N° d'ordre : 666

THESE

présentée à

L'UNIVERSITE DES SCIENCES ET TECHNIQUES DE LILLE
FLANDRES ARTOIS

pour l'obtention du titre de

DOCTEUR

En Productique : Automatique et Informatique Industrielle

par

Denis FOURMAUX

Contribution à la Conception des Systèmes de Contrôle Réparti

Soutenue publiquement le 11 Février 1991 devant la commission d'examen :

MM.	P. VIDAL	Président
	J. DEFRENNE	Rapporteur
	J.P. THOMESSE	Rapporteur
	C. VASSEUR	Directeur de Recherche
	M. COUVREUR	Examinateur
	J.C. GENTINA	Examinateur
	J.M. TOULOTTE	Examinateur

Le travail exposé dans ce mémoire a été effectué au Centre d'Automatique de l'Université des Sciences et Techniques de Lille Flandres Artois sous la direction de Monsieur le Professeur Christian VASSEUR. Avant d'en entreprendre l'exposé, je tiens à exprimer ma reconnaissance à toutes les personnes qui m'ont aidé dans ce travail.

Monsieur le Professeur Pierre VIDAL, Directeur du Laboratoire d'Automatique, trouvera ici mes sincères remerciements pour me faire l'honneur de présider ce jury.

Monsieur le Professeur Christian VASSEUR, Directeur de l'E.N.S.A.I.T. de Roubaix, est le responsable scientifique ; qu'il trouve ici le témoignage de ma gratitude pour ces conseils et l'aide constante qu'il m'a apportée.

Je remercie chaleureusement Monsieur Jean DEFRENNE, Professeur à l'université de Valenciennes, et Monsieur Jean Pierre THOMESSE, Professeur à l'E.N.S.E.M. de Nancy qui ont bien voulu se charger de l'examen de ce travail et participer au jury.

J'exprime ma reconnaissance à Monsieur Michel COUVREUR, Maître de conférence à l'I.U.T. de Béthune, pour ses précieux conseils et ses encouragements.

Je remercie également Monsieur Jean Claude GENTINA, Professeur à l'I.D.N. et Monsieur Jean Marc TOULOTTE, Professeur à l'U.S.T.L.F.A. pour l'intérêt qu'ils ont témoigné à ces travaux et pour leur présence parmi les membres du jury.

Table des Matières

Introduction

La réalisation manuelle d'un travail nécessite de la part de l'opérateur humain une décomposition en tâches élémentaires facilement exécutables. Ensuite l'exécution de ces tâches dans un ordre approprié, lui permet de réaliser au mieux le travail.

L'utilisation de machines conduit à l'augmentation de la production et à l'amélioration de la qualité des produits. Le rôle de l'homme consiste surtout à contrôler et à ordonner l'exécution de ces tâches sur les machines.

Aujourd'hui, en raison du développement de l'automatisation, les machines exécutent ces diverses opérations sans l'intervention humaine. De surcroît, l'évolution des communications entre machines autorise un ordonnancement automatique des tâches. Dès lors, le rôle de l'homme consiste essentiellement à concevoir et à mettre en oeuvre le système réalisant le travail. Pour cela, on définit une démarche en quatre étapes :

1ère étape : avant toute mise en oeuvre, il convient de spécifier ce que l'on veut réaliser. Ceci conduit a définir le cahier des charges.

2ème étape : elle consiste à bâtir un modèle décrivant le comportement du système conformément au cahier des charges.
Pour la définition de ce modèle, deux points de vues peuvent être envisagés :

* le premier consiste à modéliser séparément le comportement de chaque machine et à décrire l'ordonnancement des tâches en regroupant l'ensemble des modèles pour y ajouter la description de la coopération entre machines ;

* le second consiste à étudier le problème dans sa globalité en définissant un modèle du comportement global de l'ensemble des machines. L'ordonnancement des tâches est alors décrit dans la structure du modèle. Le comportement propre à une machine du système est quand à lui décrit par une partie du modèle.

3ème étape : conception d'un algorithme d'exécution du modèle.

4ème étape : mise en oeuvre de l'algorithme sur le système physique.

Cette mise en oeuvre peut conduire soit à une implantation centralisée (solution la plus fréquente), soit à une implantation distribuée.

Cette étude est consacrée à ce deuxième type de mise en oeuvre. Elle est structurée suivant le plan suivant :

* le chapitre I est consacré à une présentation de l'existant dans les différentes phases de conception des systèmes distribués.

* le chapitre II décrit comment est pris en compte le comportement de chaque machine à partir du modèle global d'un système. Il comporte deux volets :

- partition du modèle de description pour aboutir au comportement individuel de chaque machine ;

- définition d'un ensemble de données communes décrivant la coopération entre machines.

* Le chapitre III présente la méthode d'implantation distribuée choisie. L'algorithme d'exécution prend en charge :

- le comportement propre à la sous machine considérée ;

- les échanges avec les autres machines.

* Enfin, le chapitre IV décrit le réseau de communication adopté pour la mise en oeuvre..

CHAPITRE I :

La commande répartie des processus industriels

I-1) Introduction.

Un processus industriel peut être décomposé en deux parties (cf. Fig I-1) :

- La partie opérative (P.O.) qui constitue le procédé à contrôler ;

- La partie commande (P.C.) qui est chargée de contrôler la partie opérative compte tenu des informations provenant de cette dernière et des consignes utilisateur.

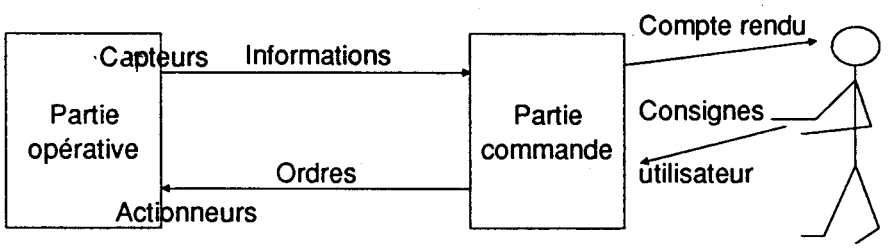

FIG. I-1 : Partie opérative et partie commande

La partie commande (processus) reçoit ses informations des capteurs (fins de courses, interrupteurs,...) et transmet ses ordres aux actionneurs (vérins,moteurs,...) .
L'adaptation des informations et des ordres nécessite le plus souvent un interfaçage.

I-2) Le cahier des charges.

Les spécifications du cahier des charges d'un automatisme ont pour but de préciser :

- Le comportement fonctionnel qu'il doit avoir dans ses interactions avec le procédé et l'opérateur ;

- Les contraintes technologiques, opérationnelles et leurs conversions physiques avec le procédé (capteurs, actionneurs) .

Il faut prêter attention à séparer ces deux niveaux de description. Le premier niveau est indépendant de la réalisation physique. Il ne doit décrire que l'aspect fonctionnel (Niveau I).On y trouve une simple description des combinaisons d'activations séquentielles ou parallèles des opérateurs, sans préjuger de la nature de ces opérateurs.
Le niveau technologique (Niveau II) définit le comportement externe du processus en tenant compte de la nature physique et des caractéristiques des opérateurs (actionneurs,capteurs,...).
On dit souvent qu'un bon croquis vaut mieux qu'un long discours. Pour la représentation des spécifications fonctionnelles des automatismes ont peut utiliser plusieurs types de représentation graphiques (les Réseaux de Petri, le Grafcet, la méthode de Calvez,...).

I-3) Processus centralisés et processus répartis

Dans de nombreuses applications, les opérateurs de coordination et de contrôle de procédés sont pris en charge par un processus de commande unique. Ceci implique que ce processus soit capable de gérer simultanément un grand nombre d'interactions avec le procédé.
Depuis quelques années, l'augmentation de la complexité des procédés de fabrication industrielle en combinaison avec la diminution du prix des composants électroniques bouleverse les méthodes de conception des automatismes.
De plus en plus les applications sont réparties entre plusieurs processus qui communiquent entre eux (cf Fig. I-2) :

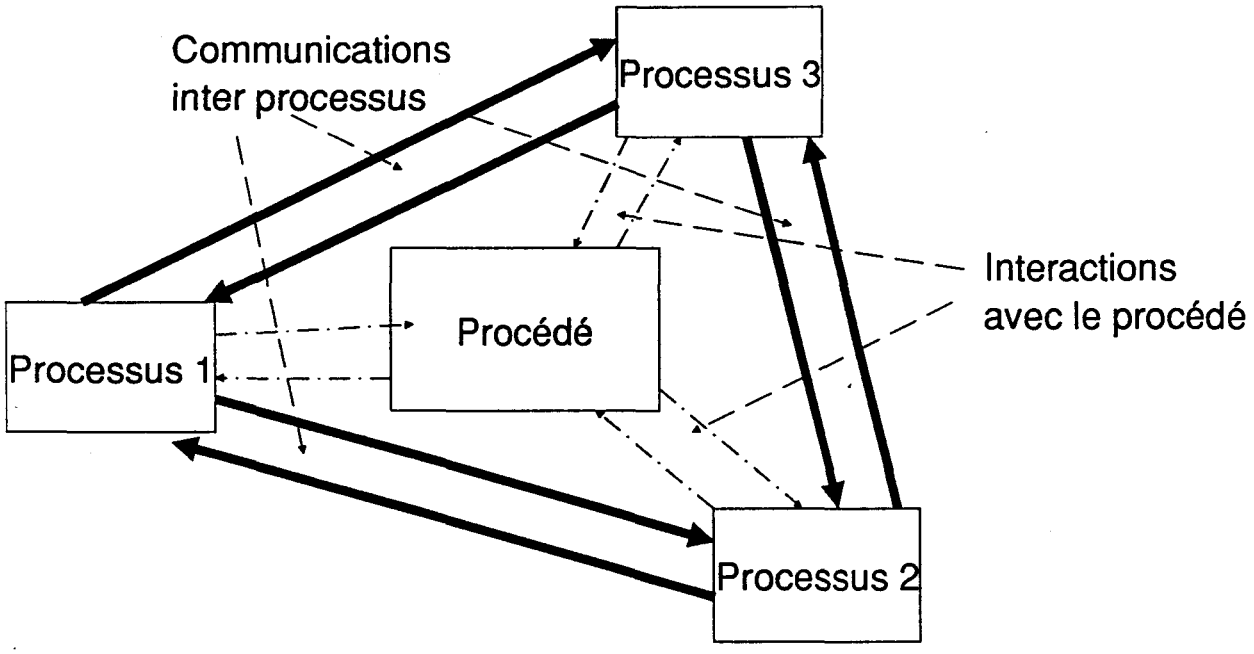

FIG. I-2 : Processus répartis

La coopération à la tâche commune nécessite l'institution d'un dialogue interprocessus.

Une des étapes importantes dans la création d'un système réparti est la modélisation de son comportement. Modéliser un système consiste à représenter son fonctionnement en s'appuyant sur des outils (Mathématiques, Graphiques, ...) permettant d'approcher son comportement.

Pour décrire un système réparti, on utilise des modèles permettant d'effectuer une description du parallélisme existant entre les différentes parties du système distribué. Ce type de modèle est appelé dans la littérature : Modèle du parallélisme [LES 89].

I-4) Evénements et temps dans les systèmes distribués de contrôle de procédés [DES 82]

I-4-1) *Les événements*

Une des différences fondamentales entre systèmes distribués appliqués à la bureautique et systèmes distribués appliqués au contrôle de procédés réside dans le fait que pour ces derniers, les divers échanges sont dépendants de l'environnement. En règle générale, l'évolution du système de commande est tributaire de la variation des entrées.

Pour les systèmes centralisés, les entrées et le passé du système sont regroupés sur un seul site et évoluent par rapport à une horloge commune et unique. Il en découle une mise en oeuvre parfaitement maîtrisée.

Par contre, pour les systèmes distribués, les entrées ainsi que le passé du système sont répartis sur les différents sites du processus de commande. De plus chaque processus possède sa propre horloge et évolue donc de manière asynchrone par rapport aux autres parties du processus global. Pour répartir un processus global en un ensemble de sous processus, il est nécessaire de le décomposer en un ensemble de tâches élémentaires qui peuvent évoluer de manière asynchrone l'une de l'autre.

