée à l'occurrence de ce même événement est sensibilisée. Pour tout autre événement $e' \neq e$, attendu aussi par une instance de la chronique, la borne maximale de la fenêtre temporelle de *e'* ne doit pas être dépassée lors du tir de *t* sinon l'instance de la chronique est considérée « morte » et la chronique ne peut plus être reconnue.
Quand un jeton atteint sa date limite sans être consommé, l'instance de la chronique ne peut plus reconnaître tous les événements, cette instance ne peut plus conduire à la reconnaissance de la chronique. Quand une transition, sensibilisée, ne peut pas être tirée dans l'intervalle de franchissement qui lui est associé, une contrainte n'est pas vérifiée et la chronique ne peut plus être reconnue.

Après avoir traité le cas où un événement possède une occurrence unique, nous étudions, dans la suite, le cas le plus général dans lequel un événement possède plusieurs occurrences.

IV.3.2.2. Prise en compte des occurrences multiples d'un événement

Le problème de la reconnaissance d'une chronique dans le cas où un événement possède des occurrences multiples, c'est à dire qu'il se produit plusieurs fois dans le système, peut être résolu de deux manières. La première a été introduite par (Mok et al 1997), consiste à spécifier les occurrences des événements dans les contraintes. Ceci permet de considérer les occurrences d'un même événement comme des événements différents. La deuxième solution, utilisée par (Dousson et al 1994), consiste à prendre en compte les occurrences multiples d'un événement au fur et à mesure de l'évolution du processus de reconnaissance de la chronique.

IV.3.2.2.1. Spécification des occurrences multiples d'un événement dans les contraintes

Comme nous l'avons introduit au § II.5.2, un événement, est dans ce cas, associé à une occurrence particulière. La date d'occurrence d'une instance *i* d'un événement est supérieure à celle de l'instance *i-1* du même événement, $\forall\ i \in N^+$ (Mok et al 1997). Pour expliquer cette approche nous considérons un exemple simple.

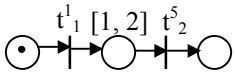


Figure 5. *Exemple de modélisation de contrainte liant des occurrences particulières d'événements.*

La figure 5 représente la contrainte $1 \leq \Theta(e_2, 5) - \Theta(e_1, 1) \leq 2$, liant la première occurrence de e_1 à la cinquième occurrence de e_2.

Dans toutes les contraintes temporelles, un événement *e* est nécessairement lié à une occurrence *i*. Ceci suppose l'existence d'un processus qui estampille les événements et qui permet ainsi d'associer une étiquette ou un indice global à l'occurrence d'un événement.
Cette méthode présente l'avantage de favoriser la simplification des mécanismes de vérification de contraintes, et ainsi la reconnaissance de chronique. En considérant les occurrences d'un événement comme des événements différents, les mécanismes que nous avons présentés pour les réseaux de Petri p-t-temporel n'ont pas à être étendus ou modifiés. Par contre, la spécification et la modélisation des contraintes sont d'autant plus compliquées que le nombre d'occurrences d'événements utilisées est grand.

IV.3.2.2.2. Prise en compte des occurrences multiples d'un événement lors de la reconnaissance de la chronique

Dans ce cas, seul l'événement est considéré dans la spécification des contraintes. Les occurrences multiples d'un événement sont prises en compte dans la dynamique de vérification des contraintes et de reconnaissance de la chronique. Ainsi, chaque occurrence d'un événement est intégrée dans un modèle de reconnaissance de chronique. (Dousson et al 1994) a proposé de dupliquer le modèle de reconnaissance de chronique afin de pouvoir gérer plusieurs instances de chronique en parallèle. Pour cela, une structure arborescente a été adoptée. Dans ce cas chaque nœud de l'arborescence représente une instance.

Considérons les contraintes suivantes : $e_1 < e_2$ et $1 \leq O(e_3) - O(e_2) \leq 2$. Supposons également que le flot des événements d'entrée se compose de e_1 à 0s suivi de e_2 à 1s, suivi d'une autre occurrence de e_2 à 2s et d'un e_3 à 3.5s. Si le modèle de reconnaissance prend en compte la première occurrence de e_2, il ne peut reconnaître sa deuxième occurrence qui permet de vérifier la contrainte de type intervalle. En effet, la reconnaissance de la première occurrence de e_2 ne permet de vérifier la contrainte puisque la durée séparant cette occurrence à celle de e_3 est supérieure à la durée représentée par la contrainte. L'instance en cours est ainsi considérée morte.

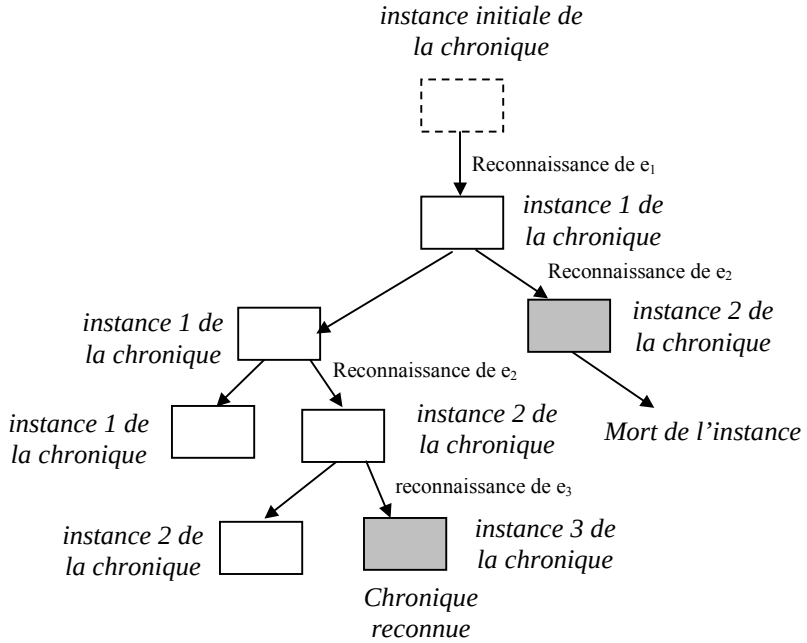


Figure 6. *Arborescence représentant la reconnaissance d'instances d'une chronique.*

La duplication de l'instance en cours de la chronique est donc nécessaire à chaque reconnaissance d'un événement, figure 6. Cette duplication se fait de manière systématique puisque le processus de reconnaissance de chronique n'a aucune information quant à la reproduction ou pas de l'occurrence de l'événement reconnu.
Nous allons montrer, par la suite, l'apport des réseaux de Petri dans la modélisation et l'analyse des occurrences multiples d'un événement.

IV.3.2.2.3. Utilisation des réseaux de Petri pour la prise en compte des occurrences multiples d'un événement

La prise en compte des occurrences multiples d'un événement par RdP peut se faire de deux manières. La première, intègre les mécanismes de prise en compte des occurrences multiples d'un événement dans le réseau même. Les places du RdP peuvent ainsi être marquées par plusieurs jetons par tirs successifs des transitions les précédant. Il est donc nécessaire d'intégrer dans le modèle des mécanismes permettant de gérer la *multi-sensibilisation*. La deuxième manière, consiste à gérer les occurrences multiples d'événements en dupliquant le réseau de Petri à chaque franchissement de transition.

- **Notion de multi-sensibilisation**

Une transition *t* est multi-sensibilisée par un marquage M, s'il existe un entier k > 1 tel que M ≥ k**.**Pre(., t) (Berthomieu 2001). Un exemple est donné à la figure 7.

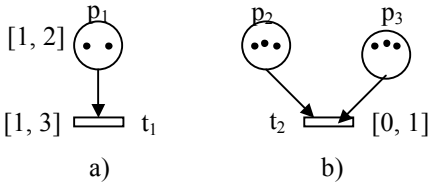


Figure 7. *Exemple de multi-sensibilisation.*

Dans la figure 7,a), au premier jeton p^1_1 apparu dans la place p_1 est associé l'intervalle de consommation [1, 2]. La transition t_1 est alors sensibilisée. En tenant compte de l'intervalle de consommation de p^1_1 et de l'intervalle de tir de t_1, celle-ci peut être tirée après une durée appartenant à l'intervalle [1, 2] = [1, 2] ∩ [1, 3] après l'apparition de p^1_1. Supposons qu'après une durée de 0.5s, un deuxième jeton p^2_1 est déposé dans la place p_1. L'intervalle de consommation [1, 2] est associé à ce jeton. Nous obtenons ainsi un nouveau marquage. L'intervalle de consommation associé à p^1_1 est ainsi mis à jour utilisant la durée 0.5s. Le nouvel intervalle qui lui est associé est [0.5, 1.5]. La transition t_1 devient maintenant multi-sensibilisée. Par rapport à l'origine des temps représentée par l'apparition de p^2_1, t_1 peut être tirée après une durée appartenant à l'intervalle [1, 2] en consommant p^2_1 ou après une durée appartenant à l'intervalle [0.5, 1.5] = [1-0.5, 2-0.5] en consommant p^1_1, (cf § IV.3.2.1.), (figure 8).

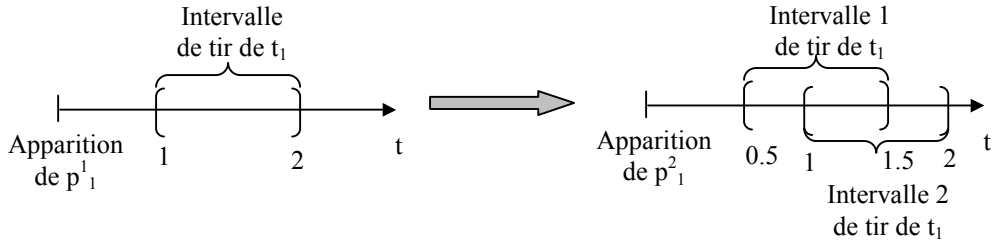


Figure 8. *Exemple illustratif de mise à jour de l'intervalle de tir.*

En gardant toujours l'apparition de p^2_1 comme origine des temps, nous remarquons que dans l'intervalle [1, 1.5], (figure 8), la transition t_1 peut être tirée en consommant p^1_1 ou p^2_1. Pour résoudre cette situation de conflit relative à la consommation des jetons, deux stratégies peuvent être utilisées :

- les jetons sont considérés indépendamment. Cette stratégie, bien que générale, est non déterministe. En effet, lors du tir de la transition il faut spécifier le(s) jeton(s) à consommer,
- les jetons sont ordonnés selon leur âge. Ainsi les jetons sensibilisant la transition sont consommés selon leur âge et la transition tirée selon la stratégie *PSPT* (Première Sensibilisée – Première Tirée), (Berthomieu 2001). Cette stratégie est restrictive puisqu'elle ne prend pas en compte tous les cas de consommation des jetons. En effet, à chaque occurrence d'un événement, il faut envisager tous les tirs possibles de la transition qui lui est associée en consommant un jeton (respectivement, des jetons) quelconque(s) de la place (respectivement, des places). Ceci nous amène à aborder le problème de la duplication du réseau de Petri.

- **Duplication de RdP**

Une autre solution permettant de traiter le problème de l'occurrence multiple d'un événement est la duplication du réseau de Petri p-t-temporel. En effet, à chaque fois qu'une transition doit être franchie, un réseau de Petri est dupliqué à partir du réseau initial, et un nouveau réseau est crée avec le marquage obtenu par le tir de la transition. Un exemple est montré à la figure 9. Dans cet exemple toutes les dates relatives aux occurrences des événements sont référencées par rapport à un repère absolu.

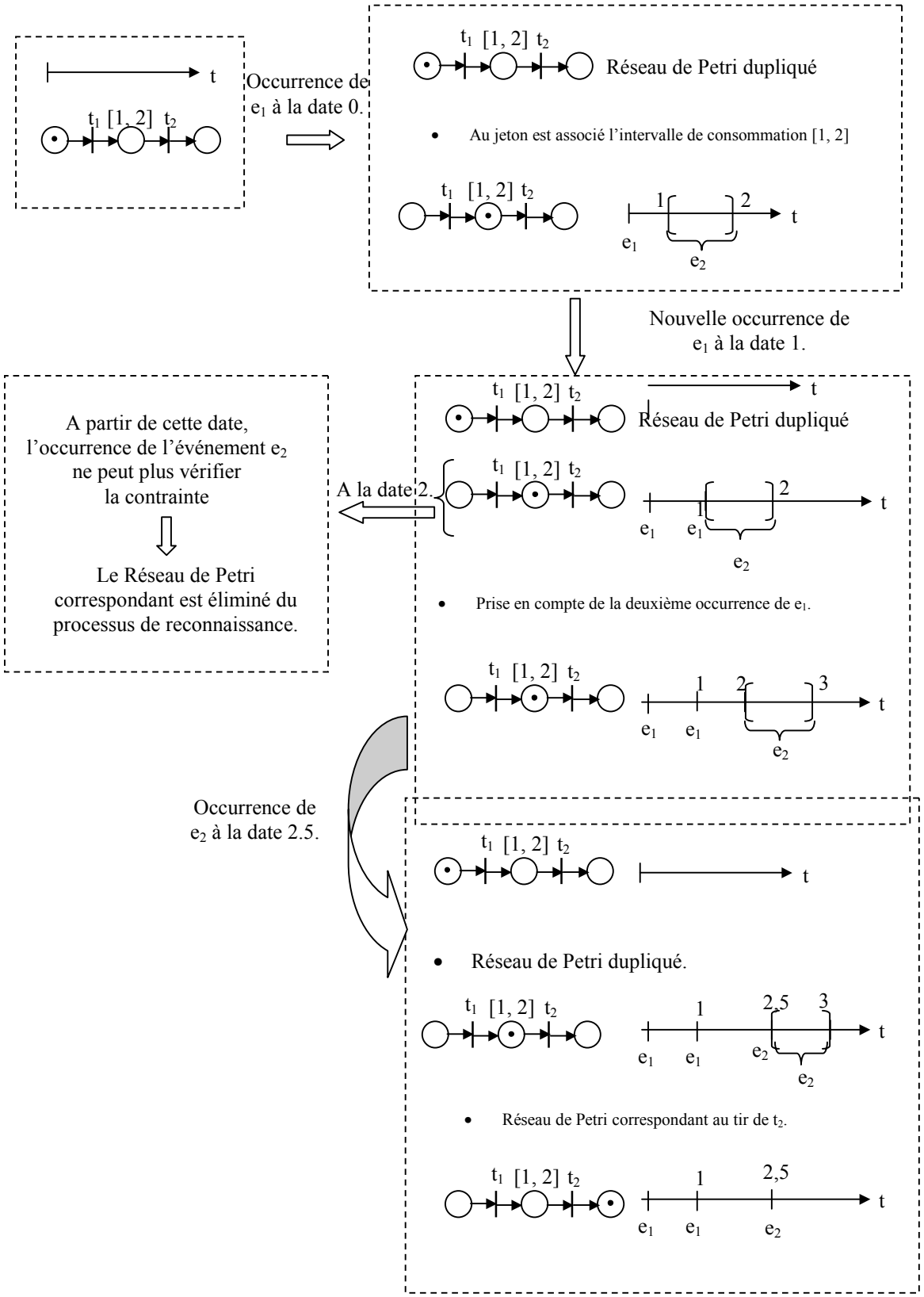


Figure 9. *Exemple de duplication du réseau de Petri.*

Les occurrences des événements sont datées par rapport à un repère temporel global alors que les jetons sont consommés dans des intervalles relatifs à la date d'obtention du marquage courant. Si un jeton est mort dans réseau de Petri, celui-ci est retiré du processus de reconnaissance.
Nous supposons, par la suite et sans perte de généralité, que tous les événements ont une occurrence unique.
Après avoir étudié les aspects relevant de la modélisation d'une chronique, nous nous intéressons maintenant à la modélisation d'une sous-chronique par réseaux de Petri.

IV.4. Modélisation de sous-chronique sans prise en compte des délais

La modélisation d'une sous-chronique par réseaux de Petri peut se faire de deux manières. La première méthode se base sur le modèle réseau de Petri de la chronique. En décomposant le modèle réseau de Petri de la chronique selon des places, nous obtenons plusieurs sous-modèles réseaux de Petri représentant chacun une sous-chronique à surveiller. La deuxième méthode, construit à partir des modèles de contraintes un ensemble de sous-réseaux de Petri représentant les sous-chroniques à surveiller.

IV.4.1. Décomposition du réseau de Petri représentant une chronique

Plusieurs travaux se sont intéressés à la distribution de modèles de réseaux de Petri atemporels. (Da Silveira 2002) a proposé une procédure systématique permettant d'obtenir un modèle distribué avec redondance partielle à partir d'un modèle centralisé spécifié par réseaux de Petri en se basant sur la théorie des invariants linéaires. Pour (Ben Khalifa 1997), le mode de décomposition le plus connu est appelé *décomposition spatiale*. Ceci signifie qu'un réseau de Petri est décomposé en plusieurs sous-réseaux. Pour ce type de décomposition, il existe trois manières de procéder : décomposition selon les places, décomposition selon les transitions, ou en utilisant une approche mixte. La décomposition selon les transitions conduit à la possibilité d'utiliser un modèle synchrone de communication ou de coopération. La décomposition selon les places met en œuvre des données en commun entre les sous-réseaux et conduit à la possibilité d'utiliser un modèle asynchrone de communication permettant d'échanger ces données entre les différents réseaux.

Dans le cadre de cette thèse, nous nous proposons de décomposer un réseau de Petri représentant une chronique en un ensemble de sous-chroniques communicantes et coopérantes. Dans le chapitre III, nous avons montré que la communication et la coopération entre sous-chroniques se font de manière asynchrone, sur réception de l'occurrence d'un événement particulier. Ceci nous amène donc à adopter un modèle de communication et de coopération asynchrone. Ainsi, la décomposition du modèle réseau de Petri de la chronique se fera selon les places, (figure 10).

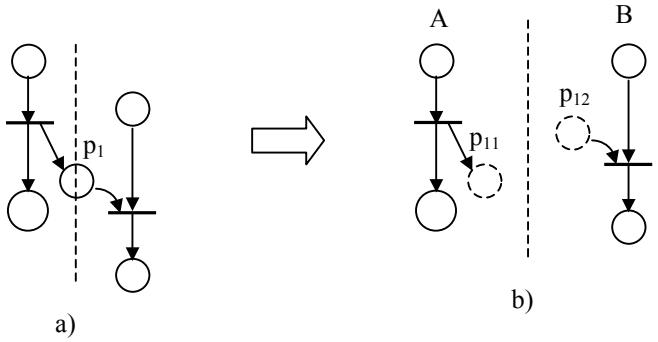


Figure 10. *Modèle asynchrone de communication et de coopération.*

Dans la figure 10,a), la décomposition selon la place p_1 donne lieu à deux nouvelles places p_{11} et p_{12}, figure 10,b). La place p_{12} est, dans ce cas, appelée *place d'entrée*, alors que p_{11} est appelée *place de sortie*. Lorsque la transition du modèle *A* est franchie, celle-ci dépose un jeton dans la place p_{11} qui doit être envoyé au modèle *B* qui le reçoit dans la place p_{12}. Si un intervalle temporel est associé à la place selon laquelle se fait la décomposition, celui-ci sera seulement associé à la place d'entrée obtenue.
La seule règle à respecter dans une décomposition selon les places est que toutes les transitions de sortie d'une place doivent appartenir au même réseau de Petri, c'est à dire à la même sous-chronique. Ceci représente un cas de conflit (cf § IV.2.1) qui doit être résolu dans la même sous-chronique. Comme nous nous intéressons aux chroniques comme étant des conjonctions de contraintes temporelles, notre modèle temporel est par définition exempt de tout conflit.
Nous présentons l'approche de décomposition à travers un exemple.

Exemple de décomposition

Nous présentons dans cet exemple une chronique modélisée par réseau de Petri p-t-temporel représentant cinq opérations à exécuter par cinq ressources distinctes, c'est à dire physiquement distribuées, sur des pièces, (figure 11). La décomposition de cette chronique en sous-chroniques permet de visualiser les mécanismes de décomposition selon les places présentés précédemment.

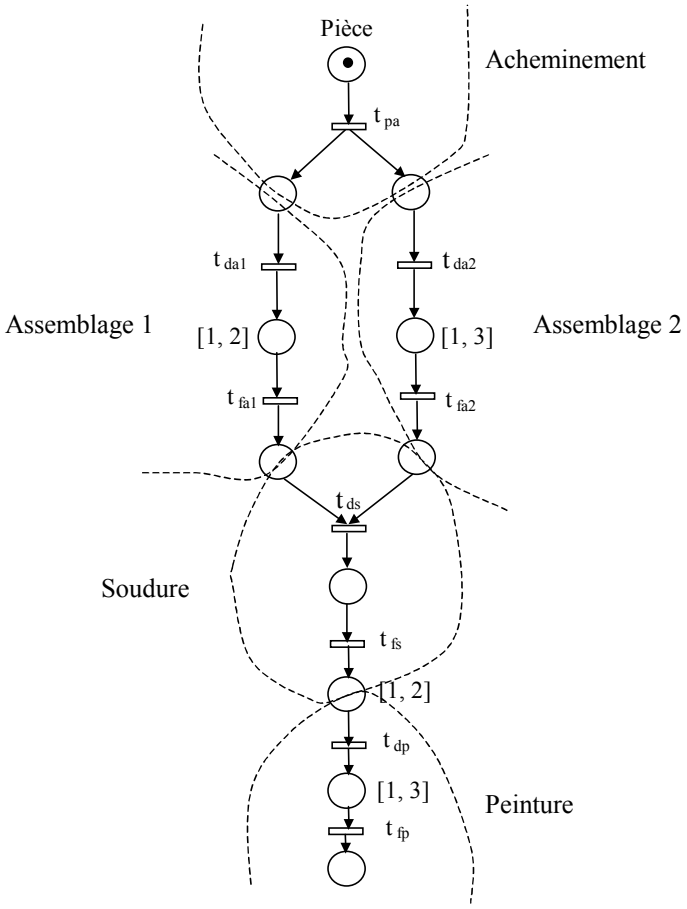


Figure 11. *Chronique à décomposer associée aux opérations à effectuer.*

Les opérations à effectuer par les ressources se composent de : acheminement de pièces (événement e_{pa}), assemblage1 de pièces (événements e_{da1} et e_{fa1}), assemblage2 de pièces (événements e_{da2} et

e_{fa2}), soudure (événements e_{ds}, e_{fs}), et peinture (événements e_{dp} et e_{fp}). Quand une pièce est arrivée au niveau des deux premières ressources, commencent alors deux opérations d'assemblage en parallèle. Après la fin de ces deux opérations, la soudure commence. Quand la soudure est terminée, l'opération de peinture de la pièce est exécutée.

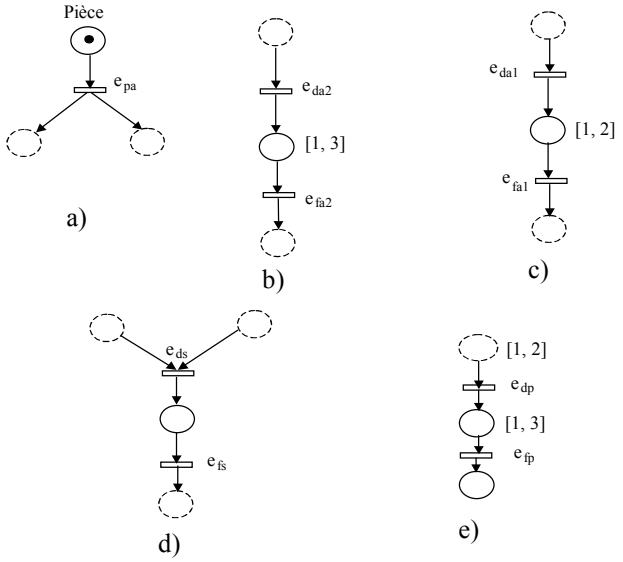


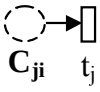
Figure 12. *Sous-chroniques obtenues par décomposition de la chronique.*

La décomposition proposée de la chronique de la figure 8 donne les sous-chroniques a), b), c), d) et e) représentées par la figure 12. Les places en trait interrompu sont les places de communication d'entrée et de sortie.

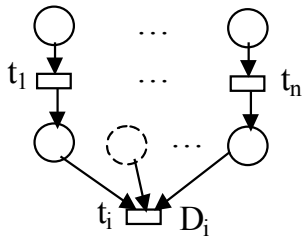
IV.4.2. Regroupement des réseaux de Petri représentant des contraintes temporelles

La procédure de distribution des contraintes temporelles présentée au chapitre III nous permet de connaître l'ensemble des contraintes temporelles associées à chaque sous-chronique. Si la contrainte est locale à la sous-chronique, nous obtenons les modèles réseaux de Petri des contraintes présentés au § IV.3.1.1. Si une contrainte est globale, elle correspond à l'un des modèles suivants, sachant qu'une place en trait interrompu représente une place d'entrée du modèle :

- La contrainte de précédence est modélisée par : 

- La contrainte de type intervalle est modélisée par : 

- La contrainte de type fenêtre d'admissibilité est modélisée par :



L'application de mécanismes de regroupement similaires à ceux explicités au § IV.3.1.2. permet ainsi d'obtenir les modèles réseaux de Petri des différentes sous-chroniques.

Après avoir étudié le cas où les délais ne sont pas pris en compte, nous abordons maintenant le problème de la prise en compte des délais, qui sont comme nous l'avons dit nécessaires pour la vérification des contraintes, dans la modélisation de sous-chronique.

IV.5. Modélisation de sous-chronique avec prise en compte des délais

Au chapitre III, nous avons étudié le problème de la vérification distribuée de contraintes temporelles en présence de délai. Nous nous intéressons maintenant au problème de leur modélisation par réseaux de Petri.
Afin de prendre en compte la notion floue qu'est la possibilité de vérifier une contrainte qui a été définie au chapitre III, nous avons étendu la définition des réseaux de Petri p-t-temporels aux réseaux de Petri p-t-temporels flous. Ces réseaux de Petri se basent sur les réseaux de Petri flous introduits pas (Cardoso et al 96) et permettent de modéliser les mécanismes susceptibles de vérifier les contraintes globales moyennant une possibilité.

Définition : Un réseau de Petri p-t-temporel flou est défini comme un 10-tuple $<P,T,Pre,Post,M_0,IST,ISP,\mu,\alpha>$ dans lequel $<P,T,Pre,Post,M_0>$ est un réseau de Petri, IST: $T \rightarrow Q^+ \times (Q^+ \cup \{+\infty\})$ est la fonction intervalle statique sur les transitions, ISP: $P \rightarrow Q^+ \times (Q^+ \cup \{+\infty\})$ est la fonction intervalle statique sur les places, μ est une fonction d'appartenance associée au tir de chaque transition, et α : $P \rightarrow [0, 1]$ est une fonction associative qui établit une valeur de possibilité α_j (nombre flou) aux jetons dans les places.

Le tir d'une transition *T*, à laquelle est associée la fonction d'appartenance μ, sera validé par la réception des événements e_j et e_k dont la mesure de la durée séparant leurs occurrences permet de vérifier une contrainte. La représentation d'un tel mécanisme est inspirée de la représentation des réseaux de Petri à synchronisation interne (Racoceanu et al 2002), (figure 13).

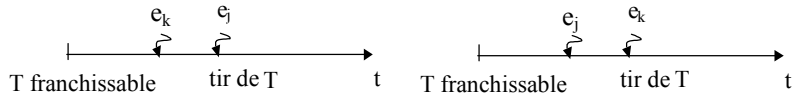


Figure 13. *Validation et tir de T.*

Cette représentation est justifiée par le fait qu'il est impossible de connaître à l'avance qui de e_j et de e_k est reçu en premier par le superviseur auquel est associée la contrainte. Pour cela, la transition *T* reste sensibilisée sur réception d'une occurrence de l'un des deux événements jusqu'à ce que le deuxième se produise.

Dans la suite, nous donnons les modèles réseaux de Petri p-t-temporel flous représentant les différents types de contraintes composant une sous-chronique.

IV.5.1. Modélisation d'une contrainte de type intervalle avec prise en compte des délais

La figure 14 donne le réseau de Petri p-t-temporel flou d'une contrainte de type intervalle C_{ji}. μ représente la fonction d'appartenance de la durée $O(e_j) - O(e_k)$ à l'ensemble flou A défini par

la variable linguistique « vérification de la contrainte C_{ji} ». Le marquage d'une place est associé à une valeur de possibilité α appartenant à l'intervalle [0, 1]. Dans la figure 14, la place *C* est marqué par un jeton en provenance du tir de t_i (occurrence de e_i). Comme la contrainte liant e_i à e_j est globale puisque l'occurrence de e_i n'est pas datée localement par le superviseur, la vérification de cette contrainte est conditionnée par le tir de la transition *T*. La valeur floue α_j, associée au marquage de la place B est égale à celle donnée par μ. La valeur donnée par μ est calculée grâce à ce qui a été présenté au § III.4.1.1 et au § III.4.2. La partie de la figure 14 représentée en trait interrompu est un modèle fictif ne faisant pas partie du modèle de la contrainte et représentant la contrainte à vérifier et le délai à prendre en compte.

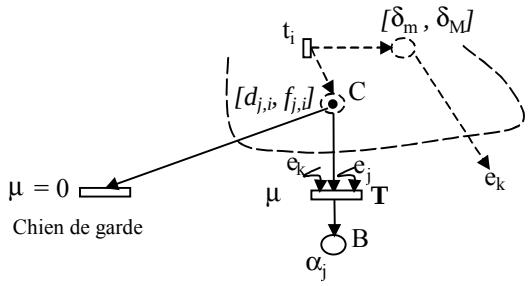


Figure 14. *Principe de modélisation de la contrainte C_{ji}.*

Des chiens de garde peuvent être déclenchés après l'écoulement d'une certaine durée, $f_{j,i} - \delta_m$ ou $f_{j,i} + \delta_M$ selon le cas, § III.4.1.1 et au § III.4.2, afin d'éviter une attente infinie. La valeur de un μ donnée par ce déclenchement est nulle, figure 14.

Considérons maintenant le second type de contrainte qu'est la contrainte de type fenêtre d'admissibilité.

IV.5.2. Modélisation d'une contrainte de type fenêtre d'admissibilité avec prise en compte des délais

La modélisation d'une contrainte de type fenêtre d'admissibilité avec un réseau de Petri p-t-temporel flou donne la figure 15. Nous supposons qu'un seul événement e_k nécessaire à la vérification de la contrainte est reçu avec délai par le superviseur. La vérification de la contrainte de type fenêtre d'admissibilité D_j se fait donc moyennant la prise en compte des occurrences des événements locaux et de e_k. Lorsque tous les événements faisant partie de la contrainte sont datés par le superviseur, celle-ci est vérifiée avec une possibilité égale au minimum entre la possibilité de vérification de la contrainte compte tenu de l'occurrence des événements locaux et la possibilité de vérification de la contrainte tenant compte de l'occurrence de l'événement e_k.