Pour respecter un fonctionnement global cohérent et déterministe, ces différentes tâches élémentaires agissent en interaction. Pendant les interactions, l'ensemble des processus élémentaires doit avoir un comportement synchrone.

Une solution à ce problème a été proposée par Lamport [LAM 78] où la synchronisation des horloges internes d'un système distribué est réalisée en datant les messages. Malheureusement, cette solution ne peut convenir au contrôle de procédé pour lequel l'évolution du système de commande dépend non seulement d'une horloge interne mais aussi et surtout de la variation des entrées du système de commande. De plus cette méthode offre une précision relativement faible car elle est dépendante des temps de transferts.

Afin de pallier cette difficulté, l'objectif du chapitre II est de montrer, à l'aide de machines séquentielles découlant d'un modèle de type Grafcet ou d'équations d'états dérivées des réseaux de Petri interprétés et de commande (R.D.P.I.C.), qu'un processus global peut être décomposé en un ensemble de tâches élémentaires possédant, selon les circonstances, deux types de comportement :

- un comportement asynchrone des autres tâches ;

- un comportement synchrone des autres tâches lors des phases de communication (rendez vous).

Dans le chapitre III est présenté un algorithme permettant la mise en oeuvre de ces deux comportements sur un système distribué.

I-4-2) *Le temps*

Dans les systèmes de contrôle de procédé, on peut considérer deux bases de temps différentes :

* une base de temps représentant le temps dit "légal" qui correspond au temps exprimé en années, mois, jours,..., généralement obtenue par comptage de "tops horloge" ;

* une base de temps évaluant le temps dit "événementiel" qui est exprimé par une suite d'instants privilégiés, représentant les changements d'états du système.

Les interactions entre processus répartis devant être réalisées de manière synchrone, deux méthodes de conceptions peuvent être envisagées [DES 82]:

* Soit le système est centralisé, au moins dans sa fonction d'interprétation du temps ;

* Soit on contrôle les durées de transferts.

Dans tous les cas, la répartition du passé et des entrées du processus global nécessite que les interactions entre sous processus soient véhiculées par un dispositif de communication d'informations.

Les informations de type logique ou analogique, indispensables à la synchronisation des systèmes entre eux, peuvent circuler sur des câbles spécialisés. Les messages sont courts mais doivent être acheminés en un temps fini et borné.

Ce système de communication offre la possibilité de transmettre des états physiques d'un site à l'autre et ainsi de faire évoluer l'ensemble du procédé comme un système centralisé (au temps de communication près).

Le chapitre II montre comment chaque partie du système peut donc être considérée comme décentralisée au niveau de la fonction d'interprétation du temps associée au comportement interne et comme centralisée pour la fonction d'interprétation du temps liée au comportement inter-processus.
Le chapitre III présente un protocole de dialogue entre processus qui permet de considérer le comportement interprocessus comme centralisé.

I-5) Modèle de description du comportement.

Il n'existe pas de solution universelle aux problèmes que pose la spécification du comportement d'un procédé. Dans la littérature [FDI 89], deux familles de modèles apparaissent : l'une quantitative (ex : les réseaux de files d'attente), l'autre qualitative (ex : les réseaux de Petri).

Les modèles quantitatifs sont des outils tournés vers l'évaluation de performance des systèmes. Pour quantifier les performances d'un système, certaines grandeurs caractéristiques du système sont utilisées (débit, temps de réponse, nombre moyen de ressources occupées, ...). Ces grandeurs permettent d'optimiser le fonctionnement du système. Les modèles sont définis de manière à produire des résultats représentatifs des critères de performance et ceci en utilisant les relations fondamentales qui existent entres les grandeurs mesurées.

Les modèles qualitatifs sont des outils permettant une description du contrôle de comportement des opérations mises en jeu dans le système. Cette description est généralement réalisée en définissant les différents états accessibles d'un système et en les reliant par des arcs qui caractérisent la structure du modèle. Le passage d'un état à un autre se fait suivant les chemins engendrés par la succession état-arc et est tributaire de ce qui est généralement appelée des "Transitions" qui sont en fait des informations de contrôle ou des opérations réalisées.
L'avantage de cette description est de permettre une analyse du comportement du modèle étudié. Un ensemble de propriétés peut être défini (ex : pour les réseaux de Petri, on parle de réseau borné, vivant, ...) et permet de vérifier le comportement du système ou de détecter des anomalies.

Dans la suite de l'exposé, sont présentés des modèles de type qualitatif, l'accent étant mis sur la possibilité qu'offre ces modèles de prendre en compte le fonctionnement distribué des processus (Parallélisme, interaction entre processus, ...).

Les modèles présentés sont classés en deux types [LES 89] :

* les modèles opérationnels ;

* les modèles ordres partiels.

I-5-1) *Les modèles opérationnels.*

Ce type de modèle permet de décrire un système par une machine dont le comportement correspond à l'évolution du système. Les modèles les plus fréquents sont généralement basés sur des systèmes de transitions (ex : réseaux de Petri, Grafcet, ...).

I-5-1-1) Les réseaux de Petri (R.D.P)

Un réseau de Petri est un graphe orienté constitué de places, de transitions et d'un ensemble fini d'arcs orientés reliant une place à une transition ou une transition à une place
(cf. Fig I-3).

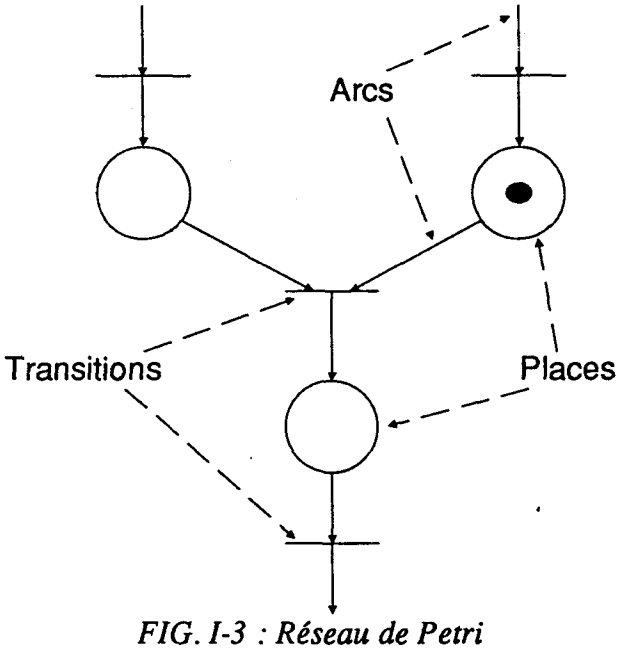

FIG. I-3 : Réseau de Petri

L'évolution séquentielle d'un R.D.P. s'effectue par déplacement d'une marque de place en place suivant l'orientation des arcs. Le déplacement est conditionné par la règle de tir qui s'applique à la transition située entre ces deux places. [BRA 83]

I-5-1-2) Modèle de R.D.P. choisi.

Il existe différents types de réseau de Petri (autonome,généralisé,...). Le modèle choisi dans la suite de l'exposé est le réseau de Petri interprété et de commande, noté R.D.P.I.C. [DAV 89]. Ce type de réseau de Petri a l'avantage d'être bien adapté à la modélisation de systèmes temps réel. Il possède les caractéristiques suivantes :

- il est sauf ;
- non temporisé ;
- déterministe ;
- ses possibilités d'entrées/sorties sont étendues pour être homogènes avec celles du Grafcet (possibilité d'avoir des actions à niveaux, impulsionnelles, conditionnelles,...).

La description du comportement externe (interactions entre le processus de commande et le

procédé à piloter) à l'aide des R.D.P.I.C. s'effectue en synchronisant l'évolution d'un réseau de Petri autonome aux interactions que le processus doit avoir avec son environnement.

I-5-1-3) Spécification du comportement inter processus à l'aide des R.D.P.I.C.

Deux approches sont généralement retenues pour traduire, dans une spécification, les interactions entre processus communicant [TAN 88]

* Dans la première approche, les interactions sont décrites sous forme de places partagées [VAL 82], [COU 83], [BER 85b].

Pour émettre, le processus émetteur tir une transition qui marque la place partagée.

Pour réceptionner le message émis, le processus récepteur doit franchir une transition qui démarque la place partagée (cf. Fig 1-4).

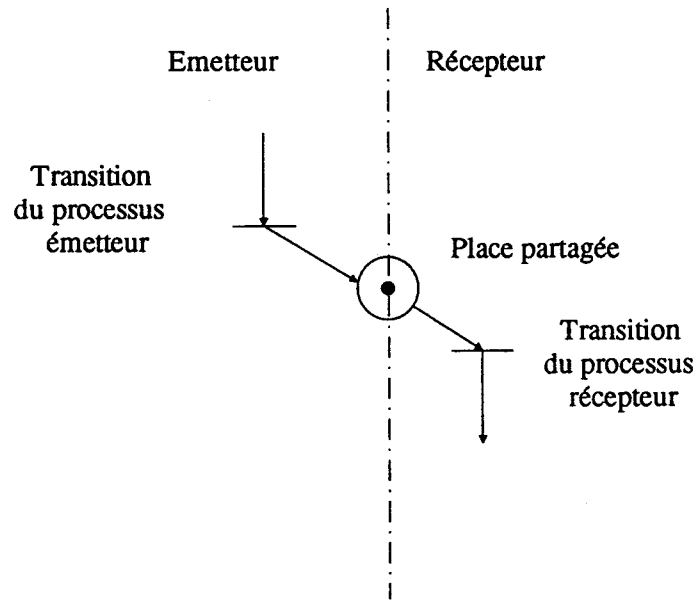

FIG. I-4 : 1 ère approche

* Dans une deuxième approche, les interactions entre processus communicant sont exprimées en synchronisant le tir des transitions à des opérations de communications [BOC 77], en étiquetant l'émission d'un message (notée E!m) à une transition et la réception de ce message (notée R?m) à une autre transition.

Cette notation exprime le fait que les deux transitions ne peuvent être franchies que simultanément (cf. Fig 1-5) :

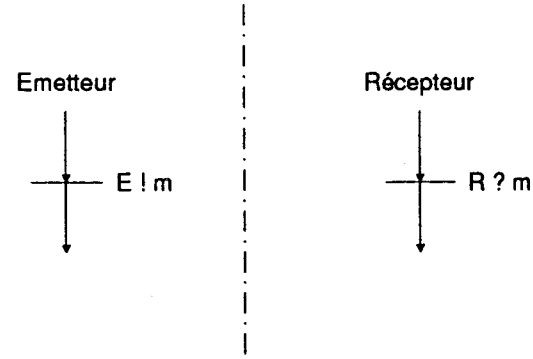

FIG. I-5 : 2 ème approche

I-5-1-4) Les communications.

En retenant la deuxième approche, on peut facilement modéliser les communications entre processus.

I-5-1-4-1) *Communications synchrones.*

Les interactions de communication sont ici considérées comme des événements dont l'occurrence permet le tir de la transition à laquelle ils sont associés. Ces événements sont notés par les expressions !M (émission d'un message M) et ?M (réception d'un message M) [TAN 88] .

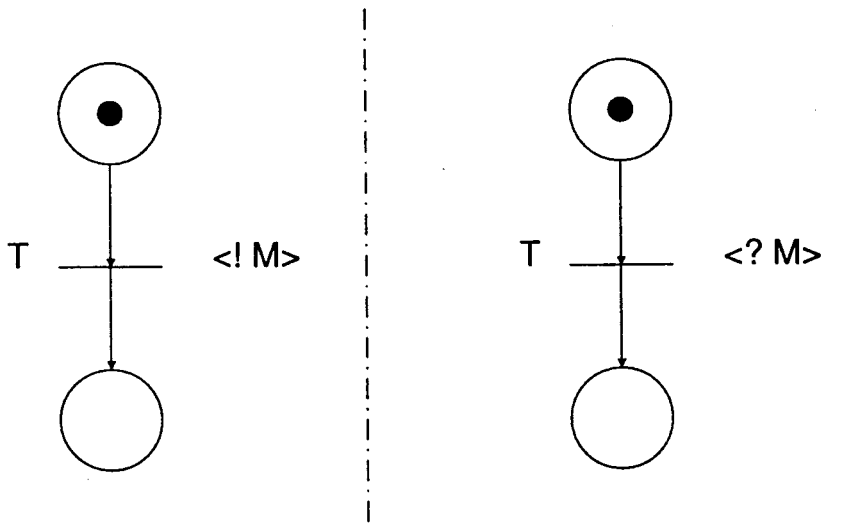

FIG. I-6 : Communication synchrone

Règle de franchissement : une transition T, associée à un événement d'émission ou de réception, devient franchissable si elle est validée et dès que le rendez vous peut avoir lieu.