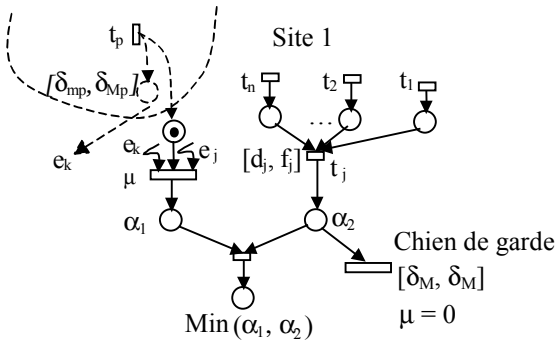


Figure 15. *Principe de modélisation de la contrainte D_j.*

Le chien de garde modélisé dans la figure 15 permet d'éviter une attente infinie de l'arrivée de l'événement e_k. La valeur qui lui associée correspond à la valeur maximale du délai.

Considérons que cette contrainte est associée à un superviseur S_1. Dans le cas général, si S_1 reçoit les événements e_k^p, p=2..n, des différents superviseurs Sp, sachant que les durées Δp qui se sont écoulées entre la réception de e_k^p par le superviseur S_1 et de e_p par S_p appartiennent à l'intervalle $[\delta_{mp}, \delta_{Mp}]$, alors nous obtenons la modélisation suivante :

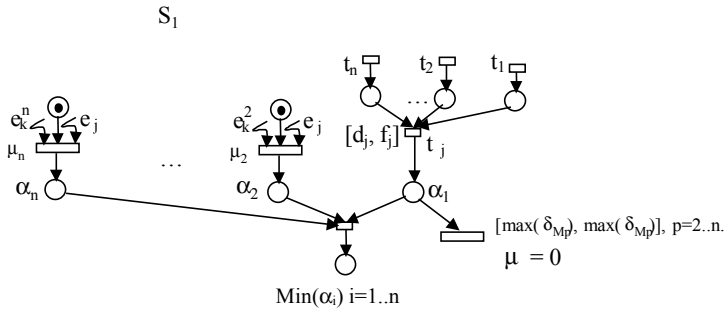


Figure 16. *Principe de modélisation de la contrainte D_j, cas général.*

Ainsi la vérification d'une contrainte de type fenêtre d'admissibilité doit être faite dans le cas général de la prise en compte des occurrences des événements locaux et celles des événements e_k^p reçus des superviseurs S_p.

Traitons enfin le dernier type de contrainte qu'est la contrainte de précédence.

IV.5.3. Modélisation d'une contrainte de précédence avec prise en compte des délais

La figure 17 modélise une contrainte de précédence avec un réseau de Petri p-t-temporel flou. Le modèle est identique à celui d'une contrainte de type intervalle, seule la valeur de μ diffère. Ceci est dû à ce que la contrainte de précédence peut être considérée comme un cas particulier d'une contrainte de type intervalle en prenant $d_{j,i} = 0$ et $f_{j,i} = +\infty$.

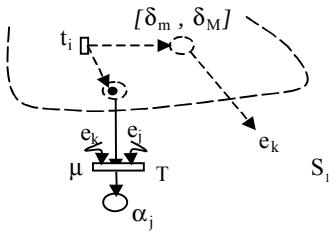


Figure 17. *Principe de modélisation de la contrainte de précédence.*

Après avoir modélisé les contraintes temporelles et les mécanismes susceptibles de les vérifier, nous nous intéressons maintenant à la modélisation et à la reconnaissance de sous-chroniques.

VI.5.4. Modélisation et reconnaissance de sous-chronique

Comme une sous-chronique est composée d'un ensemble d'événements, d'un ensemble de contraintes locales et d'un ensemble de contraintes globales, la vérification des contraintes la constituant permet de conclure sur la reconnaissance de la sous-chronique. Ainsi une sous-chronique peut être vue comme la conjonction d'un modèle représentant les contraintes locales et d'un modèle représentant l'ensemble des contraintes globales prises séparément, (figure 18). Cette séparation est due au fait que la vérification du modèle de contraintes locales et celle du modèle de contraintes globales peuvent se faire en parallèle.

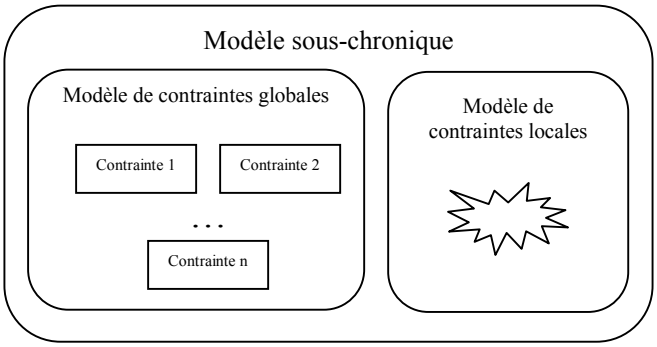


Figure 18. *Modèle de sous-chronique.*

Nous avons montré qu'en utilisant une approche de modélisation par réseaux de Petri, le modèle de contraintes locales pourrait être représenté grâce aux réseaux de Petri p-t-temporel. Mais, la vérification du modèle de contraintes locales ne suffit pas à elle seule pour conclure sur la reconnaissance de la sous-chronique. Il faut en effet que le modèle de contraintes globales soit lui aussi vérifié. Le modèle de contraintes globales doit assurer la communication et la coopération entre différents superviseurs supportant ces modèles. Ceci est, comme nous l'avons dit, assuré grâce à des places de communication d'entrée et de sortie. D'un côté, la communication permet de collecter l'information nécessaire à la vérification d'une contrainte et de l'autre côté, la coopération permet l'affinement de la vérification, cf § III.4.6. Ces deux étapes sont donc nécessaires afin de se prononcer sur la possibilité de vérification d'une contrainte globale.

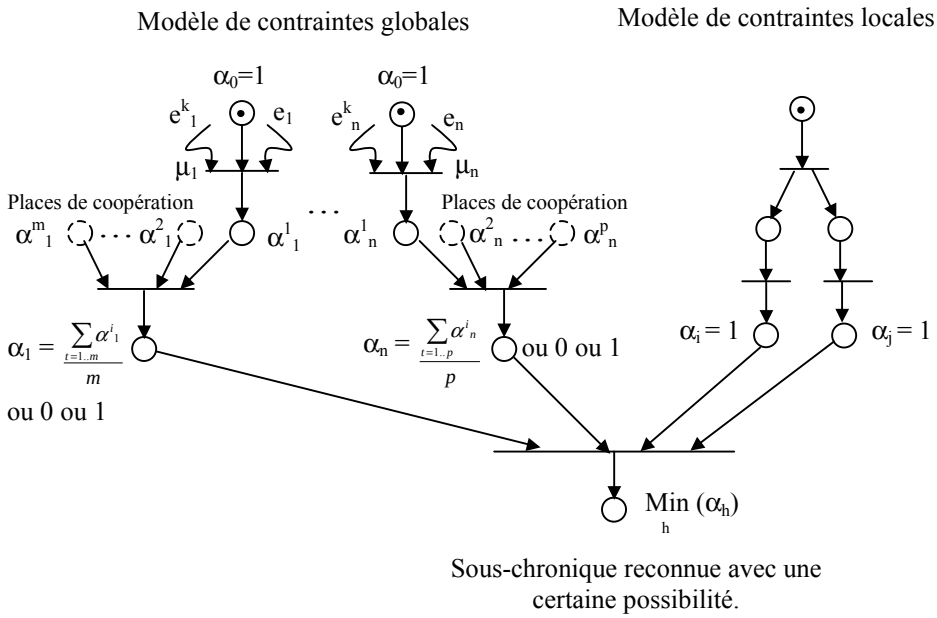


Figure 19. *Reconnaissance de sous-chronique.*

La conjonction des résultats de vérification de contraintes obtenus par un modèle et par l'autre permet de conclure sur la reconnaissance de la sous-chronique, (figure 19). Cette figure donne le modèle réseau de Petri flou d'une sous-chronique composée de deux contraintes locales et de deux contraintes globales. La possibilité de vérification d'une contrainte globale est donnée en fonction du résultat de vérification de la contrainte par le superviseur local ainsi que des résultats de la coopération. La possibilité de reconnaissance de la sous-chronique est le minimum des possibilités de vérification des deux modèles de contraintes. Signalons enfin que si une contrainte est vérifiée avec une possibilité nulle alors le processus de vérification s'arrête et la sous-chronique est non reconnue.

Afin de mettre en pratique les mécanismes de reconnaissance et de modélisation présentés précédemment, nous présentons, dans la suite, un exemple d'application aux systèmes de transport dans le cas d'un terminal intermodal de fret.

IV.6. Application aux systèmes de transport

Pour illustrer en détail le rôle de la fonction détection dans le cas d'opérations successives nécessitant l'utilisation des mêmes ressources, nous considérons un exemple qui traite de la gestion de trois conteneurs C_1 et C_2 et C_3. Cette exemple sera complété dans le chapitre V.
Les réseaux de Petri représentant des portions des cycles opératoires des trois conteneurs sont montrés à la figure 20. Les transitions représentées par des barres épaisses sont les points de synchronisation. Pour des raisons de simplification, nous considérons deux ressources qui sont l'« engin de stockage » et la « remorque », comme ressources nécessaires à la réalisation des cycles opératoires des trois conteneurs (Degano et al 2002).

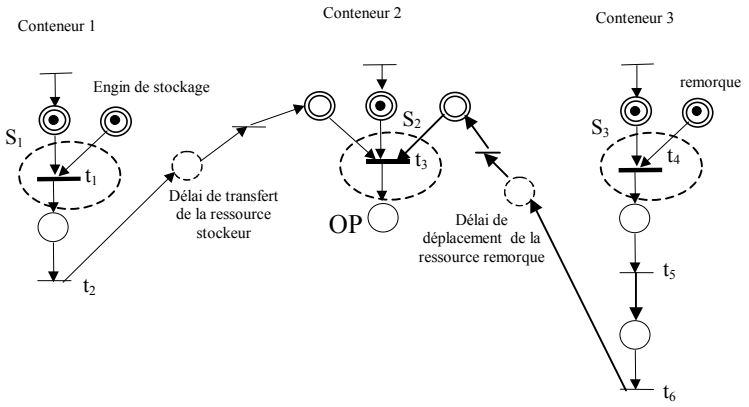


Figure 20. *Modélisation de portions des cycles opératoires des trois conteneurs.*

La surveillance d'un tel système est réalisée uniquement en observant les événements e_1, e_3 et e_4 dont l'occurrence est à l'origine du franchissement des transitions de synchronisation si toutes les ressources nécessaires sont disponibles (Degano et al 2002). Pour cela, nous associons les superviseurs S_1 à la réception de e_1, S_2 à la réception de e_3 et S_3 à la réception de e_4. La transmission des données entre ces superviseurs est assurée via un réseau de communication. Une transition t_i est associée à l'occurrence d'un événement e_i.
Nous supposons avoir les contraintes temporelles de fonctionnement suivantes :

$$3 \leq O(e_3) - O(e_2) \leq 8 \quad : C_{3\,2}$$
$$2 \leq O(e_3) - O(e_6) \leq 7 \quad : C_{3\,6}$$
$$1 \leq \min_{p}(O(e_3) - O(e_p)) \leq 3 \quad : D_3$$

Les contraintes de type intervalle expriment des durées opératoires ou des durées de transfert. La contrainte de type fenêtre d'admissibilité exprime un rendez-vous entre ressources nécessaires à l'exécution de l'opération OP, figure 20, et dont l'utilisation ne commencera qu'après une durée comprise entre 1 et 3 unités de temps. Cette dernière durée exprime un délai de configuration ou de préparation des ressources considérées.

Nous obtenons la chronique suivante :

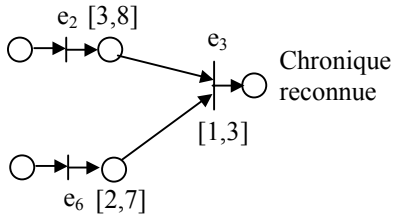


Figure 21. *Modélisation de la chronique.*

La tâche des trois superviseurs est de surveiller les contraintes D_3, C_{32} et C_{36} à partir des durées opératoires connues $\Delta_1 = O(e_2)-O(e_1) \in [1, 2]$ et $\Delta_2 = (O(e_6)-O(e_5))+(O(e_5)-O(e_4)) \in [1+2, 2+3]$ et de la réception des événements e_1, e_3 et e_4. Les autres opérations sont supposées se dérouler normalement. Ceci correspond à surveiller des délais de transfert des ressources au conteneur 2 et le début de leur utilisation. Pour réaliser cela, le superviseur S_2 se base sur le modèle sous-chronique suivant, comme il a été défini au § IV.5.4 :

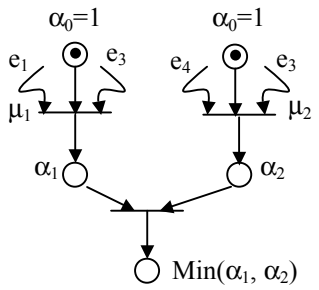
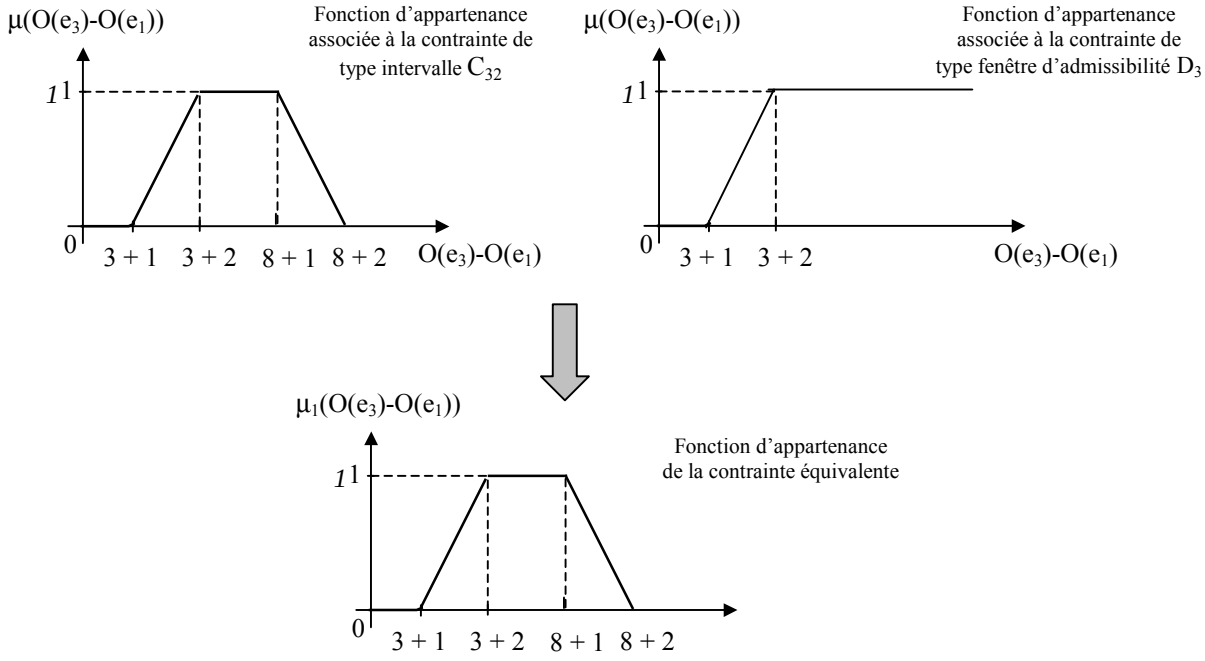


Figure 22. *Modélisation de la sous-chronique.*

Les événements e_3 et e_2 sont liés par deux contraintes temporelles (de type intervalle et de type fenêtre d'admissibilité). La fonction d'appartenance de la contrainte équivalente, que nous cherchons pas à retrouver ici, entre ces deux événements est $\mu_{1\ et}$ représente le minimum entre la fonction d'appartenance associée à la contrainte de type intervalle C_{32} et celle associée à la contrainte de type fenêtre d'admissibilité D_3 entre les deux événements. Le raisonnement est similaire pour μ_2 concernant les événements e_3 et e_6.