Dans l'exemple présenté, la notation !M et ?M exprime le rendez vous et le franchissement simultané des deux transitions.

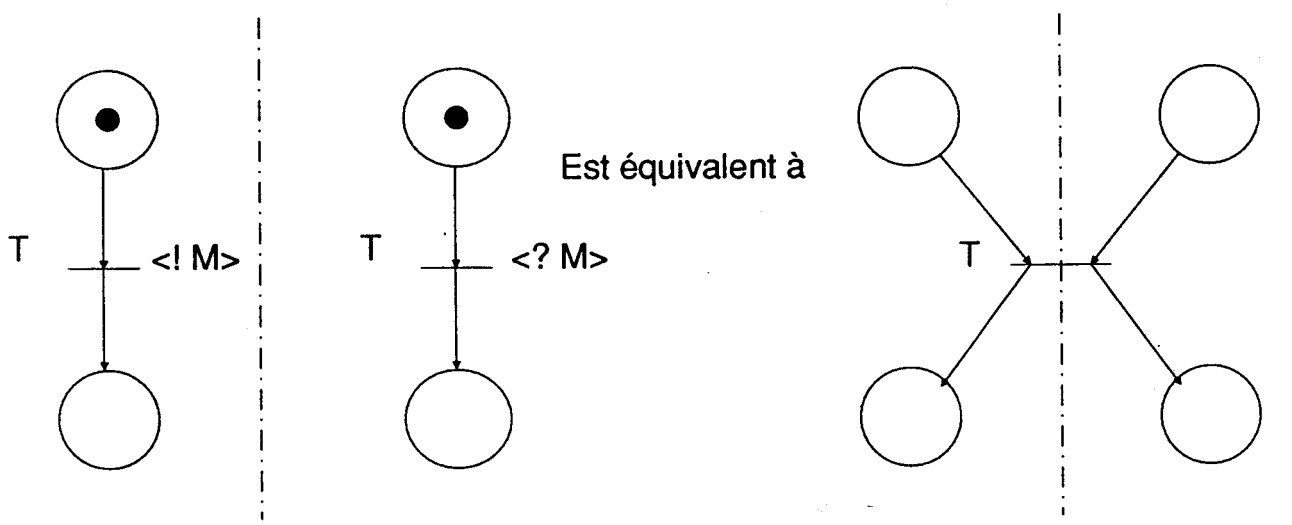

FIG. I-7 : Expression du rendez vous

I-5-1-4-2) *Communications asynchrones.*

Les événements exprimant les interactions entre processus sont notés de la façon suivante :

* !/M Emission d'un message M

* ?/M Réception d'un message M

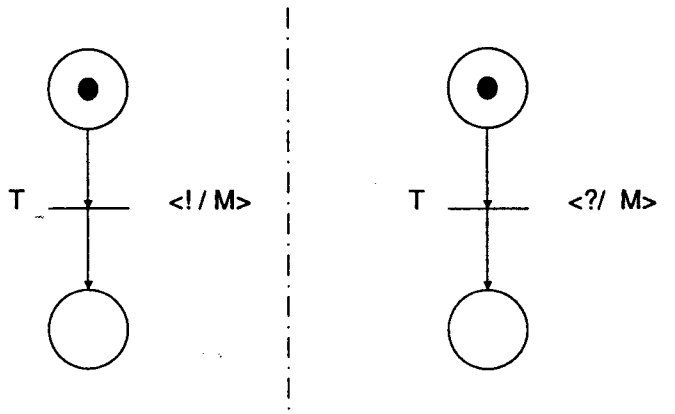

FIG. I-8 : Communication asynchrone

Règle de franchissement : Une transition T, associée à l'événement d'émission, devient franchissable si elle est validée et dès que le message M peut être émis ;

Cette transition T, associée à l'événement de réception, devient franchissable si elle est validée et dès que le message M a été reçu, (Cf. FIG. I-9).

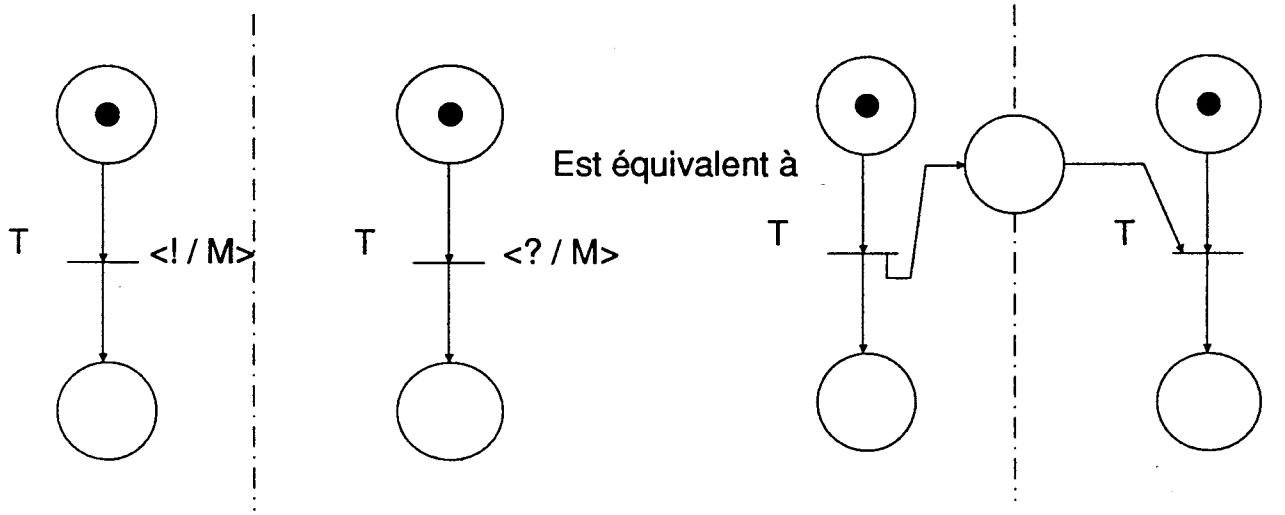

FIG. I-9

I-5-1-5) Le Grafcet.

Le Grafcet est un graphe "booléen" orienté qui résulte d'une collaboration entre l'Université et l'Industrie au sein de l'AFCET (Association Francaise pour la Cybernétique Economique et Technique). Il découle des R.D.P. et est plus simple à mettre en oeuvre pratiquement.

Un Grafcet est un graphe orienté constitué d'étapes, de transitions et d'un ensemble fini d'arcs orientés reliant une étape à une transition ou une transition à une étape. L'évolution séquentielle d'un Grafcet s'effectue en désactivant les étapes amont et en activant les étapes aval de la transition à franchir. L'évolution est conditionnée par la règle de franchissement qui s'applique aux transitions situées en aval de toutes les étapes actives.

La description du comportement externe d'un processus de commande à l'aide du Grafcet s'effectue comme pour les R.D.P.I.C., en synchronisant l'évolution du Grafcet aux interactions que le processus doit avoir avec son environnement.

Dans la suite de l'exposé, à chaque étape j et transition i est associée une variable logique notée respectivement yj et xi (Cf. FIG. I-10). La variable xi représente l'état de la réceptivité associée à la transition i et la variable yj l'activité de l'étape j. Par analogie, l'habitude sera prise d'appeller une transition i par xi et une étape j par yj.

I-5-1-6) Spécification du comportement interne à l'aide du Grafcet.

En utilisant la règle d'évolution du Grafcet qui dit que si plusieurs transitions sont simultanément franchissables, elles sont simultanément franchies, le Grafcet global du processus de commande peut se décomposer facilement en plusieurs diagrammes interconnectés, de la manière suivante :

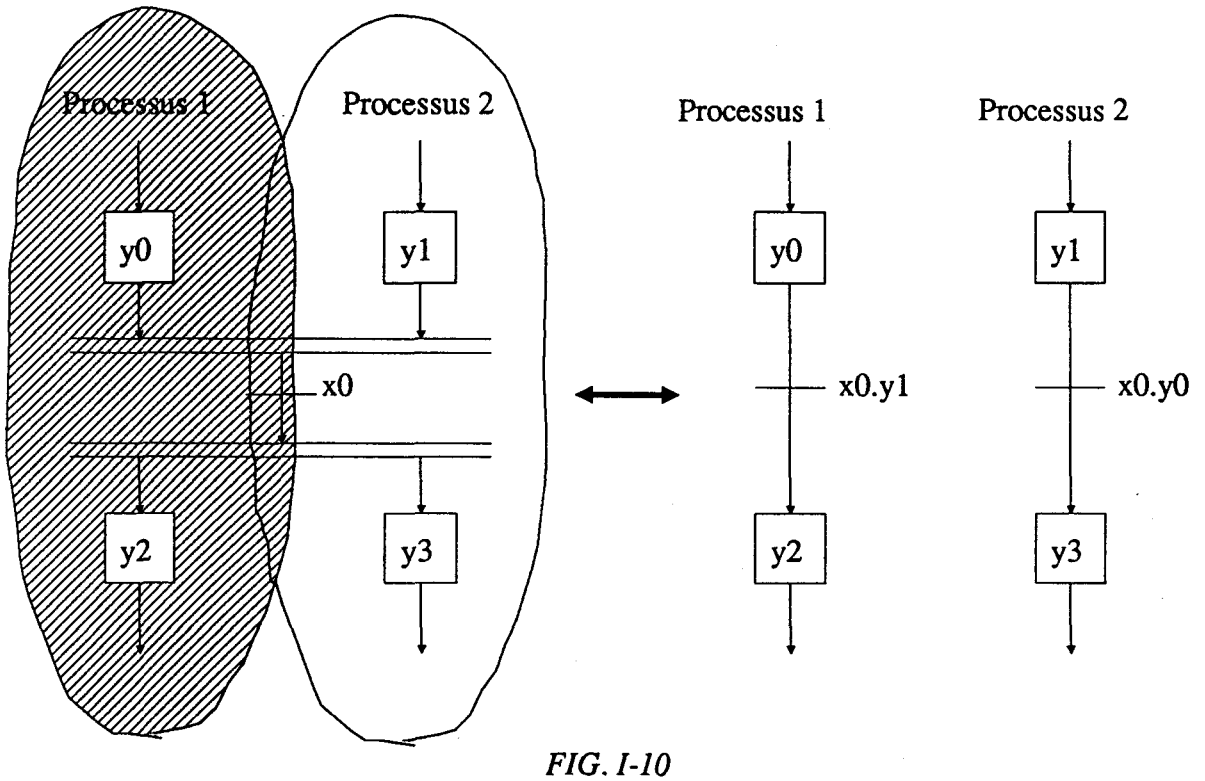

FIG. I-10

Pour un diagramme donné, l'état (pris dans le graphe global) des étapes immédiatement amont à la transition à franchir et n'appartenant pas au processus considéré doit apparaître dans la réceptivité associée à la transition.

I-5-2) *Parallèle entre Grafcet et R.D.P.I.C.[MOA 85], [DAV 89].*

I-5-2-1) Points communs.

- Un Grafcet ou un R.D.P.I.C. comportent deux types de noeuds qui sont les étapes et les transitions pour le Grafcet. Les places correspondent aux étapes dans un R.D.P.I.C. Le Grafcet comme les R.D.P.I.C. ont une représentation graphique assez similaire ;

- les évolutions des marquages, dans les deux modèles, sont synchronisées sur des occurrences d'événements ;

- une nouvelle occurrence d'événement externe ne peut être prise en compte tant que le modèle n'a pas atteint une situation stable.

I-5-2-2) Différences.

- Dans un Grafcet, une étape a deux états possibles : active ou inactive, le marquage d'une étape est booléen. Dans un R.D.P.I.C., les marques s'accumulent dans les places. A tous moment, l'état d'une place est déterminé par le nombre de marques qu'elle contient.

Un R.D.P.I.C. sauf est conçu de manière à ce que chaque place n'ait au plus qu'une seul marque ;

- dans un Grafcet, les transitions simultanément franchissables sont simultanément franchies, ce qui n'est pas toujours le cas dans un R.D.P.I.C. Cela tient au fait qu'une marque dans un R.D.P.I.C. est considérée comme un objet indivisible. Les opérations réalisées sur le Grafcet sont des opérations logiques (ET,OU,...) alors que pour un réseau de Petri, ces opérations sont de type arithmétique (addition, soustraction,...).

- l'état d'une transition dans un Grafcet peut dépendre de l'état du marquage, ce qui n'est pas le cas dans un R.D.P.I.C.

Remarque : La dernière différence peut facilement être contournée [THE 78] en associant à la place dont l'occurrence doit apparaître dans la réceptivité d'une transition, une variable logique qui prend la valeur 1 lorsque cette place est marquée et la valeur 0 lorsque cette place n'est plus marquée.

I-5-3) *Les modèles ordres partiels.*

Ces modèles ont pour but de modéliser le comportement entre sous systèmes. Un système distribué peut être décrit comme un ensemble de sous systèmes interragissant entre eux. Ce type de modélisation porte surtout sur les liens qui rassemblent ou séparent les sous systèmes.

I-5-3-1) La méthode de Calvez. [CAL 82]

Calvez propose une méthodologie de conception des systèmes multi-micro ordinateurs appliquée à la commande en temps réel. En partant de la structure fonctionnelle d'un processus, il aboutit à un modèle structurel décrivant un processus réparti comme un ensemble d'actions élémentaires communicant entre elles (variables partagées, événements, messages).
Une action élémentaire se comporte comme une machine séquentielle cyclique et peut donc être facilement modélisée par un Grafcet.

I-5-3-2) Spécification du comportement interne à l'aide de la méthode de Calvez.

Le comportement propre à chaque processus élémentaire ou fonction [CAL 82] est décrit par une action. Ces actions communiquent entre elles par échanges d'informations que l'on peut classer en trois catégories :

- synchronisation temporelle (événements) ;
- échanges par partage de variables (variables partagées) ;
- échanges par transferts d'informations (messages) via des ports d'entrées/sorties.

Exemple:

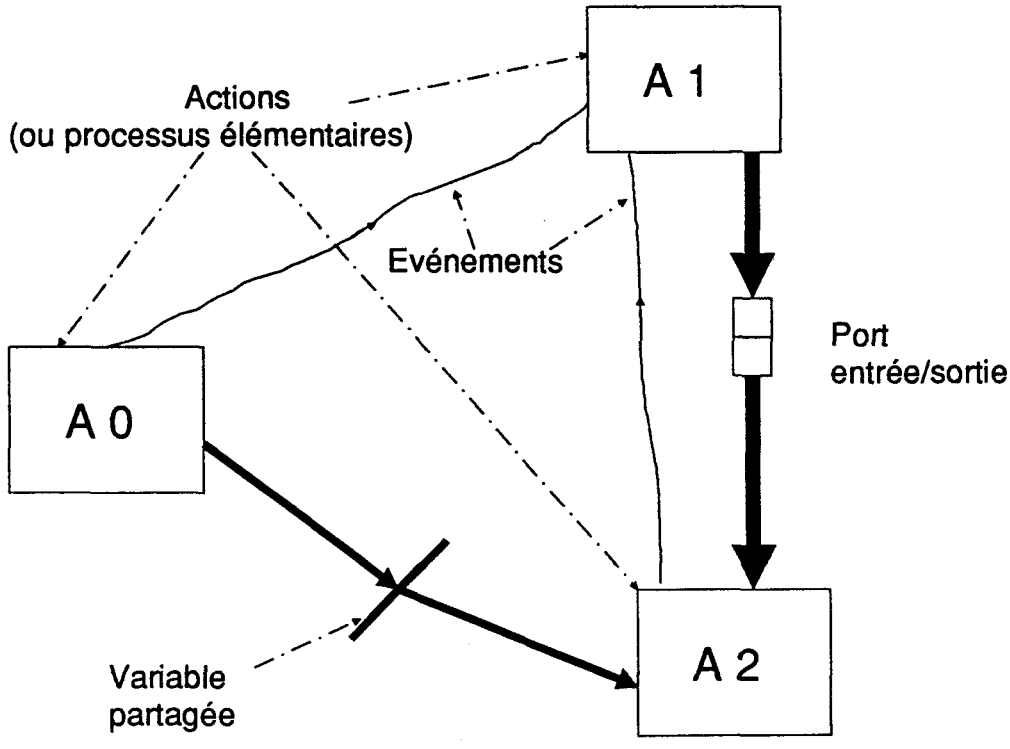

FIG. I-11 : Modèle de Calvez

Dans la suite de ce mémoire, on utilise des modèles opérationnels (Grafcet et R.D.P.) qui se prêtent mieux à une étude du comportement.

I-6) Implantation d'un modèle.

Les modèles du type Grafcet ou R.D.P.I.C. sont des outils graphiques de description du comportement d'un processus. Pour commander un processus, il importe après avoir défini le modèle de comportement de la partie commande, de mettre en oeuvre cette description sur un système physique. Pour ceci, il est nécessaire de définir un algorithme d'interprétation permettant l'exécution du modèle sur un système programmé.
La suite de l'exposé présente quelques méthodes permettant d'implanter un modèle sur un système programmé.

I-6-1) *Processus centralisé.*

Pour une exécution dans un contexte centralisé, il est facile de contraindre un processus à se comporter conformément aux intentions d'un spécifieur en définissant un algorithme d'interprétation chargé d'exploiter une structure de données représentant ces spécifications fonctionnelles.

I-6-1-1) Interprétation d'un Grafcet

Hypothèse : Cet algorithme d'interprétation est basé sur deux conditions :

- deux événements externes non corrélés ne peuvent être simultanés. Ce qui rend le Grafcet déterministe ;

- entre l'occurrence de deux événements externes, on suppose que le Grafcet a le temps d'atteindre un état stable.

Dans la littérature, il existe plusieurs types d'algorithme d'interprétation. Celui présenté ci après permet d'intégrer l'interprétation d'actions conditionnelles, impulsionnelles et d'effectuer une recherche d'états stables.

Premier méthode d'implantation [DAV 89]:

Pas 1 : Initialisation : Activation des étapes initiales et exécution des actions impulsionnelles qui y sont associées. Aller au pas 5.

Pas 2 : Quand un nouvel événement externe se produit, déterminer l'ensemble T1 des transitions franchissables sur occurrence de cet événement. Si T1 n'est pas vide, aller au pas 3. Sinon, modifier éventuellement l'état des actions conditionnelles associées aux étapes actives. Attendre un nouvel événement externe au pas 2.

Pas 3 : Franchir toutes les transitions franchissables. Si la situation est inchangée Après ce franchissement simultané, aller au pas 6.

Pas 4 : Exécuter toutes les actions impulsionnelles associées aux étapes devenues actives au pas 3.

Pas 5 : Déterminer l'ensemble T2 des transitions franchissables sur occurrence de l'événement e (toujours occurrent). Si T2 n'est pas vide aller au pas 3.

Pas 6 : Une situation stable est atteinte.

- Déterminer l'ensemble A0 des actions à niveau qui doivent être désactivées.

- Déterminer l'ensemble A1 des actions à niveau qui doivent être activées.

- Mettre à 0 toutes les actions qui appartiennent à A0 et qui n'appartiennent pas à A1. mettre à 1 toutes les actions qui appartiennent à A1. Aller au pas 2.

Deuxième méthode : En faisant l'hypothèse que le Grafcet ne comporte pas d'action conditionnelle, ni d'action impulsionnelle, il est facile de simplifier cet algorithme:

Pas 1 : Initialisation : Activation des étapes initiales. Aller au pas 5.

Pas 2 : Quand un nouvel événement externe se produit, déterminer l'ensemble T1 des transitions franchissables sur occurrence de cet événement. Si T1 n'est pas vide, aller au pas 3, sinon attendre un nouvel événement externe au pas 2.

Pas 3 : Franchir toutes les transitions franchissables.

Pas 4 : Occurrence de cet événement. Si T1 n'est pas vide, aller au pas 3, sinon attendre un nouvel événement externe au pas 2.

Pas 3 : Franchir toutes les transitions franchissables.

Pas 4 : Déterminer à nouveau l'ensemble T1 des transitions franchissables sur occurrence de ce même événement (recherche d'état stable). Si T1 n'est pas vide, aller au pas 3.

Pas 5 : Une situation stable est atteinte

- Déterminer l'ensemble A0 des actions qui doivent être désactivées.
- Déterminer l'ensemble A1 des actions qui doivent être activées.
- Mettre à 0 toutes les actions appartenant à A0 et qui n'appartiennent pas à A1.
- Mettre à 1 toutes les actions qui appartiennent à A1.
- Aller au pas 2.

Remarque : La recherche d'états stables permet d'obtenir un comportement cohérent avec l'interprétation. En effet, le non respect de cette recherche pourrait conduire à un comportement non déterministe du Grafcet.

I-6-1-2)Interprétation d'un R.D.P.I.C..

L'algorithme d'interprétation de ce type de R.D.P. peut avoir l'allure suivante :

Pas 1 : Initialisation du marquage, effectuer toutes les opérations associées aux places initialement marquées. aller au pas 3.

Pas 2 : Prise en compte des nouveaux événements.

Pas 3 : Si l'ensemble des transitions franchissables sur l'occurrence des nouveaux événements est vide, aller au pas 2. Sinon, effectuer le tir d'une séquence de simulation complète choisie sur occurrence des nouveaux événements.

Pas 4 : Effectuer toutes les opérations associées aux places dont le marquage a augmenté au pas 3. Aller au pas 3 (recherche d'états stables).

I-6-2) *Processus réparti.*

Dans un processus réparti, il n'est pas possible pratiquement de faire évoluer un système en respectant les intentions du spécifieur sans introduire une horloge commune. De plus, l'évolution des différentes parties d'un système peut dépendre, à un instant donné, d'une connaissance de l'état global du système dont on ne dispose pas dans un contexte réparti.

On parle [RAY 85] de contrôle distribué dans un ensemble de processus lorsqu'il n'y a pas de processus maître qui assure en permanence le contrôle global du système. La répartition d'un processus est réalisable en dupliquant l'algorithme d'interprétation du processus sur chaque site composant le processus réparti et en y ajoutant la prise en compte des interactions entre processus distribués.

Pour l'évolution des différentes parties du processus, il n'est en général pas nécessaire que ces dernières connaissent l'état global de tout le système. Une partie du système évolue en fonction de son état local et de données transmises par les autres parties. Grâce à cela, le nombre de messages échangés est faible ce qui permet de garantir une meilleure résistance aux pannes.

Dans le chapitre II, nous montrons comment sont optimisés les messages échangés. Le chapitre III va tenter, quand à lui, de présenter des techniques d'implantation répartie de modèle sur un ensemble de systèmes programmés. La répartition du modèle impose que les systèmes composant le processus global soient interconnectés.

I-7) Mise en oeuvre

Le Grafcet ou les R.D.P sont des outils de description très puissants. Ils sont intéressants pour la réalisation d'automatismes logiques. Leur mise en oeuvre peut être réalisée de manière câblée ou programmée sur des systèmes centralisés ou répartis.

I-7-1) *Mise en oeuvre centralisée [THE 80]*

I-7-1-1) Mise en oeuvre câblée.

Le Grafcet est un outil qui se prête facilement à ce type de mise en oeuvre. La technique utilisée est appelée méthode des cellules Appel-Réponse. Elle consiste à associer à chaque étape une bascule de type RS et à définir un câblage logique entre les différentes étapes et transitions.

I-7-1-2) Mise en oeuvre programmée.