En utilisant le calcul développé au paragraphe III.4.2., nous obtenons les résultats suivants :

- $e_k = e_1$, $e_i = e_2$, $e_j = e_3$,



- De même pour $e_k = e_4$, $e_i = e_6$, $e_j = e_3$, nous obtenons la fonction d'appartenance de la contrainte équivalente suivante :

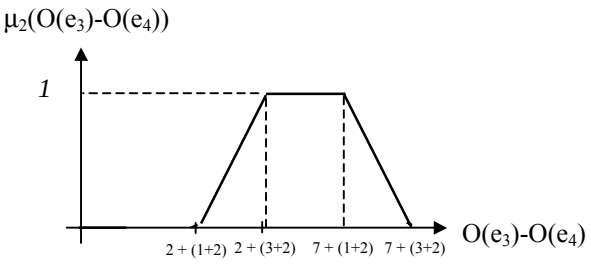


Figure 20. *Les deux fonctions d'appartenance.*

Prenant par exemple le cas de fonctionnement suivant en supposant que :

$$O(e_3) - O(e_1) = 6 \text{ et } O(e_3) - O(e_4) = 6$$

Le superviseur 2 détecte que l'opération *OP* nécessitant les ressources engin de stockage et remorque a bien commencé et que les possibilités que les opérations de transfert ayant déjà été effectuées se soient bien passées sont respectivement de 1 (μ_1) pour le transfert de l'engin de stockage et de 0.5 (μ_2) pour le déplacement de la remorque. Le superviseur 2 détecte qu'un mauvais fonctionnement a pu avoir lieu lors de l'opération de déplacement de la remorque. Ceci peut être dû à une défaillance interne de la remorque ou à la réalisation d'une opération de maintenance non prévue ou de durée non déterminée. L'opération *OP* à entreprendre peut bien commencer mais avec une possibilité égale à 0.5 = min (1, 0.5) qu'elle se passe bien à cause de la défaillance probable de la remorque.

Un affinement des détection-diagnostic pourrait être effectué par S_2 en consultant d'autres superviseurs surveillant d'autres opérations de la remorque et conclure ainsi de manière coopérative sur l'apparition ou non d'un symptôme de défaillance lié à la remorque.

IV.7. Conclusion

Nous avons présenté dans ce chapitre la modélisation par réseaux de Petri p-t-temporels de chronique et de sous-chroniques. Les aspects statiques et dynamiques de la modélisation de chronique ont été précisés. Le cas le plus général concernant l'occurrence multiple d'un événement a également été traité par prise en compte de la multi-sensibilisation dans l'évolution du réseau de Petri ou par duplication du réseau de Petri.
Nous avons aussi présenté différentes méthodes pour obtenir des modèles réseau de Petri représentant des sous-chroniques à partir d'un modèle réseau de Petri représentant une chronique, ou à partir de modèles réseau de Petri représentant les contraintes temporelles composant une sous-chronique. Ceci a été réalisé dans le cas de la non prise en compte des délais de communication ou de transport entre superviseurs.
Lorsque les délais de communication ou de transport entre superviseurs existent, nous avons défini les réseaux de Petri p-t-temporels flous comme outils de modélisation de sous-chroniques distribuées.
Le modèle général de modélisation de sous-chroniques ainsi que les mécanismes permettant de les reconnaître a ainsi été défini et peut être utilisé comme modèle général pour des réseaux de Petri communicants dans le cas où des délais existent ou pas.

La suite logique de notre travail est d'illustrer l'ensemble de nos résultats sur un exemple réel. C'est l'objet du chapitre V.

Chapitre V

Application à la surveillance d'un terminal intermodal

V.1. Introduction

Ces dernières années, le transport a eu un rôle important pour les logistiques internes et externes dans une chaîne de production. Particulièrement dans les logistiques externes, les terminaux et les points de transfert sont des composants fondamentaux. De tels composants sont souvent considérés comme des systèmes de production à part entière, où les produits « virtuels » sont les services fournis. Dans les terminaux intermodaux, les facteurs qui affectent le plus les performances du système sont les retards liés à l'arrivée d'un conteneur et les défaillances liées aux ressources. Un comportement anormal résulte de n'importe quelle déviation du plan opératoire prévu. Ainsi, le besoin de respecter l'ordonnancement des différentes opérations nécessite une surveillance continue, telle que la détection afin de prévenir le plus tôt possible de l'apparition d'événements imprévisibles.
Dans ce chapitre, nous traitons le cas d'un terminal de fret intermodal considéré comme une installation de transport particulière dans lequel seuls des conteneurs sont manipulés. Comme d'autres installations de transport, un terminal intermodal peut être représenté comme un système à événements discrets (SED) (DeWitt et al 2000), permettant ainsi la représentation des stratégies de commande. Le modèle proposé par (Degano et al 2002) utilise les réseaux de Petri et exploite leurs possibilités à représenter les besoins de synchronisation dans l'utilisation concurrente de ressources partagées pour la gestion des conteneurs.
L'approche de la détection distribuée que nous avons développée trouve son application dans ce type de système. En effet, nous proposons de surveiller les opérations de transfert de ressources entre conteneur moyennant, quand c'est nécessaire, la connaissance de durées relatives aux opérations à effectuer sur les conteneurs et à travers la communication et la coopération entre superviseurs associés aux différents conteneurs.

La première partie de ce chapitre est une présentation de quelques définitions relevant du domaine auquel appartient l'exemple. Le terminal intermodal étudié y est décrit ainsi que les différents cycles opératoires que doivent suivre les conteneurs. Dans la deuxième partie de ce chapitre, nous nous intéressons à la modélisation des différents cycles opératoire par réseau Petri ainsi qu'à la

modélisation des différentes contraintes à surveiller et qui représentent la chronique à reconnaître. La troisième partie est consacrée à la description de la distribution des contraintes temporelles à surveiller et à la modélisation des sous-chroniques ainsi obtenues. La dernière partie traite de l'utilisation en-ligne des modèles des sous-chroniques pour la détection distribuée de symptôme de défaillance. Ceci est effectué grâce à la communication et la coopération des différents superviseurs.

V.2. Définitions

Ce paragraphe présente un ensemble de définitions nécessaires à la compréhension de ce chapitre. Ces définitions concernent des composants d'un terminal intermodal et les opérations qui peuvent y avoir lieu.

Un *conteneur* est un coffre rigide destiné à contenir de la marchandise, (figure 1).

Conteneurs-citernes

Arrimage de conteneurs

Conteneur réfrigéré

Figure 1. *Les conteneurs*.

L'unité de chargement est un conteneur ou une caisse mobile.

L'unité de Transport Intermodale (UTI) représente des conteneurs, des caisses mobiles et des semi-remorques convenant au transport intermodal.

L'empotage/dépotage est une opération de chargement ou de déchargement de marchandises à l'intérieur d'une UTI.

Le transport intermodal est l'acheminement d'une marchandise utilisant deux modes de transport ou plus mais dans la même unité de chargement ou le même véhicule routier, et sans empotage ni dépotage.
Par extension, l'*intermodalité* , (figure 2), caractérise un système de transport en vertu duquel deux modes de transport ou plus sont utilisés par la même unité de chargement ou le même véhicule routier, sans empotage ou dépotage.

Figure 2. *L'intermodalité : chargement d'un conteneur sur un camion.*

Le *transbordement* est le mouvement des UTI d'un moyen de transport à un autre.

La *consolidation (groupage)* est l'action consistant à réunir les envois de marchandises en provenance de plusieurs expéditeurs.

La *déconsolidation (dégroupage)* est l'action de disperser des marchandises à l'arrivée vers différents destinataires.

Le *terminal*, (figure 3), est un lieu équipé pour le transbordement et le stockage des UTI.

Figure 3. *Le terminal Intermodal de Milwaukee.*

Nous présentons dans le paragraphe suivant une brève description du terminal intermodal ainsi que des opérations qui s'y déroulent.

V.3. Description du terminal intermodal

Un terminal intermodal est considéré comme une installation permettant la manipulation de conteneurs depuis leur arrivée jusqu'à leur départ du terminal. Il est subdivisé en zones, chacune associée à une opération de manipulation particulière du conteneur (Degano et al 2002). Chaque zone, c'est à dire, le *quai*, *l'aire de stockage*, la *gare de triage*, et les *portails* d'entrée/sortie, (figure 4), est équipée par son système de matériel de manipulation formé d'un ensemble de ressources non déplaçables. Plus spécifiquement, le quai est équipé par un ensemble de ressources appelées *portiques*, capables de charger et de décharger les conteneurs du/au navire. L'aire de stockage est utilisée pour les opérations de consolidation et de déconsolidation des conteneurs. La gare de triage est équipée de ressources dédiées appelées *grues*. Les portails d'entrée/sortie sont considérés comme des ressources. Ils sont en fait de trois types : les portails de sortie, les portails d'entrée et les portails d'entrée/sortie.

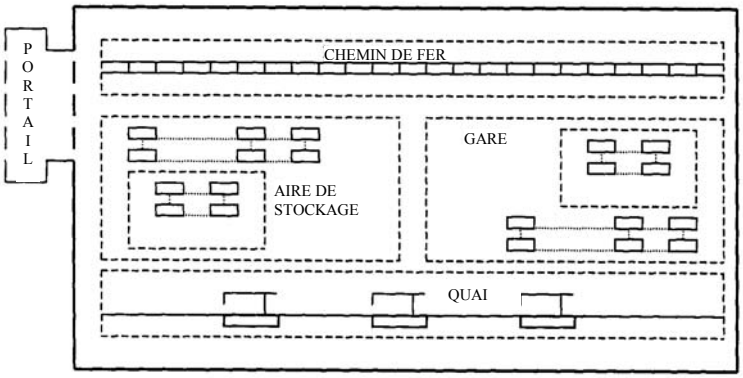


Figure 4. *Organisation d'un terminal intermodal.*

Le système de manipulation de matériel du terminal est formé par deux composants : les ressources non déplaçables, dédiées à des zones uniques, et les véhicules de manipulation de matériel qui peuvent être *privés* ou *partagés*. Un véhicule privé est dédié au déplacement de marchandise dans une zone unique (quai, gare de triage, aire de stockage). Un véhicule partagé sert à transférer les conteneurs d'une zone à une autre. Pour cela, les zones du terminal sont liées par un réseau de transport consistant en des routes bi-directionnelles.

Un terminal à conteneurs supporte trois types de modes de transport : mer, rail, route. Les conteneurs sont manipulés dans le terminal selon différents cycles opératoires distingués en trois typologies : export, import, et transbordement. Chaque cycle opératoire est caractérisé par le mode de transport utilisé pour permettre au fret d'arriver ou de partir à/du terminal, c'est à dire, mer, rail, ou route, en plus d'opérations de consolidation/déconsolidation que doit subir un conteneur dans l'aire de stockage.

Neuf cycles opératoires sont représentés sur la figure 5 et se composent de quatre cycles d'import I_i, de quatre cycle d'export E_i et d'un cycle de transbordement T. Un conteneur arrivant au terminal doit suivre l'un de ces cycles.

Cycles opératoire	Transporteur arrivant	Transporteur partant	consolidation
E1	Camion	Navire	non
E2	Camion	Navire	oui
E3	Train	Navire	non
E4	Train	Navire	oui
I1	Navire	Camion	non
I2	Navire	Camion	oui
I3	Navire	Train	non
I4	Navire	Train	oui
T	Navire	Navire	non

Figure 5. *Les cycles opératoires.*

Chaque cycle opératoire consiste en une séquence d'opérations qui doivent être exécutées respectant les contraintes temporelles de bon fonctionnement. Pour cela, tout conteneur à manipuler

doit être associé de manière exclusive à une ou plusieurs ressources pour chaque tâche appartenant au cycle opératoire qui lui est associé.
Dans le modèle proposé par (Degano et al 2002), tous les mouvements de matériel dans le terminal sont planifiés à priori sur un horizon d'un jour. En effet, il est supposé connu un jour à l'avance, le nombre et la typologie des conteneurs arrivants, leurs dates limites d'embarcation et leurs origine et destination dans le terminal. En trouvant les séquences d'opérations pour chaque ressource et en leur associant des données temporelles, le Plan du Processus Journalier (PPJ) est généré (Degano et al 2002). Le PPJ se compose d'un ensemble d'ordres (les cycles opératoires), chacun d'eux consiste en un ensemble de tâches à accomplir dans des fenêtres temporelles prédéfinies.

V.4. Modélisation de l'installation de transport par réseau de Petri

Le réseau de Petri représentant le système considéré peut être divisé en plusieurs sous-réseaux associés, représentant chacun le cycle opératoire d'un conteneur, et qui peuvent être vus comme des installations à part entière interagissant à travers des *transitions de synchronisation* (Degano et al 2002). Ces transitions représentent le début d'une opération sur un conteneur nécessitant une ressource ou plus. Ceci permet au système de surveillance de détecter des déviations de l'ordonnancement prévu des opérations en observant seulement les transitions de synchronisation.

L'exemple proposé est extrait de (Degano et al 2002) et traite de la manipulation de cinq conteneurs, C_1, C_2, C_3, C_4, C_5, appartenant à trois cycles opératoires différents d'une portion du Plan du Processus Journalier. Le réseau de Petri représentant la portion du PPJ est montré à la figure 7. Les cinq conteneurs sont caractérisés comme le montre la figure 6.

Conteneur	Cycle opératoire	Mode de transport	transporteur
C_1	E3	Navire	Carthage
C_2	E1	Navire	Carthage
C_3	E1	Navire	Carthage
C_4	I3	Train	SNCFT01
C_5	I4	Train	SNCFT01

Figure 6. *Portion du PPJ.*

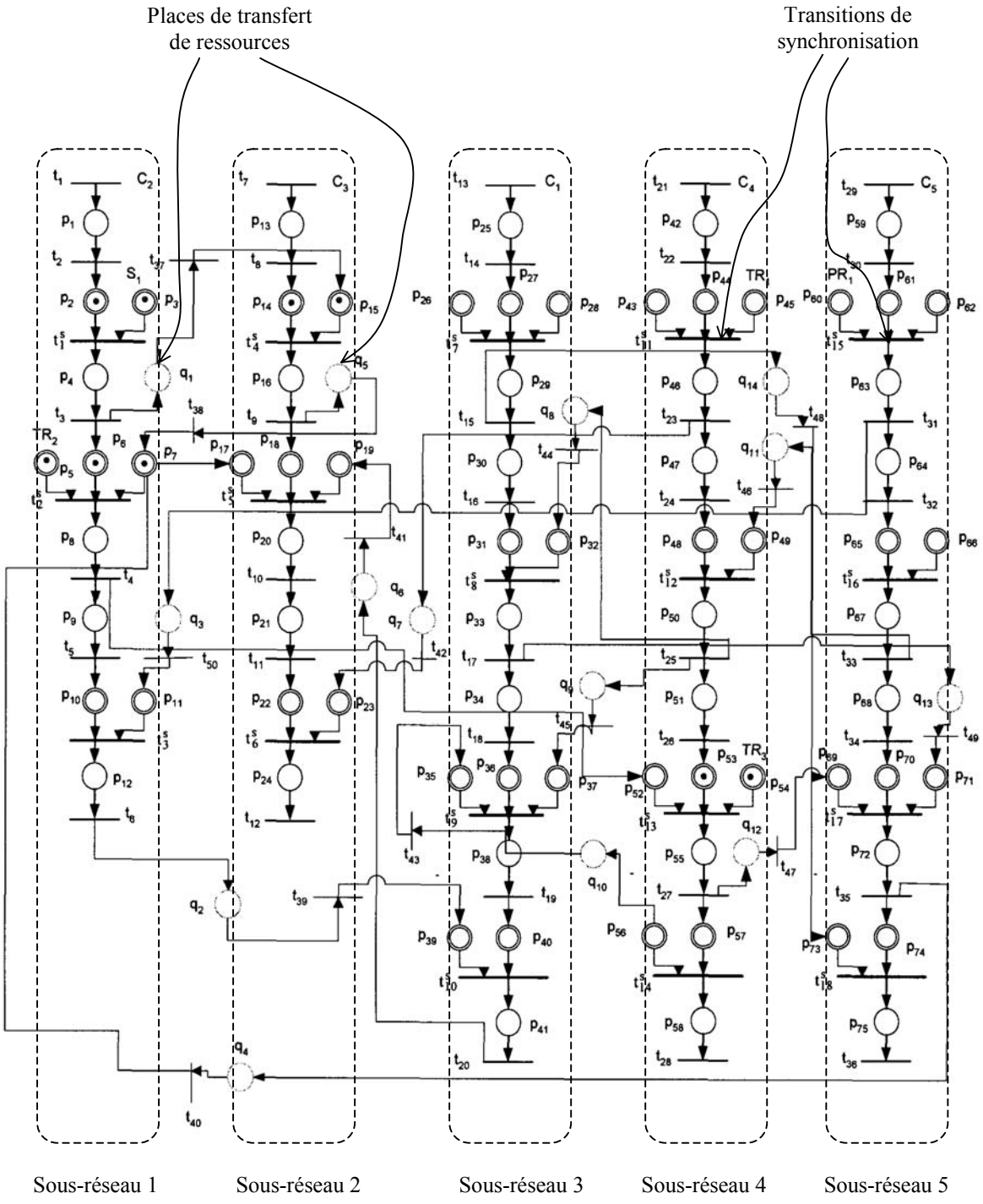


Figure . *Le modèle réseau de Petri et les sous-réseaux.*

Les transitions de synchronisation sont notées par t_i^s, i=1,…18. Les places représentées par des doubles cercles sont des conditions qui permettent aux opérations de commencer lorsqu'elles sont satisfaites. La présence de jetons dans ces places reflète la disponibilité des ressources nécessaires. De plus, les places notées q_h, h=1,…, 14, représentent le délai de transfert des ressources entre deux points de synchronisation, c'est à dire généralement entre deux conteneurs.
Le réseau de Petri obtenu peut être considéré comme composé de cinq sous-réseaux chacun relatif à un conteneur et à son cycle opératoire, (figure 7). Les ressources impliquées dans la portion du PPJ sont : un portique notée PR_1, un engin de stockage des matériaux (gerbeur) noté S_1, trois remorques notées TR_1, TR_2 et TR_3 respectivement. Considérons le portique PR_1, les transitions t^s_{15}, t_{31}, t_{50}, t^s_3, t_6, t_{39}, t^s_{10} modélisent la séquence opératoire planifiée du portique. Ainsi le portique est réservé à la transition t^s_{15} afin de décharger le conteneur C_5 du navire, à la transition t^s_3 pour charger le conteneur C_2 sur le navire et à la transition t^s_{10} pour charger le conteneur C_1 sur le navire. Les transitions t^s_1, t_3, t_{37}, t^s_4, t_9, t_{38}, t^s_2, t_4, t_{51}, t^s_{13}, t_{27}, t_{47}, t^s_{17} représentent la séquence opératoire planifiée de l'engin de stockage de matériaux. Ainsi S_1 est réservé à la transition t^s_1 afin de décharger C_2 du camion, à la transition t^s_4 pour décharger C_3 du camion, à la transition t^s_2 pour charger C_2 sur la remorque TR_2, à la transition t^s_{13} pour charger C_4 sur la remorque TR_3, et à la transition t^s_{17} afin de charger C_5 sur la remorque TR_1. Les séquences opératoires planifiées des autres ressources (remorques) dans le réseau de Petri sont modélisées de manière similaire.
Cette description nous permet de dégager l'ensemble Σ des événements, liés aux opérations à effectuer, que nous allons considérer dans la suite. Ces événements sont décrits comme suit :

e^s_{15} : début du déchargement du conteneur C_5 du navire,
e_{31} : fin du déchargement du conteneur C_5 du navire,
e_{50} : fin du transfert du portique au conteneur C_2,
e^s_3 : début du chargement du conteneur C_2 sur le navire,
e_6 : fin du chargement du conteneur C_2 sur le navire,
e_{39} : fin du transfert du portique au conteneur C_1,
e^s_{10} : début du chargement du conteneur C_1 sur le navire,
e^s_1 : début du déchargement du conteneur C_2 du camion,
e_3 : fin du déchargement du conteneur C_2 du camion,
e_{37} : fin du transfert de l'engin de stockage au conteneur C_4,
e^s_4 : début du déchargement du conteneur C_3 du camion,
e_9 : fin du déchargement du conteneur C_3 du camion,
e_{38} : fin du transfert de l'engin de stockage au conteneur C_2,
e^s_2 : début du chargement du conteneur C_2 sur la remorque TR_2,
e_4 : fin du chargement du conteneur C_2 sur la remorque TR_2,
e_{51} : fin du transfert de l'engin de stockage au conteneur C_4,
e^s_{13} : début du chargement du conteneur C_4 sur la remorque TR_3,
e_{27} : fin du chargement du conteneur C_4 sur la remorque TR_3,
e_{47} : fin du transfert de l'engin de stockage au conteneur C_5,
e^s_{17} : début du chargement du conteneur C_5 sur la remorque TR_1.
e^s_{16} : début de la consolidation du conteneur C_5.

Comme la surveillance, et plus particulièrement la détection, se base sur la datation des événements générés par le tir des transitions de synchronisation, l'ensemble des événements reçus du système surveillé est donné par les événements e^s_i, i=1..18. L'ensemble des événements non reçus par aucun des superviseurs est défini par $\Sigma \setminus \{e^s_i, i=1..18\}$. Les événements qui ne peuvent pas être reçus par les superviseurs sont des événements qui ne peuvent pas être générés par le système surveillé par manque de capteurs ou/et d'opérateurs dédiés.

A l'aide des événements de Σ, nous définissons les contraintes temporelles suivantes du plan opératoire à surveiller :

$$
\begin{aligned}
O(e_3) < O(e^s_2) & \\
2 \le O(e^s_3) - O(e_{31}) \le 7 & \quad : C_{s3\ 31} \\
3 \le O(e^s_{10}) - O(e_6) \le 8 & \quad : C_{s10\ 6} \\
1 \le O(e^s_1) - O(e^s_{10}) \le 4 & \quad : C_{s1\ s10} \\
2 \le O(e^s_{17}) - O(e^s_{15}) \le 6 & \quad : C_{s17\ s15} \\
1 \le O(e_{37}) - O(e^s_1) \le 2 & \quad : C_{37\ s1} \\
1 \le O(e^s_2) - O(e_9) \le 9 & \quad : C_{s2\ 9} \\
2 \le O(e^s_{17}) - O(e_{31}) \le 6 & \quad : C_{s17\ 31} \\
1 \le \min_p(O(e^s_2) - O(e_p)) \le 6 & \quad : D_{s2}
\end{aligned}
$$

Ces contraintes proviennent, comme nous l'avons dit, du plan opératoire journalier représentant des durées bornées de l'exécution des différentes opérations considérées. Le non respect d'une de ces contraintes implique un retard dans l'exécution du plan opératoire défini ou la non réalisation de l'opération dont la durée est représentée par la contrainte.
La modélisation de ces contraintes se basant sur les mécanismes présentés au chapitre IV donne la figure suivante :

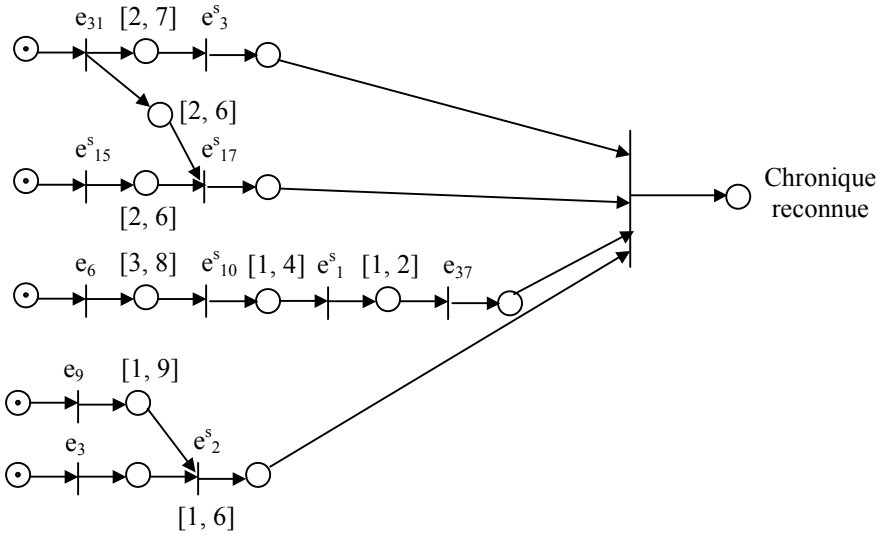


Figure 8. *Modèle RdP de la chronique associée au bon déroulement de l'exécution du plan opératoire.*

Afin d'utiliser les avantages d'une architecture de surveillance distribuée que nous avons présentée au chapitre I, nous procédons, dans le paragraphe suivant à la distribution des différentes contraintes temporelles à surveiller

V.5. Distribution des contraintes temporelles et modélisation des sous-chroniques

En voulant distribuer les contraintes temorelles ou le modèle de chronique les représentant, nous associons à chaque conteneur un superviseur S_i, i=1..5. Chaque superviseur surveille un sous-ensemble des contraintes représenté par une sous-chronique. Nous obtenons ainsi cinq sous-chroniques associées chacune à un superviseur. Lorsqu'un délai connu est nécessaire à la vérification d'une contrainte globale, l'incertitude sur le délai est du même ordre mais inférieure à la tolérance sur la contrainte. Ceci est assuré par le fait que la majorité des contraintes surveillées représentent des durées de transfert de ressources qui sont relativement supérieures aux délais opératoires pris en compte dans leur vérification. Cette donnée, comme nous l'avons dit au chapitre III, permet d'assurer un maximum de précision quant aux possibilités de vérification d'une contrainte globale. Rappelons qu'une contrainte est vérifiée avec précision si elle est satisfaite ou non satisfaite. Nous détaillons par la suite l'ensemble des cinq sous-chroniques obtenues.

Sous-chronique 1 associée au superviseur S_1

L'ensemble des événements pouvant être reçus par ce superviseur en provenance du système est donné par

$$\Sigma_1 = \{e^s_7, e^s_8, e^s_9, e^s_{10}\}.$$

La contrainte à vérifier par S_1 est $C_{s10\,6}$ donnée par :

$$3 \leq O(e^s_{10}) - O(e_6) \leq 8 \quad : C_{s10\,6}$$

Comme le superviseur S_1 ne peut pas dater l'occurrence de l'événement e_6, la contrainte $C_{s10\,6}$ est une contrainte globale. En effet, l'événement e_6 n'est reçu par aucun des superviseurs. Nous considérons l'événement intermédiaire e^s_3, et le délai connu $O(e_6) - O(e^s_3) \in [2, 3]$ obtenu du plan opératoire. Nous obtenons la modélisation de la sous-chronique suivante :

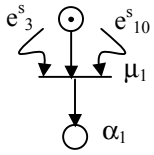


Figure 9. *Modèle de la sous-chronique 1.*

A partir des résultats obtenus dans le § III.4.1., nous obtenons la figure suivante représentant la possibilité de vérifier la contrainte $C_{s10\,6}$:

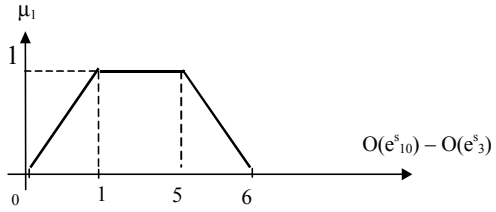


Figure 10. *Fonction d'appartenance μ_1 représentant les possibilités de vérifier la contrainte $C_{s10\,6}$.*

Considérons maintenant la sous-chronique 2 associée au superviseur S_2.

Sous-chronique 2 associée au superviseur S_2

L'ensemble des événements pouvant être reçus en provenance du système par ce superviseur est donné par

$$\Sigma_2 = \{e^s_1, e^s_2, e^s_3\}.$$

Les contraintes à vérifier par S_2 sont les suivantes :

$$\begin{aligned} O(e_3) < O(e^s_2) & \\ 2 \le O(e^s_3) - O(e_{31}) \le 7 & \quad : C_{s3\ 31} \\ 1 \le O(e^s_1) - O(e^s_{10}) \le 4 & \quad : C_{s1\ s10} \\ 1 \le O(e_{37}) - O(e^s_1) \le 2 & \quad : C_{37\ s1} \\ 1 \le O(e^s_2) - O(e_9) \le 9 & \quad : C_{s2\ 9} \\ 1 \le \min_p(O(e^s_2) - O(e_p)) \le 6 & \quad : D_{s2} \end{aligned}$$

Comme le superviseur S_2 ne peut pas dater les occurrences des événements e_3, e_{31}, e_{37}, e^s_{10}, e_9, toutes les contraintes incluant ces événements sont des contraintes globales. Les événements e_3, e_{31}, e_{37} et e_9 ne peuvent pas être reçus par les superviseurs et l'événements e^s_{10} est peut être reçu par le superviseur S_1 en provenance du système surveillé. Pour que S_2 puisse vérifier la contrainte $C_{s1\ s10,}$ il doit nécessairement dater l'occurrence de l'événement e^s_{10}. Comme le délai de communication entre les superviseurs est négligeable vis à vis des durées surveillées, cette contrainte sera modélisée moyennant les mécanismes présentés au § IV.4.2. Pour les autres contraintes, nous considérons respectivement les événements intermédiaires e^s_4, e^s_{15} et e^s_{10} , cf § III.3.3., sachant des délais connus $O(e^s_4) - O(e_3) \in [1, 2]$, $O(e_{31}) - O(e^s_{15}) \in [1, 2]$, $O(e_{37}) - O(e^s_{10}) \in [2, 3]$, $O(e_9) - O(e^s_4) \in [1, 3]$. Nous obtenons la modélisation suivante de la sous-chronique :