Pour obtenir une mise en oeuvre cohérente, il suffit de réaliser un interpréteur de modèle qui s'appuie sur les algorithmes d'interprétation précédemment définis.

Une technique fréquemment employée pour éviter l'accroissement de la taille du programme en fonction de celle du modèle, est de dissocier les données du Grafcet de son interprétation. Le Grafcet est alors saisie à l'aide d'un éditeur qui peut être textuel ou graphique.

Dans un éditeur textuel, la structure du Grafcet est introduite sous forme de liste que l'interpréteur analysera pour faire évoluer ce Grafcet.

Dans un éditeur graphique, le Grafcet est introduit sous forme de dessin, ce qui permet une vérification plus facile de la bonne construction de ce Grafcet.

I-7-2) *Mise en oeuvre répartie.*

Elle se réalise en coordonnant le comportement de l'ensemble des processus répartis par un dispositif de communication (réseaux d'atelier, cf. annexe). Les techniques généralement utilisées par ce type de réseaux se résument essentiellement à la recopie d'un ensemble de données communes sur chaque site du système distribué (Mapway, Telway, Modebus, ...). Ce bloc de données communes est automatiquement remis à jour sur l'ensemble des sites dès qu'une des données change de valeur.
La transmission de données entre processus est réalisée de manière asynchrone à leur cycle d'évolution respectif et peut conduire, si l'on n'y prend garde à des comportements qui ne sont plus déterministes. Le remède adopté par les constructeurs pour pallier cet inconvénient a été d'accroître considérablement la vitesse de transfert des données entre processus. Cette méthode permet de rendre quasiment négligeable les temps de transfert à coté des temps de cycle automate et donc contribue à diminuer les chances de non déterminisme du comportement global.

I-7-3) *Description de la coopération à l'aide des réseaux d'atelier.*

Les réseaux d'atelier (Mapway, Telway, Jbus, ...) visent essentiellement les applications de coordination entre automates programmables, commandes numériques de machines outils ou de robots. Ce type de réseau utilise généralement le service des mots communs. L'ensemble des mots communs constitue une base de données distribuée entre tout ou partie des équipements du réseau. Chaque station du réseau est chargée d'actualiser une partie de cette base qui lui est réservée et où elle a accès en écriture et en lecture. Le reste de la base ne lui est accessible qu'en lecture.

Cette zone de mémoire commune est mise à jour périodiquement dans toutes les stations connectées.

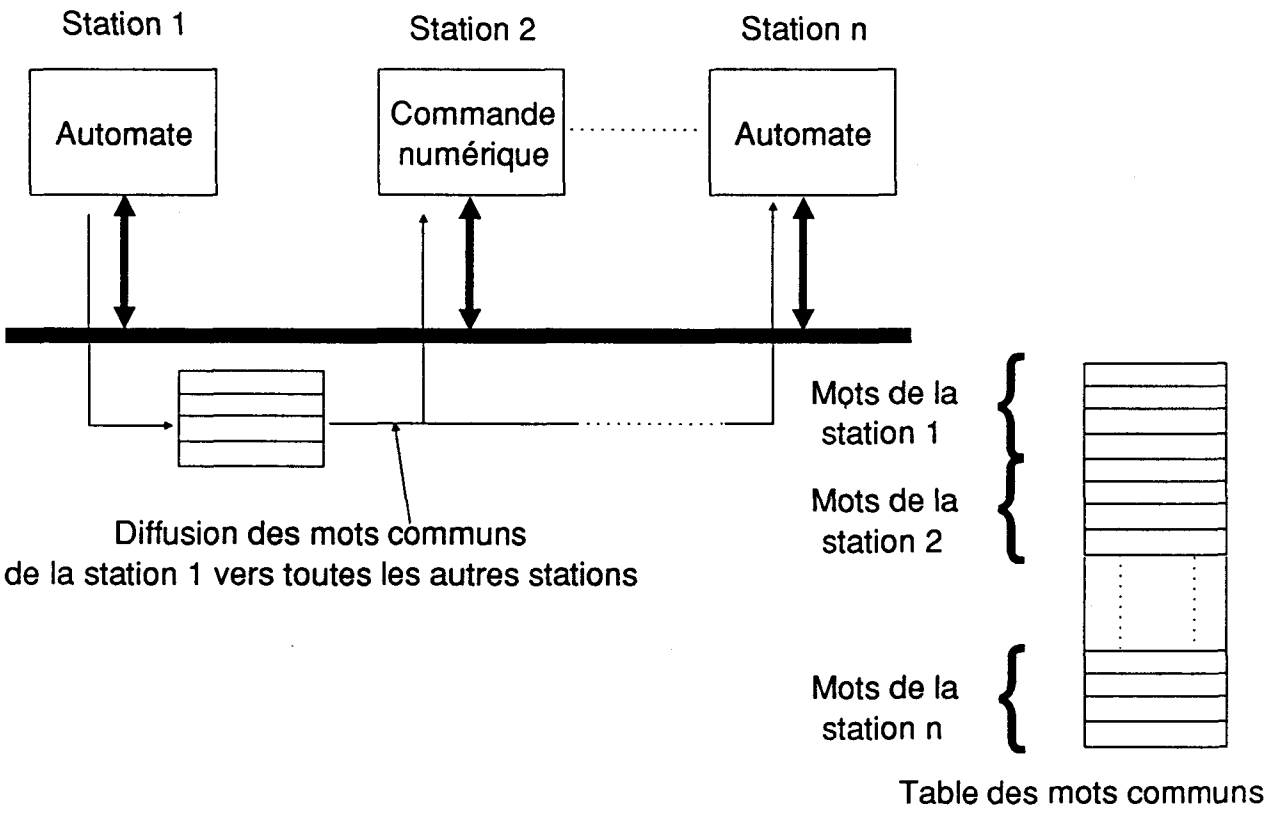

FIG. I-12 : Réseau d'atelier

Exemple de mise en oeuvre sur automate de type TSX 47 (Télémécanique).

Ce type d'automate, programmable en Grafcet, se base pour la mise en oeuvre de la coopération sur les principes présentés en "I-5-1-5)" et utilise, pour coopérer, l'ensemble des mots communs.

Exemple : soit la figure suivante, représentant un rendez vous entre deux processus exprimé à l'aide du GRAFCET :

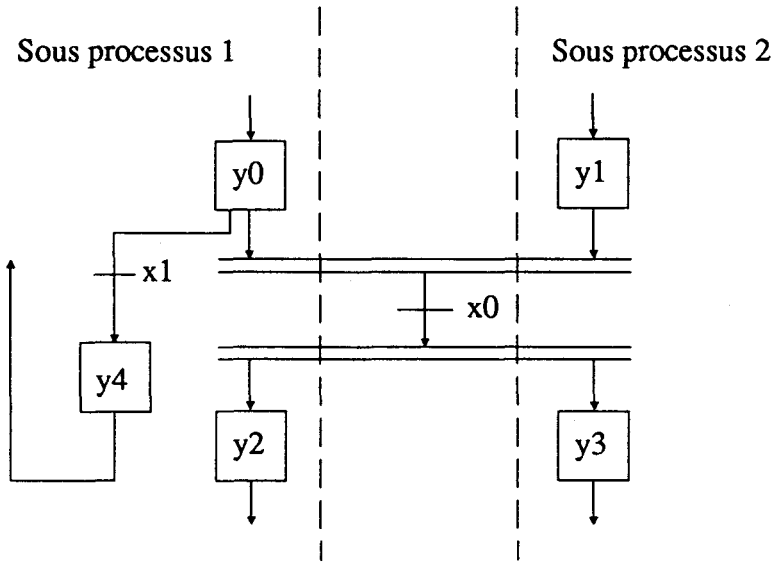

FIG. I-13 : Rendez vous entre 2 processus

Décomposée sous la forme de diagrammes interconnectés, cela donne :

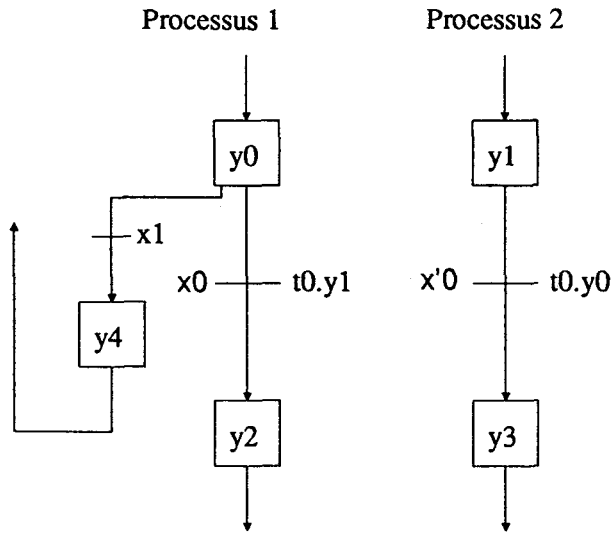

FIG. I-14

Où t0 représente la réceptivité associée à la transition x0 du graphe initiale (FIG. I-18).

Pour coopérer, les processus doivent affecter des variables communes (V0 et V1) qui indiquent que les étapes respectives y0 et y1 sont actives. D'où la transition x0 du graphe initiale a pour réceptivité :

pour le processus 1, t0.V0 ;

pour le processus 2, t0.V1.

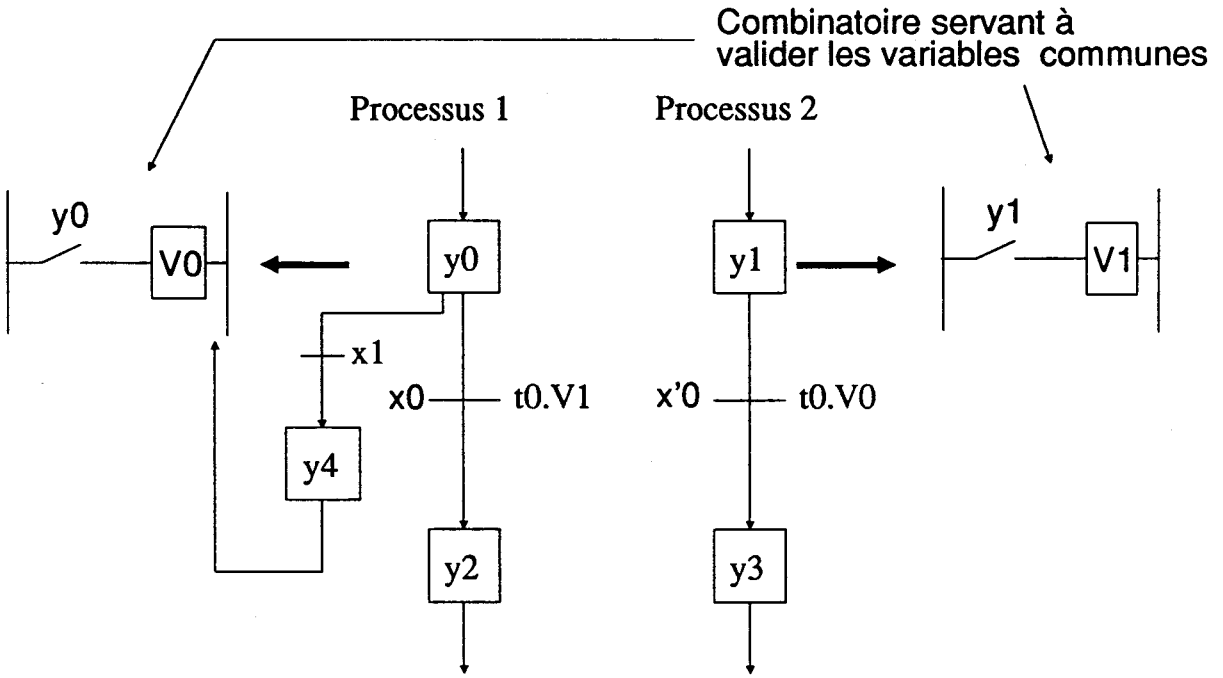

FIG. I-15

Discussion : Les temps de propagation de l'information entre les deux processus ne sont pas complètement négligeables. Par exemple, si y0 est active bien avant y1, dès l'instant où y1 devient active, le système peut évoluer en fonction du temps tel que le représente le schéma suivant :