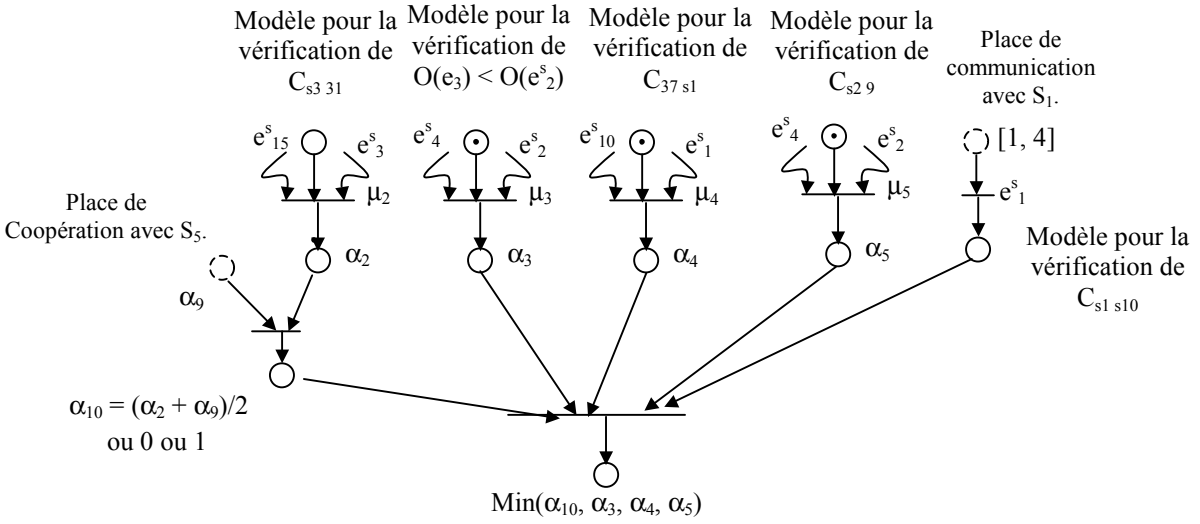


Figure 11. *Modèle de la sous-chronique 2.*

Signalons que la place à laquelle est associée l'étiquette α_9 est une place de coopération avec le superviseur S_5. Cette coopération est possible car les deux superviseurs possèdent une contrainte globale chacun ayant un événement en commun qui est e_{31}, cf § III.4.6.
Les événements e_3 et e_{s2} sont liés par deux contraintes temporelles (de précédence $O(e_3) < O(e^s_2)$ et de type fenêtre d'admissibilité D_{s2}). μ_3 est la fonction d'appartenance associée à la contrainte

équivalente, cf § IV.6. Le raisonnement est similaire pour μ_5 relative aux événements e_9 et e^s_2 qui sont liés par deux contraintes, une de type de intervalle $C_{s2\ 9}$ et une de type fenêtre d'admissibilité D^s_2. A partir des résultats obtenus dans le § III.4.1., nous obtenons la figure suivante représentant les possibilités de vérifier les contraintes globales sauf dans le cas de la contrainte $C_{s1\ s10}$ dont la possibilité prend les valeurs discrètes 0 ou 1 :

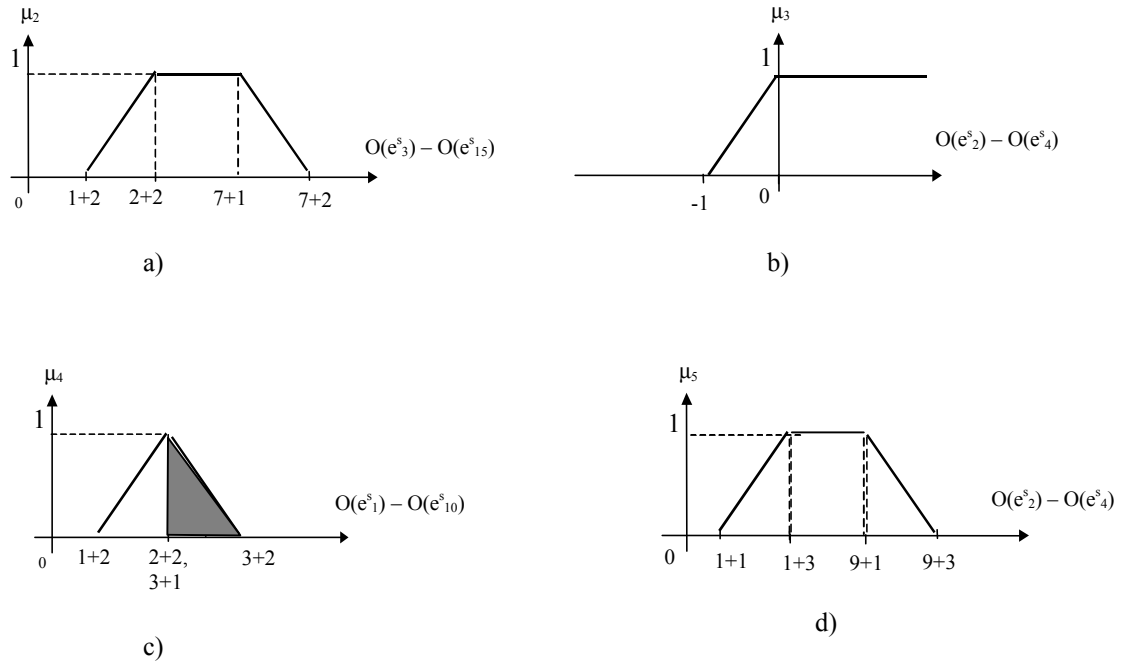


Figure 12. *Fonctions d'appartenance représentant les possibilités de vérifier les contraintes.*

La partie foncée dans la figure 12, c) représente une zone interdite puisque la durée $O(e^s_1) - O(e^s_{10}) \in [1, 4]$, de par la contrainte $C_{s1\ s10}$. Aussi, la durée $O(e^s_2) - O(e^s_4)$ doit être toujours positive puisque l'occurrence de l'événement e^s_2 suit forcément l'occurrence de l'événement e^s_4 dans le cadre de notre étude. Ceci est assuré par le mouvement de l'engin de stockage dans le terminal. Ainsi à a partir de la figure 12, b), nous déduisons que la contrainte $O(e_3) < O(e^s_2)$ est toujours vérifiée et que l'occurrence de e_3 ne peut pas être à l'origine de la violation de la contrainte de type fenêtre d'admissibilité D^s_2.

Nous considérons maintenant la sous-chronique 3 associée au superviseur S_3.

Sous-chronique 3 associée au superviseur S_3

L'ensemble des événements pouvant être reçus du système par ce superviseur est donné par $\Sigma_3 = \{e^s_4, e^s_5, e^s_6\}$. La distribution des contraintes temporelles ou de la chronique n'associe aucune contrainte à vérifier au superviseur S_3. Nous considérons, par la suite, la sous-chronique 4 associée au superviseur S_4.

Sous-chronique 4 associée au superviseur S_4

L'ensemble des événements pouvant être reçus du système par ce superviseur est donné par $\Sigma_4 = \{e^s_{11}, e^s_{12}, e^s_{13}, e^s_{14}\}$. De même que pour le superviseur S_3, la distribution des contraintes

temporelles ou de la chronique n'associe aucune contrainte à vérifier au superviseur S_4. Nous considérons, par la suite, la sous-chronique 5 associée au superviseur S_5.

Sous-chronique 5 associée au superviseur S_5

L'ensemble des événements pouvant être reçus par ce superviseur en provenance du système surveillé est donné par $\Sigma_5 = \{e^s_{15}, e^s_{16}, e^s_{17}, e^s_{18}\}$.
Les contraintes à vérifier par S_5 sont les suivantes :

$$2 \leq O(e^s_{17}) - O(e^s_{15}) \leq 6 \ : C_{s17\ s15}$$
$$2 \leq O(e^s_{17}) - O(e_{31}) \leq 6 \ : C_{s17\ 31}$$

Comme l'événement e_{31} ne peut pas être reçu par les superviseurs, nous considérons l'événement intermédiaire e^s_{16} sachant un délai connu $O(e^s_{16}) - O(e_{31}) \in [1, 2]$ obtenu à partir du plan opératoire journalier. Nous avons la modélisation suivante de la sous-chronique :

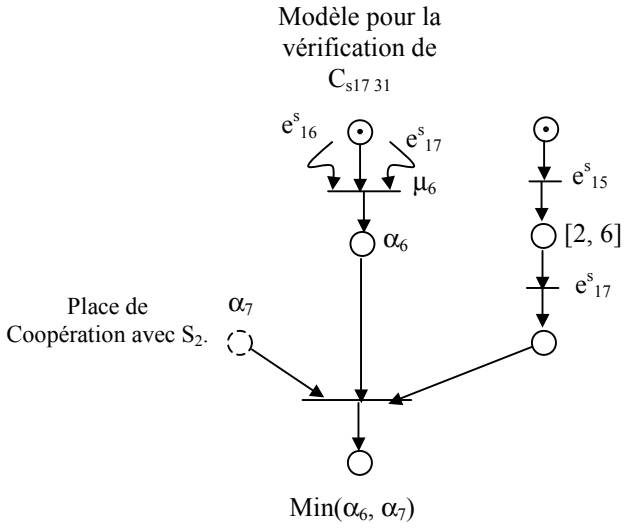


Figure 13. *Modèle de la sous-chronique 5.*

La place à laquelle est associée l'étiquette α_7 est une place de coopération avec le superviseur S_2. Cette coopération est possible car les deux superviseurs possèdent une contrainte globale chacun ayant un événement en commun qui est e_{31}, cf § III.4.6.
A partir des résultats obtenus dans le § III.4.1., nous obtenons la figure suivante représentant la possibilité de vérifier la contrainte $C_{s17\ 31}$

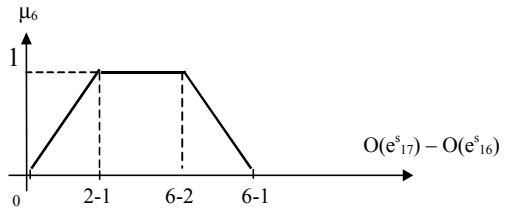


Figure 14. *Fonction d'appartenance μ_6 représentant les possibilités de vérifier la contrainte $C_{s17\ 31}$.*

Après avoir défini hors-ligne tous les aspects relevant des mécanismes de la détection distribuée, nous développons maintenant le déroulement des aspects en-ligne de la détection distribuée.

V.6. Reconnaissance en ligne des sous-chroniques

Nous nous intéressons aux mécanismes de reconnaissance des différentes sous-chroniques. Pour cela, comme il a été défini au chapitre III, nous associons aux superviseurs leurs listes de renvoi et de coopération successives, (figure 15).

	S_1	S_2	S_3	S_4	S_5
L_r	$(e^s_{10}, \{S_2\})$	$(e^s_3, \{S_1\})$	$(e^s_4, \{S_2\})$	$\varnothing$	$(e^s_{15}, \{S_2\})$
L_c	$\varnothing$	$(e^s_3, \{S_5\})$	$\varnothing$	$\varnothing$	$(e^s_{17}, \{S_2\})$

Figure 15. *Information relative aux listes de renvoi et de coopération entre superviseurs.*

Dès qu'un superviseur reçoit un événement du terminal, il consulte sa liste de renvoi L_r et communique aussitôt l'occurrence de cet événement, lorsque ceci est nécessaire, aux superviseurs qui en ont besoin pour vérifier leurs propres contraintes, (figure 16).

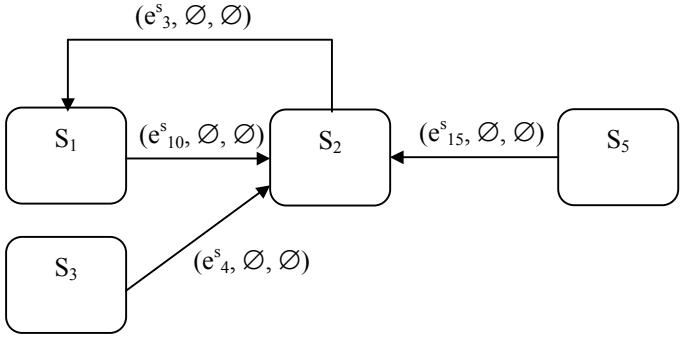


Figure 16. *Communication des événements à partir de la liste de renvoi.*

Quand le superviseur S_2 reçoit l'occurrence des événements e^s_{10} (début du chargement du conteneur C_1 sur le navire), e^s_{15} (début du déchargement du conteneur C_5 du navire) et e^s_4 (début du déchargement du conteneur C_3 du camion), il date ces événements et calcule les durées $O(e^s_2)$ - $O(e^s_4)$, $O(e^s_3)$ - $O(e^s_{15})$ et $O(e^s_1)$ - $O(e^s_{10})$. Signalons qu'à chaque réception d'un message portant l'occurrence d'un de ces événements, le modèle de la sous-chronique 2 évolue en franchissant la transition correspondante.
Supposons qu'après datation des événements e^s_1 (début du déchargement du conteneur C_2 du camion), e^s_2 (début du chargement du conteneur C_2 sur la remorque TR2) et e^s_3 (début du chargement du conteneur C_2 sur le navire), les durées calculées par S_2 ont les valeurs suivantes :

$$O(e^s_2) - O(e^s_4) = 5,\ O(e^s_3) - O(e^s_{15}) = 6,\ O(e^s_1) - O(e^s_{10}) = 4.$$

L'évolution du modèle réseau de Petri de la sous-chronique 2 permet de conclure que cette sous-chronique a été reconnue avec une possibilité égale à 1. Les opérations surveillées par cette sous-chronique se sont bien déroulées et le plan opératoire du conteneur 2 n'est pas retardé.

Considérons maintenant le scénario dans lequel :

$O(e^s_2) - O(e^s_4) = 5$, $O(e^s_3) - O(e^s_{15}) = \mathbf{3.5}$, $O(e^s_1) - O(e^s_{10}) = 4$.

La contrainte $C_{s3\ 31}$ a été vérifiée avec une possibilité égale à 0.5. Le superviseur S_2 peut décider de lancer une coopération avec le superviseur S_5 en lui envoyant un message de coopération, (figure 17).

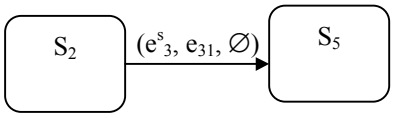


Figure 17. *Envoi d'un message de coopération.*

En mesurant la durée entre l'occurrence de e^s_{16} (début de la consolidation du conteneur C_5) et de celle de e^s_3, le superviseur S_5 associe une nouvelle possibilité à la vérification de la contrainte $C_{s3\ 31}$. Si par exemple cette possibilité vaut 1, le superviseur S_5 envoie un message de résultat de coopération à S_2, (figure 18).

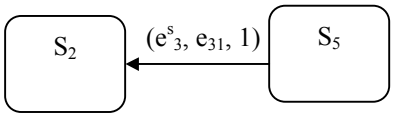


Figure 18. *Envoi d'un message de résultat de coopération.*

Nous avons la même interprétation que précédemment.

Par contre, dans le cas où la possibilité renvoyée dans un message de résultat de coopération vaut 0, le superviseur S_2 détecte alors une violation de la contrainte $C_{s3\ 31}$ et la sous-chronique 2 est non reconnue. Ce qui signifie que la durée autorisée séparant la fin du déchargement du conteneur C_5 et la fin déchargement du conteneur C_2 n'est pas respectée. Les ressources engagées pour effectuer ces déchargements ont probablement généré un retard dans le PPJ qui pourrait affecter les dates limites de départ des deux conteneurs du terminal, ou bien les opérations de déchargement ne sont pas réalisées.
Si la possibilité renvoyée dans le message de résultat de coopération par S_5 vaut par exemple 0.7, le superviseur S_2 conclut que la contrainte $C_{s3\ 31}$ a été vérifiée avec une possibilité égale à 0.5 = Min(0.5, 0.7). La sous-chronique est, dans ce cas de figure, reconnue avec la même possibilité. Si un seuil de vérification de la contrainte $C_{s3\ 31}$ est préalablement fixé, alors cette contrainte peut être vérifiée avec une possibilité valant 1 ou 0. Dans le cas contraire, un opérateur pourrait contrôler les ressources engagées pour les opérations de déchargement et vérifier que les conteneurs ont été bien déchargés.

V.7. Conclusion

L'exemple de nous venons de traiter dans cette partie est basé sur la surveillance du fonctionnement d'un terminal intermodal. Les réseaux de Petri ont été utilisés afin de modéliser les installations de transport, avec l'objectif d'exploiter leur capacité à représenter les besoins de synchronisation dans l'utilisation concurrente de ressources partagées. Une architecture de surveillance distribuée a été utilisée se basant sur un ensemble de sous-chroniques représentant une portion du plan opératoire journalier du terminal. Les mécanismes de communication et de

coopération entre les différents superviseurs ont été mis en évidence par l'étude de scénarios de fonctionnement particulier des composantes considérées du terminal. Les résultats obtenus permettent la détection d'un retard dans le plan opératoire et de l'inachèvement de certaines opérations. Ceci est particulièrement utile lorsque certains événements faisant partie des contraintes à vérifier ne peuvent pas être reçus par les superviseurs.

Conclusion Générale

Les architectures distribuées présentent des solutions adéquates aux problèmes posés par les autres architectures conventionnelles. La réduction de la complexité, la tolérance aux fautes, la réduction des coûts, l'amélioration des caractéristiques tout en maintenant la robustesse et la flexibilité sont les avantages des architectures distribuées.

Nous nous sommes intéressés dans cette thèse à la surveillance distribuée de procédés complexes. L'architecture proposée comporte des sites autonomes et coopératifs réalisant les objectifs de la détection de symptômes de défaillance. Pour cela, la décomposition d'une architecture de surveillance centralisée a été étudiée afin d'obtenir une architecture distribuée. Ceci englobe la décomposition du système surveillé et la distribution des mécanismes de surveillance. Les limites de chacun des sous-systèmes surveillés, issus de la décomposition du système global, ont été précisées en définissant les composants (ressources capteurs, etc.) gérés par chaque sous-système de surveillance et le sous-ensemble d'événements qu'ils génèrent. Nous nous sommes intéressés par la suite à la distribution de la chronique représentant les différentes contraintes temporelles liant les événements générés par le système surveillé. Ceci est réalisé en distribuant les contraintes temporelles sur les sous-systèmes de surveillance donnant lieu à un ensemble de sous-chroniques. La vérification de toutes les contraintes temporelles constituant une sous-chronique implique la reconnaissance de celle-ci et la reconnaissance de toutes les sous-chroniques implique la reconnaissance de la chronique.

La vérification des contraintes temporelles impose aux différents sous-systèmes de surveillance de communiquer lorsque l'information nécessaire n'est pas totalement disponible localement. Ainsi un protocole de communication et de coopération impliquant ces sous-systèmes est défini. Dans ce contexte, il est nécessaire de prendre en compte les délais de communication, supposés négligeables jusque là, et de traiter le cas où certains événements ne peuvent pas être datés par les sous-systèmes de surveillance en prenant en compte, cette fois, des délais opératoires. La prise en compte de délais variables et bornés engendrent une imprécision sur la vérification des contraintes temporelles due à l'incertitude sur la valeur du délai. Le résultat obtenu est une possibilité de vérification d'une contrainte.

Les aspects statiques et dynamiques de la modélisation de chronique ont été précisés. Le cas le plus général concernant l'occurrence multiple d'un événement a également été traité par prise en compte de la multi-sensibilisation dans l'évolution du réseau de Petri et par duplication du réseau de Petri. Lorsque les délais de communication ou opératoires existent, nous avons proposé les réseaux de Petri p-t-temporels flous comme outils de modélisation des sous-chroniques distribuées. Le modèle général de modélisation de sous-chroniques ainsi que les mécanismes permettant de les reconnaître a ainsi été défini et peut être utilisé comme modèle général pour des réseaux de Petri communicants dans le cas où des délais existent ou pas.

Dans la dernière partie de ce travail, nous avons appliqué notre approche sur un système réel représentant un terminal intermodal de fret. Une architecture de surveillance distribuée a été proposée se basant sur un ensemble de sous-chroniques représentant une portion du plan opératoire journalier du terminal. Les mécanismes de communication et de coopération entre les différents superviseurs ont été mis en évidence par l'étude de scénarios de fonctionnement particulier des

composants du terminal. Les résultats obtenus permettent la détection de toute déviation du plan de fonctionnement.

Au terme de ces travaux, plusieurs axes de recherche se dégagent pour envisager, du point de vue des perspectives, de prolonger l'étude menée pendant ces trois ans.

A court terme, une réalisation complète des mécanismes développés pour la surveillance distribuée s'impose. Cette réalisation peut être faite sur un système réel, comme par exemple le terminal intermodal de fret décrit dans cette thèse, ou plus généralement, sur tout système dans lequel l'incertitude sur les délais de communication ou de transport entre sites de commande/surveillance et les tolérances sur les contraintes temporelles sont du même ordre. Comme dans les systèmes informatiques, manufacturiers, embarqués, médicaux, critiques (surveillance de centrales nucléaires), etc. D'autre part, une simulation de ces mécanismes permettrait de valider l'approche proposée.

Sur un autre plan, au moins trois axes de recherche devraient naître.

- La coopération, abordée dans cette thèse et permettant de s'assurer précisément de l'occurrence d'un symptôme de défaillance et d'identifier éventuellement son origine, représente des mécanismes de décision distribuée pour lesquels une étude approfondie doit être faite. La nature du système surveillé et ses caractéristiques doivent être pris en compte lors de la décision de lancement d'une coopération.
- Il est aussi possible de fixer hors-ligne les délais inhérents aux opérations non surveillées afin d'obtenir les performances de la détection escomptées. Pour cela une approche permettant de quantifier les performances du système de détection doit être développée. Cette quantification s'exprime par la possibilité que l'on veut pour la reconnaissance. Ceci permettrait de donner des valeurs possibles pour les bornes du délai afin d'obtenir la ou les possibilité voulue(s).
- Le cas où les bornes du délai ne sont pas connues à priori présente lui aussi une incertitude qu'il faut envisager dans les mécanismes de la détection distribuée. Cette incertitude provient du fait que généralement les bornes des délais (de communication ou opératoires) ne sont pas connues à priori avec précision. Ceci est dû à ce que lorsque ces bornes sont fixées dans un contexte de fonctionnement donné, elles peuvent varier pour un autre contexte dans lequel les données de fonctionnement sont différentes. Une étude statistique des valeurs de ces bornes peut être réalisée sur un échantillon du système surveillé. Il faut signaler enfin que les mécanismes de détection distribuée à développer doivent ainsi prendre en compte l'incertitude relative aux bornes du délai et l'incertitude relative au délai lui même.

A plus long terme, les recherches peuvent se poursuivre afin d'intégrer la fonction détection distribuée dans un modèle distribué global de surveillance-commande. En effet, nous avons indiqué dans cette thèse que des travaux ont amorcé l'étude de problèmes relatifs à la commande décentralisée de systèmes à événements discrets. A partir de là, la réflexion peut être dirigée vers la distribution des fonctions de la surveillance comme le diagnostic, la décision, la reprise, le pronostic, etc.

Références bibliographiques

Azua M. H., "Synthèse de lois de surveillance pour les procédés industriels complexes », thèse de Doctorat, *Institut National Polytechnique de Grenoble*, Septembre 2002.

Balemi S., "Communication delays in connections of input/output discrete event systems", *Proceedings of 31st IEEE Conference on Decision and Control*, pages 3374-3379, 1992.

Balemi S., "Input/Output Discrete Event Processes and Communication Delays", *Discrete Event Dynamic Systems : Theory and Application*, Vol 4, No 1, pp 41-86, Février 1994.

Barrett G., Lafortune S., « A novel framework for decentralized supervisory control with communication", Dans *Proc. IEEE Systems, Man, and Cybernetics Conference*, NewYork, 1998.

Barrett G., Lafortune S, "Decentralized supervisory control with communicating controllers", *IEEE Trans. Automatic Control*, vol. 45, pp. 1620-1638, 2000.

Ben Khalifa N., "Contribution à l'étude de la simulation distribuée des systèmes décrits par réseaux de Petri", *Thèse de Doctorat*, Université Paul Sabatier, Toulouse, France, 1997.

Berruet P., "Contribution au recouvrement des systèmes flexibles de production manufacturière : analyse de la tolérance et reconfiguration ». *Thèse de doctorat*, Université de Lille, Décembre 1998.

Berruet P., Toguyeni A., Elkhattabi S., Craye E. "Toward an implementation of Recovery procedures for flexible manufacturing systems Supervision", *Computers in Industry*, Vol. 43, 2000, pp. 227-236.

Berthomieu B., « La méthode des classes d'états pour l'analyse des réseaux temporels, Mise en œuvre, extension à la multi-sensibilisation », *MSR2001*, Toulouse(France), 17-19 Octobre 2001.

Bischoff J., « The fractal company – success factors of high-performance organisation", *dans Proceedings of Integration in Manufacturing Conference*, Gotenborg, 6-8 Octobre, pp. 257-265, 1998.

Boufaied A., Combacau M., "Etude d'une fonction de surveillance : le pronostic », *4ème Congrès International de Génie Industriel (GI'2001)*, Aix-en-Provence (France), 12-15 Juin 2001, Vol.1, pp.405-414.

Boufaied A., Subias A., Combacau M., "Chronicle modeling by Petri nets for distributed detection of process failures", *IEEE SMC'02*, Hammamet (Tunisie), 6-9 Octobre 2002.

Boyer M., «Contribution à la modélisation des systèmes à temps contraint et application au multimédia », *Thèse de doctorat*, Université Paul Sabatier, Toulouse, France, Juillet 2001.

Cardoso J., Valette R., Dubois D., "Petri nets with uncertain markings", *LNCS Advances in Petri Nets*, vol. 483, G. Rozenberg, Ed. Springer Verlang, 1991, pp 64–78.

Cardoso J., Valette R., Dubois D., "Fuzzy Petri nets: an overview", *13th World IFAC Congress (IFAC'96)*, San Francisco (USA), 30 June - 5 Juillet 1996, Vol. J, pp.443-448.

Cardoso J. « Sur les réseaux de Petri avec marquages flous », *Thèse de doctorat, Université Paul Sabatier*, Octobre 1990.

Camarata S., McArthur D., Steeb R., "Strategies of cooperation in distributed problem solving", *dans Proc. 8th Int. Joint Conf. Of AI*, Karlsruche, FRG, 1983, Vol.2, pp 767-770.

Chaillet-Subias A. "Approche multi-modèles pour la commande et la surveillance en temps réel des systèmes à événements discrets", *Thèse de Doctorat*, Université Paul Sabatier, Décembre 1995.

Chand S., "Decentralized monitoring and diagnosis of manufacturing processes", Dans *International Symposium on Autonomous Decentralized Systems*, Kawasaki, Japan, Mars 1993.

Cieslak R., Desclaux C., Fawaz A., Varaiya, "Supervisory control of discrete-event processes with partial observations", *IEEE Transactions on Automatic Control*, Vol. 33, no. 3, pp. 249-260, Mars 1988.

Cohen M. A., Mallick S. M., "Global supply chains : research and applications", *Production and Operations Management*, Vol. 6, n°3, pp. 193-210, 1997.

Combacau M., Berruet P., Charbonnaud, Khatab A., Zamai E., "Supervision and monitoring of production systems", *Proceedings of MCPL'2000*, Grenoble, 4-6 juillet 2000.

Combacau M., Courvoisier M., « A hierarchical and modular structure for FMS control and monitoring ». *Dans 1st International Conference on AI Simulation and Planning in high autonomy systems*, Tucson, Arisona, Mars 1990.

Combacau M., "Commande et surveillance des systèmes à événements discrets complexes : application aux ateliers flexibles", *Thèse de doctorat*, Université Paul Sabatier , Toulouse, Décembre 1991.

Coudert T., Berruet P., Philippe J.L., "Integration of reconfiguration in transitic systems: an agent based approach", *IEEE SMC 2003*, organised session, Washington, octobre 2003.

Cruette D., "Méthodologie de conception des systèmes complexes à événements 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, Lille, Février 1991.

Da Silveira M., Combacau M., « Distribution de modèles à événements discrets : une procédure systématique basée sur les réseaux Petri », *4ème Colloque Francophone sur la Modélisation des Systèmes Réactifs (MSR'03)*, Metz (France), 6-8 Octobre 2003, pp.487-503.

Da Silveira M., Combacau M., Subias A. "From centralized to distributed models : A systematic procedure based on Petri nets", *IEEE SMC'02*, Hammamet (Tunisie), 6-9 Octobre 2002.

Deb S., Mathur A., Willett P., Pattipati K., "De-centralized real-time monitoring and diagnosis", Dans *Proc. IEEE Conf. On Systems, Man and Cybernetics*, pp. 2998-3003, Octobre 1998.

Debouk R., Lafortune S., Teneketzis, "Coordinated decentralized protocols for failure diagnosis of discrete event systems", *Journal of Discrete Event Dynamical Systems : Theory and Applications*, 10(1-2) : 22-86, Janvier 2000.

Debouk R., Lafortune S., Teneketzis D., "On the Effect of Communication Delays in Failure Diagnosis of Decentralized Discrete Events Systems*", Proceedings of the 39th IEEE Conference on Decision and Control*, Sydney, Australie, Décembre 2000.

Debouk R., Lafortune S., Teneketzis, "A coordinated decentralized architecture for failure diagnosis of discrete-event systems", *In proc of WODES 1998, International Workshop on Discrete Event Systems.* Published by IEE, London, England, August 1998.

Dechter R., Meiri I., Pearl J., "temporal constraint networks", *Artificial Intelligence, Special Volume on Knowledge Representation*, 49(1-3) : 61-95, 1991. Elsevier Science B. V.

Degano C., Di Febbraro A., "On using Petri nets to detect and recover from faulty behaviours in transportation facilities", *IEEE SMC'02*, Hammamet (Tunisie), 6-9 Octobre 2002.

DeWitt W., Clinger J., "Intermodal freight transportation", *dans "Transportation in the new millennium : state of the art and future directions – perspective from transportation research board standing committees"*, National Research Council, Janvier 2000, Washington DC.

Denning E. D., « An intrusion-detection model », *IEEE Transactions on software engineering*, SE-13 : 222-232, 1987.

Dilts D. M., Boyd N. P., Whorms H. H., "The evolution of control architectures for automated manufacturing systems", *Journal of Manufacturing Systems*, Vol. 10, 1991, pp. 79-93.

Dorothy E. D., "An intrusion-detection model". *IEEE Transactions on Software Engineering*, SE-13, pp. 222-232, 1987.

Dousson C., Ghallab M., « Suivi et reconnaissance de chroniques », *Revue d'Intelligence Artificielle*, Vol.8, N°1, pp.29-61, 1994.

Duffie N. A., "Synthesis of heterarchical manufacturing systems", *Computer in Industry*, Vol. 14, 1990, pp. 167-174.

Duffie N. A., Chitturi R., Mou J., « Fault-tolerant heterarchical control manufacturing system entities », *Journal of Manufacturing Systems*, Vol. 7, No. 4, 1988, pp. 357-362.

Duffie N. A., Prabhu V. V., « Heterarchical control of highly distributed manufacturing systems », *Int. Jou. Computer Integrated Manufacturing*, Vol. 9, Num. 4, pp. 270-281, 1996.

Durfee E. H., Montogomery T. A. “A Hierarchical protocol for Coordinating Multi-Agent behaviors”. *Dans Proc. AAAI-90*, 1990.

Erceau J., Chaudron L., Ferber J., Bouron T., « Systèmes personne(s) – machine(s) : patrimoines cognitifs distribués et monde multi-agents, coopération et prises de décision collectives », *dans « Systèmes coopératifs : de la modélisation à la conception »*, Pavard B, Octares Editions, pp. 119-152, 1994.