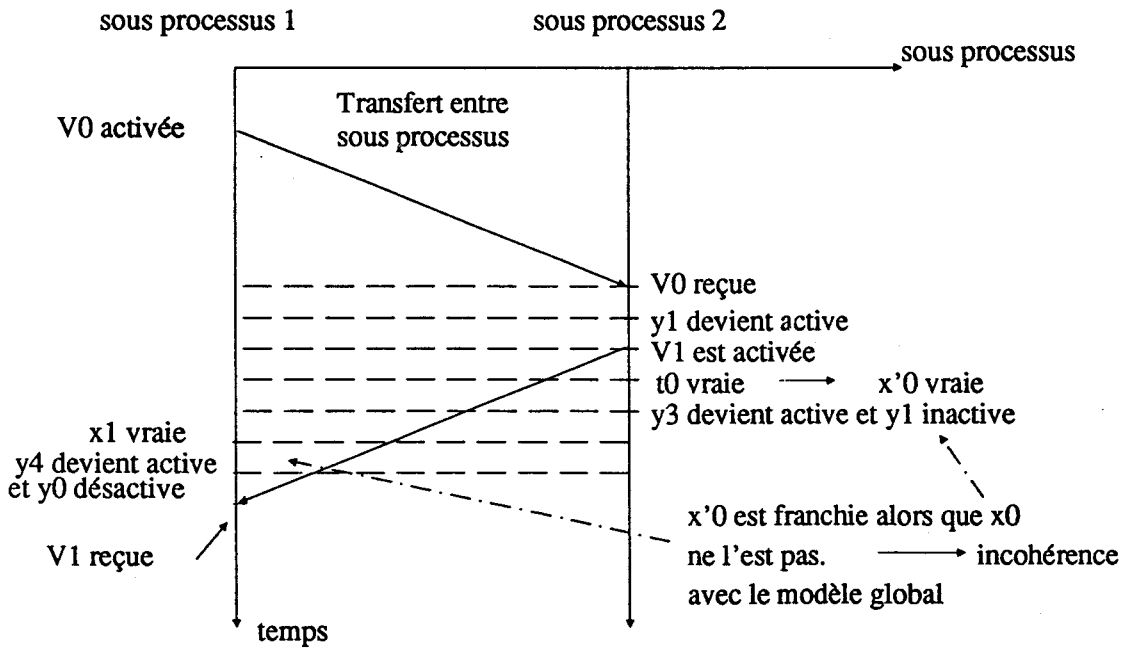

FIG. I-16

Cette exemple nous présente un cas de non déterminisme auquel peut conduire ce type de réseau d'atelier. Pour résoudre ce problème, en supposant que l'automate affecte les variables communes lorsqu'il se trouve dans un état stable, la partie de modèle de la figure I-15 peut être modifiée de la manière suivante :

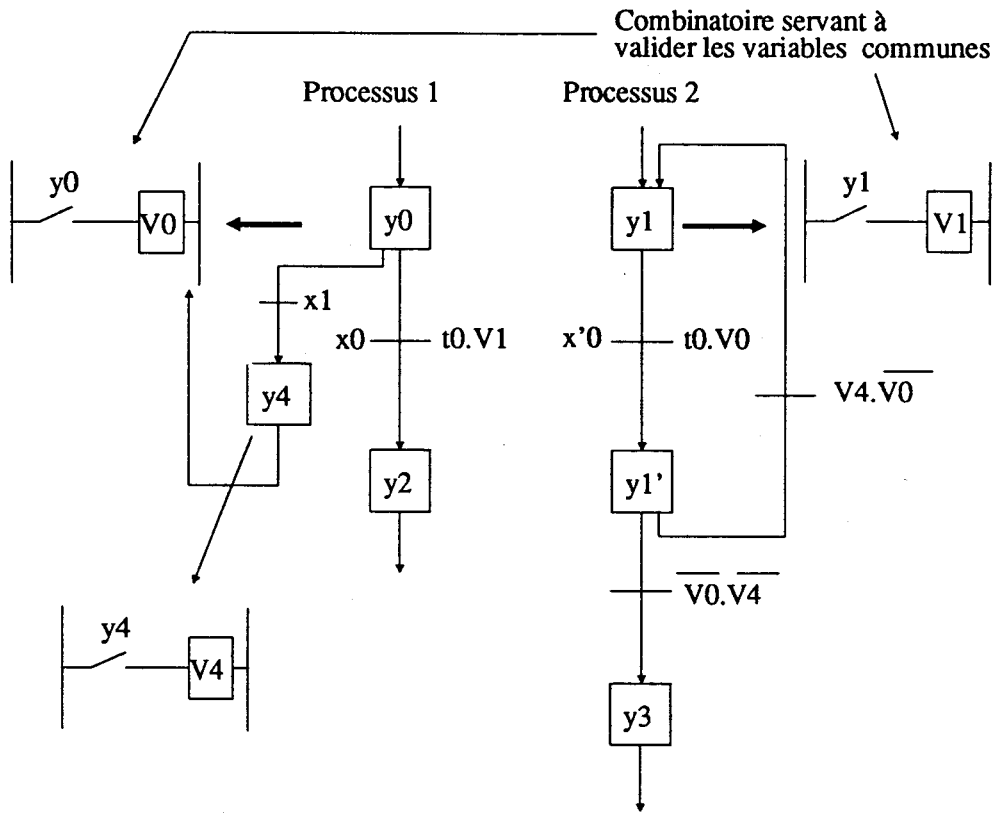

FIG. I-17

La méthode proposée dans la suite de ce mémoire offre la possibilité de résoudre ce problème en rendant synchrone le franchissement parallèle des transitions x0 et x'0.

Conclusion : Ce chapitre a fourni une brève présentation des différents points intervenant dans la réalisation d'un système de contrôle commande :

- Processus centralisé/Processus réparti ;
- Considération événementielle et temporelle ;
- Modélisation ;
- Implantation ;
- Mise en oeuvre.

L'accent a surtout été porté sur l'aspect distribué des processus, pour lequel la suite du mémoire propose une méthodologie de conception. Partant d'un modèle global de comportement, on montre comment est définie la distribution de ce modèle et comment est mise en oeuvre cette distribution.

La méthode décrite, dans la suite de l'exposé, est basée sur un échange synchrone des différents sous processus. Ceci est réalisé en arrêtant l'évolution respective de chaque sous processus pendant les échanges d'informations. Après un échange d'informations, chaque sous processus reprend son évolution avec la même image des informations transmises.

CHAPITRE II :

Partition du modèle.

II-1) Introduction

Quelle que soit la structure physique du processus de pilotage, la première opération à effectuer pour modéliser un processus est de décrire son comportement externe, ce qui fournit un modèle global. Partant de ce modèle global, il est alors facile d'en effectuer une partition sur la base du choix de regroupement des événements et des opérations définis par la structure physique du processus.
Dans la suite de ce chapitre, on présente, en premier lieu, la méthode de décomposition d'un modèle de type Grafcet. On utilise en second lieu, une description algébrique sous forme de machines séquentielles d'un modèle global de type Grafcet et en troisième lieu, on fait une transposition aux réseaux de Petri pour décrire le comportement des différentes parties du modèle partitionné et en extraire un ensemble optimale d'informations à échanger.

II-2) Décomposition sur la base de choix de regroupement des événements et des opérations.

Après avoir défini un modèle décrivant le comportement externe du processus, notre but est maintenant de définir l'architecture interne qui permettra une implantation facile sous forme d'un réseau de processus distribués. Avant de présenter comment la décomposition est effectuée, il est nécessaire d'introduire un certain nombre de notions.

II-2-1) *La notion de doublet*

Doublet Dij : On appelle doublet Dij un ensemble constitué d'une transition xi, d'une de ses étapes en amont yj et d'un arc d'incidence avant liant yj à xi.
Exemple :

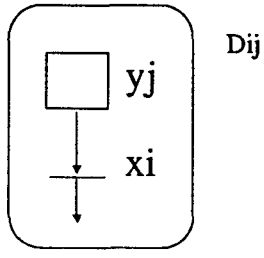

FIG. II-1 : Doublet Dij

Si dans un graphe global, on définit l'ensemble des doublets, deux cas sont envisageable :

* soit, la transition considérée n'a qu'une étape en amont ;

* soit la transition considérée possède plusieurs étapes en amont. Dans ce cas, une même transition intervient dans plusieurs doublets.

Exemple :

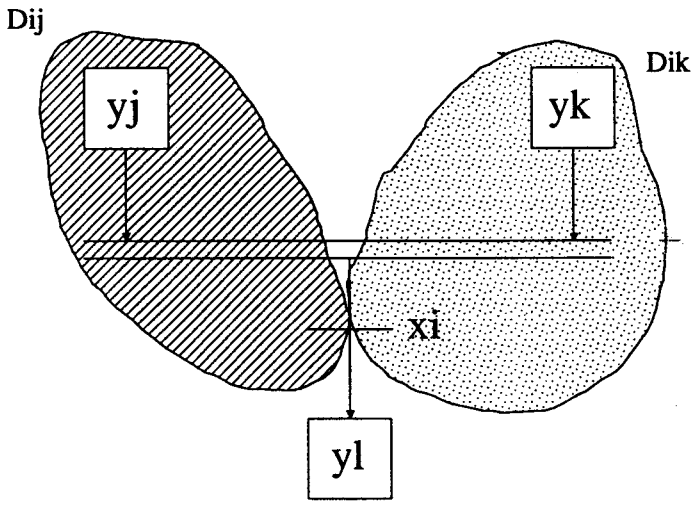

FIG. II-2 : Convergence en ET

II-2-2*) Règle de franchissement.*

En utilisant la notion de doublet, la règle de franchissement d'une transition peut être redéfinie pour un processus partitionné.

Dans le **cas global** une transition xi est franchissable si sa réceptivité est vraie et si chacune de ses étapes en amont est active.

Dans le **cas partitionné**: Deux règles sont nécessaires pour décrire la franchissabilité d'une transition xi

* la transition xi est franchissable pour un doublet D_{ij} si sa réceptivité est vraie et si l'étape yj est active ;

* la transition xi est franchissable si elle est franchissable pour l'ensemble des doublets qui lui sont associés.

Pour traduire la franchissabilité d'une transition par un terme logique, les conditions d'évolution sont utilisées.

II-2-3) *Condition d'évolution globale*

On appelle condition d'évolution [TOU 79] globale CEi(K), une fonction booléenne qui exprime la franchissabilité d'une transition xi à l'instant K. Elle est fonction de la réceptivité associée à xi et de l'activité des étapes immédiatement amont à la transition xi.

Chaque transition xi est associée à une et une seule condition d'évolution CEi(K)

Soit $\mathbf{X_K}$ = x1(K),x2(K),...,xp(K) et $\mathbf{Y_K}$ = y1(K),y2(K),...,yq(K) les ensembles des variables logiques associées respectivement aux transitions et aux étapes du Grafcet à l'instant K. Les conditions d'évolution sont exprimées de la manière suivante:

$$
\begin{aligned}
&CE1(K)=G1[Y(K),X(K)]=g1[Y(K)].x1(K)\\
&CE2(K)=G2[Y(K),X(K)]=g2[Y(K)].x2(K)\\
&\quad\cdot \qquad\qquad\quad \cdot \qquad\qquad\quad \cdot\\
&\quad\cdot \qquad\qquad\quad \cdot \qquad\qquad\quad \cdot\\
&\quad\cdot \qquad\qquad\quad \cdot \qquad\qquad\quad \cdot\\
&CEp(K)=Gp[Y(K),X(K)]=gp[Y(K)].xp(K)
\end{aligned}
$$

Expressions dans lesquelles, gi est le produit logique des variables associées aux étapes immédiatement amont à la transition xi. Sous forme matricielle :

$$
\mathbf{CE}_K = \begin{pmatrix} CE1 \\ CE2 \\ \cdot \\ CEp \end{pmatrix}_K = \begin{pmatrix} g_1 & 0 & \ldots. & 0 \\ 0 & g_2 & \ldots. & 0 \\ \cdot & \cdot & \ldots. & \cdot \\ 0 & 0 & \ldots. & g_p \end{pmatrix}_K . \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ x_p \end{pmatrix}_K \quad \text{"Eq-1"}
$$

ou sous forme condensée

$$
\mathbf{CE_K} = \mathbf{G_K}.\mathbf{X_K} \quad \text{"Eq-2"}
$$

La condition d'évolution est une condition nécessaire et suffisante au franchissement d'une transition. C'est en utilisant cette propriété que la partition va être effectuée.