Erceau J., Ferber J., « L’intelligence artificielle distribuée », *La Recheche* 233 (22), pp. 1245-1263, 1992.

Fabre E. “Monitoring distributed systems with distributed algorithms”. In *Proc of the 2002 IEEE Conf. on Decision and Control*, 411--416, Dec. 2002, Las Vegas, 2002.

Ferber J., “Les systèmes multi-agents. Vers une intelligence collective », *InterEditions*, Paris. 1995.

Ferber J., « Les systèmes multi-agents : un aperçu général », *Dans Techniques et sciences informatiques*, Vol 16 – n°8, 1997.

Froehlich P., Nejdl W., “Resolving conflicts in distributed diagnosis“, *ECAI’96, 12th European Conference on Artificial Intelligence*, John Wiley & sons, Ltd., 1996.

Fujii N. Vaari J., Ueda K., “Potentiel field based simulation of self-organisation in biological manufacturing systems”, *dans Proceedings of manufacturing system design*, 14-16 Mai, Magdeburg, Allemagne, 1997.

Ghallab M., “On chronicles: representation, on-line recognition and learning”, *5ème International Conference on Principles of Knowledge Representation and Reasoning (KR'96)*, Cambridge (USA), 5-8 Novembre 1996, pp.597-606.

Ghallab M., “Chronicles as practical representation for dealing with time, events and actions”, *conf. AIIA’98*, Padoue (Italy), 23-25 Septembre 1998, pp.6-10.

Guan X., Holloway L. E., “Distributed discrete event control structure with controller interactions”, Dans *Proceedings of 1995 American Control Conference*, pages 3151-3156, Seattle, Washington. Juin 1995.

Guan, Xiaoyi, L. E. Holloway, "Control of distributed discrete event systems modeled as Petri nets.", *American Control Conference*, Albuquerque, New Mexico, Juin 1997.

Hadj-Alouane B. N., Lafortune S., Lin F., « Variable lookahead supervisory control with state information », *IEEE Transactions on Automatic control*, vol. 39, no. 12, pp. 2398-2410, Décembre 1994.

Hatvany J., « Intelligence and cooperation in heterarchic manufacturing systems », *Robotics and Computer-Integrated Manufacturing*, Vol 2, Num 2, pp 101-104, 1985.

Holloway L., Chand S., “Time templates for discrete event fault monitoring in manufacturing systems”, *In Proc. 1994 American Control Conference*, pp. 701-706, Baltimore, Juin 1994.

Holloway L., Chand S., "Distributed fault monitoring in manufacturing systems using concurrent discrete-event observations", *Integrated Computer-Aided Engineering*, Vol. 3, No. 4, Octobre 1996, special issue on Detecting and Resolving Errors in Manufacturing Systems.

Holloway L. E., Krogh B. H., "Fault detection and diagnosis in manufacturing systems : a behavioral model approach", *Dans IEEE International Conference on Computer Integrated Manufacturing*, Vol. 2, pp. 252-259, Mai 1990.

Huguet M. J., Erscheler J, De Terssac G., "Autonomie et cohérence dans un réseau de centres de décision : contraintes et négociation", *Journées de Rochebrune : Autonomie et Interactions Fonctionnelles*, Rochebrune, France, 12p, 1994.

Isermann R., « Process fault detection based on modeling and estimation methods – a survey", *Automatica*, vol. 20, n°4, pages 387-404, 1984.

Jahanian F., Rajkumar R., Raju S., "Runtime Monitoring of Timing Constraints in Distributed Real-Time-Systems", *Real-Time Systems*, Vol. 7, No. 3, pp. 247-274, Novembre 1994.

Jones A. T., McLean C. R., "A proposed hierarchical control model for automated manufacturing systems", *Journal of manufacturing systems*, Vol. 5, n°1, 1986, pp. 15-25.

Kanchanasevee P., Biswas G., Kawamura K., Tamura S., "Architectures, Networks, and Intelligent Systems for Manufacturing Integration", *Proceedings of SPIE*, Pittsburgh, Pennsylvania, 15-16 Octobre, pp. 108-115, 1997.

Kandasamy N., Hayes P. J., Murray T. B., "Time-constrained Failure Diagnosis in Distributed Embedded Systems", *Dependable Systems and Networks (DSN 2002)*, pp. 449-459, Washington DC, 2002.

Khansa W., « Réseaux de Petri p-temporels : contribution à l'étude des systèmes à événements discrets », *Thèse de Doctorat*, Université de Savoie, Annecy, France, Mars 1997.

Krogh B. H., "Controlled Petri nets and maximally permissive feedback logic", *Proceedings of 25th Annual Allerton Conference*, Septembre 1987.

Laprie J., C., "Dependability basic concepts and terminology", *ISBN :3-211-82296-8*, 1992.

Lehner P. E., Blackmond K. L., Dubois D, "An Introduction to Issues in Higher Order Uncertainty", *IEEE transactions on Systems, Man and Cybernetics—Part A : Systems and Humans*, vol. 26, No. 3, Mai 1996.

Leitão, P., Restivo, F., "A Layered Approach to Distributed Manufacturing", *dans proceedings of 1999 Advanced Summer Institute - LIFE Cycle Approaches to Production Systems: Management, Control, Supervision*, Leuven, Belgium, 21-23 Septembre 1999.

Leitão P., Restivo F., "A framework for Distributed Manufacturing Applications", *Proceedings of ASI'2000 International Conference*, Bordeaux, France, 2000.

Leitão P., Restivo F., "An Agile and Cooperative Architecture for Distributed Manufacturing Systems", *Proceedings of Robotics and Manufacturing'2001 International Conference*, Cancun, Mexico, 21-24 Mai, 2001.

Lin G. Y., Solberg J. J., "Integrated shop floor control using autonomous agents", *IIE transactions*, Vol. 24, pp. 57-71, 1992.

Lin F., Wonham W., "Decentralized control and coordination of discrete-event systems with partial observations", *IEEE Trans. On Automat. Contr*., vol. 35, no. 12, pp. 1330-1337, 1990.

Lundelius J., Lynch N., "An upper and lower bound for clock synchronization", *Information and Control*, VOL. 62, pp. 190-204, 1984.

Merlin P., "A study of the recoverability of computer system", *Thèse de Doctorat*, Dep. Comput. Sci., Univ., Californie, Irvine, 1974.

Mok A., Liu G., "Early Detection of Timing Constraint Violations at Runtime", *Proceedings of the 18th IEEE Real-Time Systems Symposiums*; San Francisco, California; Décembre 1997.

Okino N., "A prototyping of bionic manufacturing system", *dans Proceeding of the International Conference on Object-Oriented manufacturing systems*, pp. 297-302, Calgary, Alberta, 1992.

Pandalai D. N., Holloway L. E., "Templates languages for fault monitoring of timed discrete event processes", *IEEE Trans. On Automatic Control*, Vol. 45(5), Mai 2000.

Parunak D. V. H., "Applications of distributed Artificial Intelligence in Industry", *dans Foundations of Distributed Artificial Intelligence*, Wiley Inter-Science, 1994.

Patton R. J., Chen J., "Robust model-based fault diagnosis for dynamic systems", Boston : *Kluwer Academic Publishers*, 1999.

Pencolé Y., "Decentralized diagnoser approach : applications to telecommunication networks", Dans *Proc. Of DX'2000, Eleventh International Workshop on Priciples of Diagnosis*, pp. 1762-1768, Décembre 1999.

Peterson J.L., « Petri nets ». *Computing Surveys*, vol 9, pp 153-170, 1977.

Piche P., Kreiger M., Santoro M., "Oligarchy, a control scheme for flexible manufacturing systems », *dans Proc. 2nd IASTED Int. Symp. Of Robotics and Automation*, Lugano, Switzerland, 1983, pp. 35-39.

Pouliezos A.D., Stavrakakis, "real time fault monitoring of industrial processes", *Kluwer Academic Publisher,* Boston, MA, 1994.

Racoceanu D., Zerhouni N., Addouche N., "Modular modeling and analysis of a distributed production system with distant specialized maintenance*", Proc. Of the 2002 IEEE International Conference on Robotics and Automation*, dans CD ROM, pp. 4046-4052, 11-15 Mai 2002, Washington, United States.

Rana S. P., Taneja S. K., "A distributed architecture for Automated manufacturing systems", *International Journal of Advanced Manufacturing Technology*, Vol. 3, Num. 5, pp. 81-98, 1988.

Ravichandran V. J., "Distributed diagnosis for state-based discrete event systems", *Thèse de doctorat*, Dept. of Electrical and computer Engg., University of Toronto, 2002.

Ricker S. L., Schuppen J. H. V., "Decentralized failure diagnosis with asynchronous communication between supervisors", *Proceedings European Control Conference,* pp. 1002-1006, 2001.

Ricker S. L., Rudie K., "Know means no : Incorporating knowledge into decentralized discrete-event control", *Dans proceedings of the 1997 American Control Conference*, pp. 2348-2353, Juin 1997.

Rudie K., Wonham M., "Think globally, act locally: Decentralized supervisory control", *IEEE Trans. Automat. Contr*., vol. 37, no. 11, pp. 1692-1708, 1992.

Schmid U., guest editor, "Special issue on global time in large scale distributed real time systems", *real time systems,* 12(I, II, and III):1-351, Jan, Mar et Mai 1997.

Sengupta R., "Diagnosis and communication in distributed systems", Dans *Proc. Of WODES 1998, International Workshop on Discrete Event Systems*, pp. 144-151. Published by IEE, London, England, 1998.

Shen W., Norrie D., "Hybrid agent-oriented Infrastructure for modelling enterprises", dans *Proceedings of KAW'98 Conference*, Banff, Canada, 1998.

Smith R., G., "The Contract Net Protocol : high-level communication and control in a distributed solver", *IEEE Transactions on Computers*, Vol. C-29, Num. 12, pp 1104-1113, 1980.

Stark W., Directeur de Projet, « Low energy electronics design for mobile platforms", *Rapport de projet AROMURI, Electrical Engineering and Computer Science Department*, Université du Michigan, 1999.

Sujit R. D., Holloway L. E., "Characterizing discrete event timing relationships for fault monitoring of manufacturing systems", *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 2000.

Tharumarajah A., Wells A., Nemes L., "Comparison of the bionic, fractal and holonic manufacturing systems concepts", *dans International Journal of Computer Integrated Manufacturing*, Vol. 9, Num. 3, pp. 217-226, 1996.

Toguyeni A. K. A., « Surveillance et diagnostic en ligne dans les systèmes flexibles de l'industrie manufacturière ». *Thèse de Doctorat*, Université de Lille I, Lille, Novembre 1992.

Toguyeni A. K. A., El Kahttabi, S., Craye E., « Functional and/or, structural approach for the supervision of flexible manufacturing systems", *Dans IMACS Multiconference, Computational Engineering in systems Applications CESA'96*, Lille, France, Juillet 1996.

Tripakis S., “Decentralized Control of Discrete Event Systems with Bounded or Unbounded Delay Communication”, *Dans 6th International Workshop on Discrete Event Systems*, 2002.

Valette R., ”Application and theory of Petri nets 1994”, *Lecture Notes in Computer Sciences 815*, Springer Verlag, N°ISBN 3-540-58152-9, 587p, 1994.

Villemur A., « Sûreté de fonctionnement des systèmes industriels », *Eyrolles Editions*, ISSN 0399-4198, 1998.

Winkler M., Mey M., “Holonic manufacturing systems”, *Dans Europeen Production Engineering*, 3-4 Octobre, 1994.

Warnecke H. J., “The fractal company”, *Springer-Verlag*, 1993.

Wong K., Schuppen V. J., “Decentralized supervisory control of discrete-event systems with communication”, Dans *Proc. International Workshop on discrete Event Systems (WODES’96)*, pp. 284-289, Londres, 1996.

Yeddes M ., Alla H., David R., « On the Supervisory Synthesis for distributed Control of Discrete Event Dynamic Systems with Communication Delays », *14th IEEE International Symposium on Intelligent Control/Intelligent Systems and Semiotics, ISIC/ISAS'99*, Cambridge, Massachusetts (USA), 15-17 Septembre 1999.

Zamaï E, Combacau M and Chaillet-Subias A, "Models and Strategies for Monitoring of Flexible Manufacturing Systems", *9th Symposium on Information Control in Manufacturing*, Nancy-Metz, France, Juin 1998.

Zamai E., “Architecture de Surveillance-Commande pour les Systèmes à Evénements Discrets », *Thèse de doctorat, Université Paul Sabatier*, Septembre 1997.

Résumé :

Le travail présenté dans ce mémoire d'inscrit dans le domaine de la surveillance des systèmes à événements discrets et porte plus particulièrement sur le problème de la détection distribuée de procédés complexes. L'architecture proposée repose sur des sites autonomes et coopératifs de surveillance permettant de surveiller les évolutions du procédé. Ces évolutions, traduisant le fonctionnement attendu (normal) du système ou des situations de défaillance, sont décrites par des chroniques. Chaque chronique est composée d'un ensemble d'événements et d'un ensemble de contraintes temporelles entre ces événements. Trois types de contraintes sont considérés : les contraintes de précédence, les contraintes de types intervalles et les contraintes de type fenêtre d'admissibilité. Une chronique est dite reconnue si toutes les contraintes qui la composent sont vérifiées compte tenu des durées séparant les occurrences des événements la constituant. La distribution du modèle temporel représenté par la chronique induit la distribution de la chronique en sous-chroniques distribuées sur plusieurs sites de surveillance (sous-systèmes). La reconnaissance de l'ensemble des sous-chroniques assure la reconnaissance de la chronique. Cette distribution de la chronique en sous-chroniques permet de mettre en évidence deux types de contraintes : les contraintes locales et les contraintes globales. Les délais de communication ou de transport entre sites sont bornés et sont pris en compte lors de la vérification des contraintes temporelles globales. L'incertitude induite par ces délais engendre une incertitude sur la vérification qui se traduit par une mesure de possibilité associée à la vérification d'une contrainte. Des mécanismes de coopération entre sites de surveillance sont définis. Une modélisation des différents mécanismes développés reposant sur les réseaux de Petri p-t-temporels flous est proposée. Enfin, l'ensemble des travaux est appliqué à la surveillance d'un terminal intermodal de fret.

Mots clés : surveillance distribuée, systèmes à événements discrets, chronique, sous-chronique, réseaux de Petri p-t-temporels flous.

Abstract :

The work presented in this memory is related to the domain of the distributed monitoring of discrete event systems and deals in particular with the problem of the distributed detection of complex processes. The proposed architecture is based on autonomous and co-operative monitoring sites allowing to monitor process evolutions. These evolutions representing the excepted functioning (normal) of the system or failure situations are described by chronicles. Every chronicle is composed of a set of events and a set of time constraints between these events. Three types of constraints are considered : precedence constraints, interval constraints and window admissibility constraints. A chronicle is recognised if all the constraints composing it are verified considering durations separating events occurrences associated to it. The distribution of the time model represented by the chronicle induces the distribution of the chronicle into sub-chronicles distributed on several monitoring sites (subsystems). The recognition of the sub-chronicles set insures the recognition of the chronicle. This distribution of the chronicle into sub-chronicles generates two types of constraints: the local constraints and the global constraints. Communication or transport delays between sites are bounded and are taken into account while verifying a global time constraint. The uncertainty induced by these delays generates an uncertainty on the verification that results in a measure of the possibility associated to a constraint verification. Mechanisms of co-operation between monitoring sites are defined. A modelling of the various developed mechanisms based on fuzzy p-t-temporal Petri nets is proposed. Finally, the work set is applied to the monitoring of an intermodal freight terminal.

Key words : distributed monitoring, discrete event systems, chronicle, sub-chronicle, fuzzy p-t-temporal Petri nets.