Règle de franchissement : Une transition xi est franchissable à l'instant K si sa condition d'évolution associée CEi(K) est vraie à l'instant K.

II-2-4) *Condition d'évolution et doublet*

Comme il a été défini précédemment, la condition d'évolution exprime la franchissabilité d'une transition dans le cas global. Pour le cas partitionné, il est nécessaire d'introduire un outil supplémentaire exprimant la franchissabilité d'une transition pour un doublet donné.

On appelle condition d'évolution associée à un doublet Dij et notée CEDij(K), une fonction booléenne qui exprime la franchissabilité de la transition xi pour le doublet Dij. Elle est fonction de la réceptivité associée à xi et de l'activité de l'étape en amont yj.

$$CEDij(K) = x_i.y_j$$

Propriété : Le produit logique de l'ensemble des CEDij(K) pour tous les doublets associés à la transition xi, reconstitue la condition d'évolution globale associée à xi.

Soit une transition xi ayant m étapes en amont, cette transition possède m doublets et sa condition d'évolution globale s'exprime :

$$CEi(K) = \prod_{j=1}^{m} CEDij(K)$$

Π : Produit logique

La règle de franchissement peut alors s'écrire :

Règle de franchissement : Une transition xi est franchissable à l'instant K si le produit logique des conditions d'évolution associées à chaque doublet, où intervient xi, est vrai.

II-3)Partition du modèle de comportement.

La première opération à réaliser pour ceci est d'effectuer une partition du modèle décrivant le comportement global du processus sur la base de choix de regroupement des événements et des opérations. Ceci revient à définir dans le modèle initial des parties de modèle dont le comportement décrit l'évolution d'une partie physiquement distribuée.
Exemple : Soit le Grafcet suivant

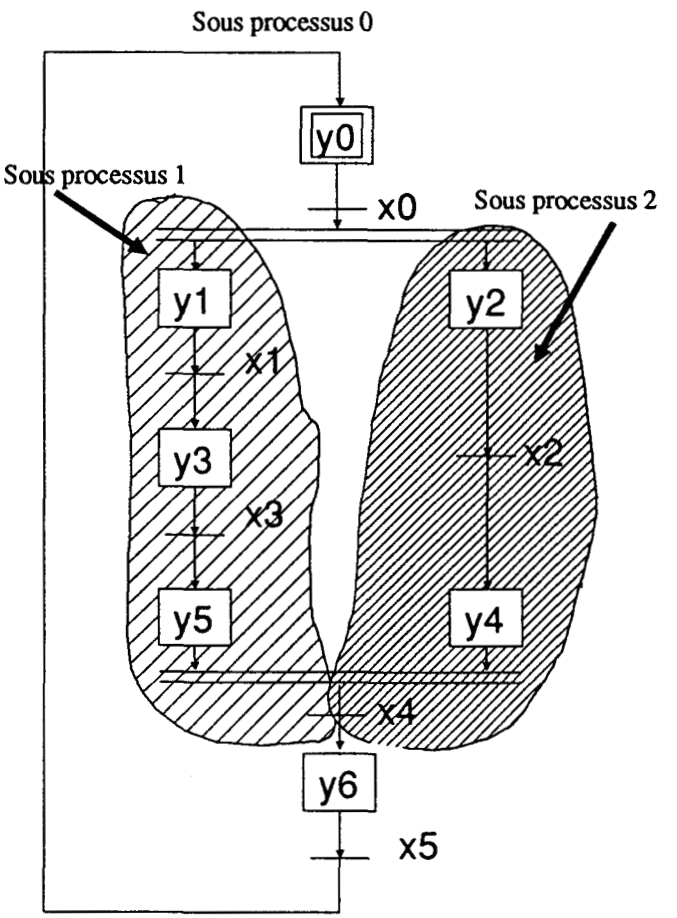

FIG. II-3 : Partition d'un modèle

La décomposition est réalisée en matérialisant les différentes parties du processus distribué par des zones associées à un sous processus donné. Ces différentes zones sont définies en associant à chaque sous processus les doublets qui appartiennent physiquement à cette partie de modèle (Cf FIG. II-4).

Pour la même partition que dans l'exemple précédent, l'ensemble des doublets est partitionné en 3 sous processus:

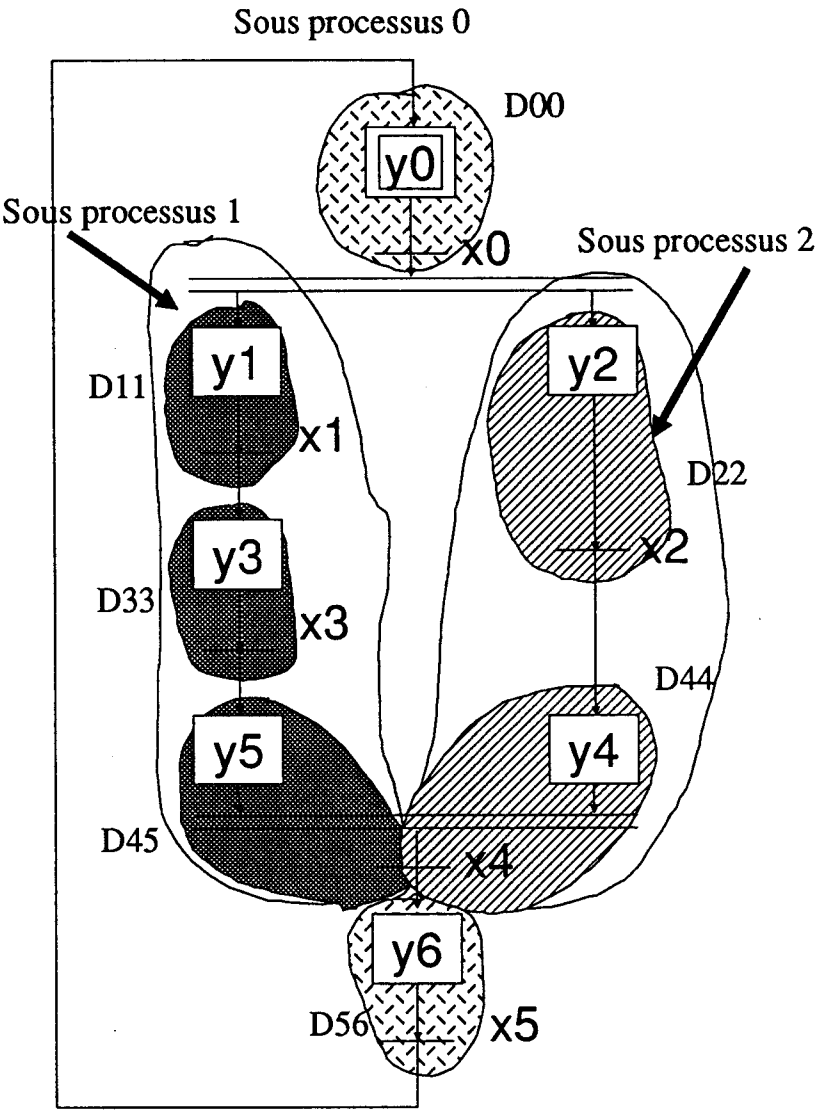

FIG. II-4

Le sous processus 0 est composé de l'ensemble des doublets {D_{00}, D_{56}} ;

le sous processus 1 est composé de l'ensemble des doublets {D_{11}, D_{33}, D_{45}} ;

le sous processus 2 est composé de l'ensemble des doublets {D_{22}, D_{44}}.

II-3-1) *Définitions*

Transition Frontière : Une transition est appelée frontière si elle relie au moins deux doublets n'appartenant pas au même sous processus.

Etape (ou Place) Frontière : Une étape (ou place) est appelée frontière si elle possède au moins une transition frontière en entrée.

Deux ensembles d'éléments de communications sont alors identifiables pour un sous processus :

- l'ensemble des transitions frontières qui constitue un port de sorties virtuel ;
- l'ensemble des étapes (ou places) frontières qui constitue un port d'entrées virtuel.

La partie de modèle correspondant à chaque sous processus peut donc être implantée sur un processeur indépendant, les liaisons inter processus (passage de l'activité entre sous processus) étant assurées par les ports d'entrées/sorties virtuels grâce à un réseau de communication. On obtient alors une répartition de la commande avec traitement du parallélisme et de la synchronisation.

Pour réaliser le passage de l'activité d'un sous processus vers d'autres sous processus, il faut tester la franchissabilité de la transition frontière liant ces sous processus. La partition étant effectuée à l'aide des doublets, le test de franchissabilité d'une transition peut être réalisé en testant la franchissabilité d'une transition pour chacun des doublets associés à cette transition. (Cf. propriété du II-2-4)

II-3-2) *Partition du système et condition d'évolution locale.*

Lors d'une partition, une condition d'évolution peut dépendre de l'activité d'étapes physiquement séparées par la partition. La notion de condition d'évolution locale permet de prendre en compte cette possibilité.

Elle regroupe dans une même variable, le produit logique des conditions d'évolution associées aux doublets d'une transition donnée appartenant au sous processus considéré.

Elle est notée $\mathbf{CEi^j(K)}$: condition d'évolution locale associée à la transition xi pour le sous processus j.

Soit une transition xi ayant m étapes en amont, sa condition d'évolution globale associée est :

$$CEi(K) = \prod_{j=1}^{m} CEDij(K)$$

Π : Produit logique

Si le processus global est partitionné en n sous processus, cela donne :

$$CEi(K) = \prod_{j=1}^{g} CEDij(K) \cdot \prod_{j=g+1}^{k} CEDij(K) \cdot \ldots \cdot \prod_{j=u}^{m} CEDij(K)$$

$$CEi(K) = CEi^1(K) \cdot CEi^2(K) \cdot \ldots \cdot CEi^n(K)$$

où g, k ,...,u dépendent du nombre de doublets d'une transition xi regroupés sur le sous processus considéré.

On peut définir la condition d'évolution locale en s'affranchissant de la notion de doublet :

Condition d'évolution locale : On appelle condition d'évolution locale $\mathbf{CEi^j(K)}$ une condition d'évolution associée à la transition frontière x_i et ayant comme étapes en amont à xi celles associées à la sous machine j.

A partir des différentes conditions d'évolution locales d'une transition frontière xi, on peut reconstituer la condition d'évolution globale du système en écrivant, pour un système partitionné en n parties :

$$CEi(K) = \prod_{m=1}^{n} CEi^m(K) \quad \text{"Eq-3"}$$

C'est à l'aide de cette définition que l'on peut prendre en compte les synchronisations entre processus.

Exemple : Soit une synchronisation entre deux processus matérialisée par une convergence en ET :

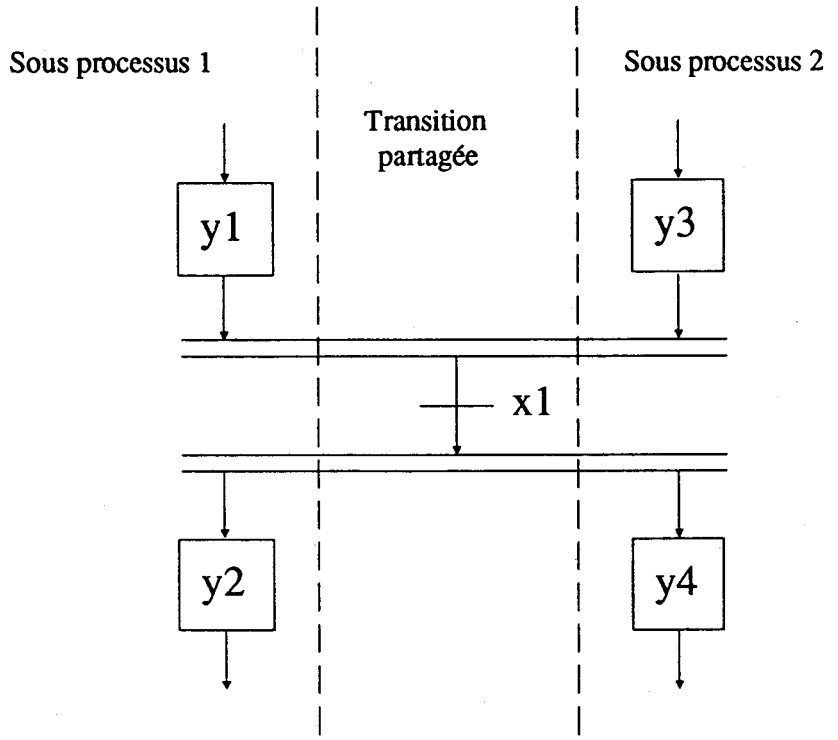

FIG. II-5 : Rendez vous entre deux processus

Cette transition x1 est franchie sur l'activité de ses étapes en amont et de la validité de la réceptivité associée à x1. La condition d'évolution correspondante s'écrie CE1 = y1.y3.x1.
Comme y1 et y3 sont physiquement séparées par la partition, deux conditions d'évolution locales sont définies :

$$CE1^1 = y1.x1$$
$$\text{ET}$$
$$CE1^2 = y3.x1$$

Le franchissement de la transition x1 s'effectue si le produit "$CE1^1$ ET $CE1^2$" est vrai.
Remarque : Les conditions d'évolution locales peuvent être définies :

* soit en associant la réceptivité de la transition partagée à plusieurs sous processus (exemple décrit ci dessus). Ceci implique une réalisation matérielle adéquate ;

* soit en associant cette réceptivité à l'un ou l'autre des sous processus suivant les contraintes matérielles.

Pour ce deuxième cas, les conditions d'évolution locales seront:
soit $CE1^1$ = y1.x1 ET $CE1^2$ = y2
ou $CE1^1$ = y1 ET $CE1^2$ = y3.x1
dont le produit logique donne toujours la condition d'évolution globale associée à x1 :
CE1 = $CE1^1.CE1^2$ = y1.y3.x1.

La suite de ce chapitre montre les effets de ce type de partition sur une description algébrique du comportement d'un processus. Cette analyse permet essentiellement de minimiser l'ensemble des informations traitées par chaque sous processus. Dans le chapitre III, cette description algébrique est reprise pour réaliser l'implantation distribuée du processus.

II-4) Modélisation du comportement d'un processus de commande par Grafcet.

Le modèle utilisé pour la description du comportement externe d'un processus de commande est le Grafcet. Cette description est obtenue en synchronisant l'évolution du Grafcet aux interactions que le processus doit avoir avec son environnement.

II-4-1) *Passage du Grafcet aux équations de récurrence.*

Le passage aux équations de récurrence décrivant une machine séquentielle se fait suivant une méthode tirée de la littérature [ZAH 80], [DEF1 86]. Cette méthode a nécessité une adaptation à la notion de condition d'évolution. Elle se décompose en cinq phases décrites ci-dessous :

a) affecter à chaque étape i une variable y_i ;

b) définir les conditions d'évolution telle que définies précédemment.

c) pour chaque étape i,

$$yi(K) = \sum_{j=0}^{q} fij(CE_K) \,.\, yj(K)$$

Σ : Somme logique où les termes fij sont définis de la manière suivante:

pour tout j correspondant à une étape non immédiatement amont à y_i, fij est identiquement nul ;

pour tous les j tel que y_j est une étape immédiatement amont à y_i, les fij sont définis sur l'ensemble des conditions d'évolution et sont associés aux transitions immédiatement amont à y_i. L'exemple ci après illustre l'évaluation des f_{ij}.

d) appliquer la propriété suivante :

$$\forall i,\ f_{ii}(CE_K) = f_{ii}(CE_K) + \Sigma\, \bar{f}_{ji}(CE_K) \quad \text{pour } j<>i$$

e) éliminer les termes redondants.

Après ces cinq phases, le système d'équations de récurrence s'écrit :

$$\mathbf{Y_{K+1} = F(CE_K).Y_K} \qquad \text{"Eq-4"}$$

$\mathbf{F(CE_k)}$ est une matrice dépendante de $\mathbf{CE_k}$.

Par ailleurs, chaque condition d'évolution qui appartient à une colonne j de la matrice **F** contient la variable y_j. On en déduit une représentation sous forme d' équation mémoire :

$$\mathbf{Y_{K+1} = S(CE_K) + R(CE_K).Y_K} \qquad \text{"Eq-5"}$$

Expression dans laquelle $S(CE_K)$ et $R(CE_K)$ désignent respectivement les matrices d'activation et de désactivation des étapes yi

Afin d'illustrer cette méthode, considérons l'exemple suivant.
phase a) soit le Grafcet :

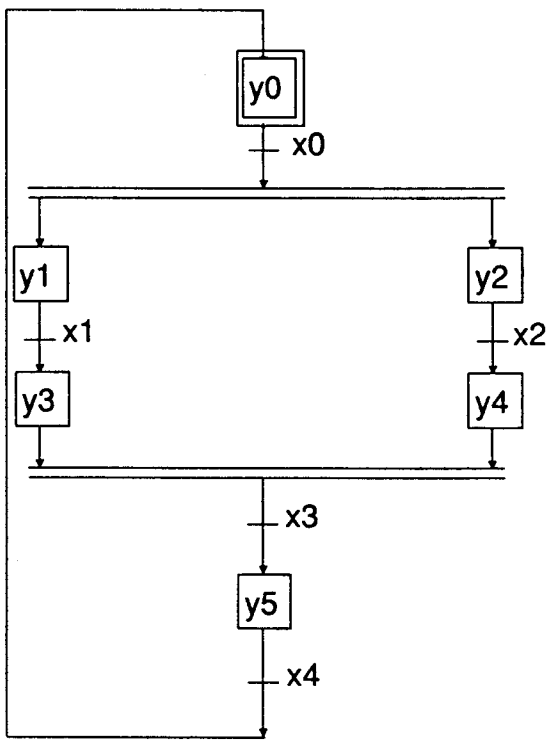

FIG. II-6

phase b)

$$CE_K = \begin{cases} CE0 = x_0.y_0 \\ CE1 = x_1.y_1 \\ CE2 = x_2.y_2 \\ CE3 = x_3.y_3.y_4 + x_3.y_4.y_3 \\ CE4 = x_4.y_5 \end{cases}$$

phase c)

$y0(K+1) = x4.y5(K) = CE4 = CE4.y5(K)$

$y1(K+1) = x0.y0(K) = CE0 = CE0.y0(K)$

$y2(K+1) = x0.y0(K) = CE0 = CE0.y0(K)$

$y3(K+1) = x1.y1(K) = CE1 = CE1.y1(K)$

$y4(K+1) = x2.y2(K) = CE2 = CE2.y2(K)$

$y5(K+1) = x3.y3(K).y4(K) + x3.y4(K).y3(K) = CE3 = CE3.y3\ (K) + CE3.y4(K)$

Sous forme matricielle :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & CE4 \\ CE0 & 0 & 0 & 0 & 0 & 0 \\ CE0 & 0 & 0 & 0 & 0 & 0 \\ 0 & CE1 & 0 & 0 & 0 & 0 \\ 0 & 0 & CE2 & 0 & 0 & 0 \\ 0 & 0 & 0 & CE3 & CE3 & 0 \end{pmatrix}_K \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_K$$

phase d) calculer les termes diagonaux tel que :

$$\forall\, i,\ f_{ii} = f_{ii} + \Sigma_{j<>i}\, \overline{f_{ij}}$$

ce qui donne :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} \overline{CE0} & 0 & 0 & 0 & 0 & CE4 \\ CE0 & \overline{CE1} & 0 & 0 & 0 & 0 \\ CE0 & 0 & \overline{CE2} & 0 & 0 & 0 \\ 0 & CE1 & 0 & \overline{CE3} & 0 & 0 \\ 0 & 0 & CE2 & 0 & \overline{CE3} & 0 \\ 0 & 0 & 0 & CE3 & CE3 & \overline{CE4} \end{pmatrix}_K \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_K$$

où encore sous forme d'équation mémoire :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} CE4 \\ CE0 \\ CE0 \\ CE1 \\ CE2 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} \overline{CE0} & 0 & 0 & 0 & 0 & 0 \\ 0 & \overline{CE1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{CE2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{CE3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \overline{CE3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \overline{CE4} \end{pmatrix}_K \cdot \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_K$$

II-4-2) *Autre expression de l'équation de récurrence.*

Les éléments des matrices $S(CE_K)$ et $R(CE_K)$ sont définis à partir des conditions d'évolution.
$S(CE_K)$ étant un vecteur, on peut trouver une matrice **H1** tel que :

$$S(CE_K) = \mathbf{H1}.CE_K$$

Les termes de **H1** sont des variables booléennes.
De plus, le produit $R(CE_K).Y(K)$ peut se transformer de la même façon :

$$R(CE_K).Y_K = \mathbf{H2}.\overline{CE_K}$$

Où **H2** est une matrice dont les éléments sont définis à partir des y_i .
On peut donc écrire :

$$Y_{K+1} = \mathbf{H1}.CE_K + \mathbf{H2}.\overline{CE_K} \qquad \text{"Eq-6"}$$

Remarque : Les matrices **H1** et **H2** sont indépendantes des entrées.
L'exemple précédent donne :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \end{pmatrix}_K + \begin{pmatrix} y_0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & 0 & 0 & 0 \\ 0 & 0 & y_2 & 0 & 0 \\ 0 & 0 & 0 & y_3 & 0 \\ 0 & 0 & 0 & y_4 & 0 \\ 0 & 0 & 0 & 0 & y_5 \end{pmatrix}_K \cdot \begin{pmatrix} \overline{CE0} \\ \overline{CE1} \\ \overline{CE2} \\ \overline{CE3} \\ \overline{CE4} \end{pmatrix}_K$$

II-4-3) *Partition d'une machine séquentielle.*

II-4-3-1) Introduction.

Une machine M peut être décomposée en n sous machines sur la base de choix de regroupements des événements et des opérations [TAN 88]. Plus précisément, à chaque sous machine, sont associés les événements et les opérations liés à la tâche à exécuter par la sous machine. Pour une machine séquentielle, on peut distinguer deux types de décomposition :

* Sous machines indépendantes.

Dans ce cas aucun événement, état interne ou sortie n'est en commun avec une autre sous machine :

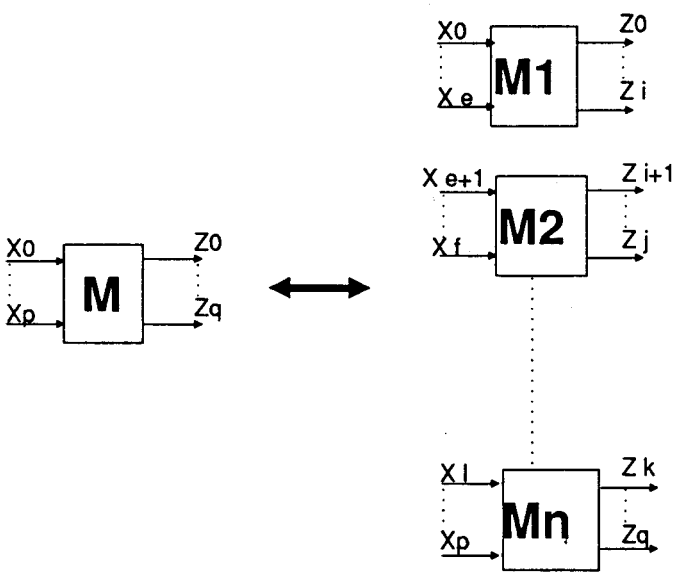

FIG. II-7 : Sous machines indépendantes

* Sous machines dépendantes.

Dans ce cas, certains événements, états internes ou sorties d'une sous machine peuvent influencer le comportement d'une autre sous machine. Cette dépendance nécessite un moyen de communication entre sous machines :

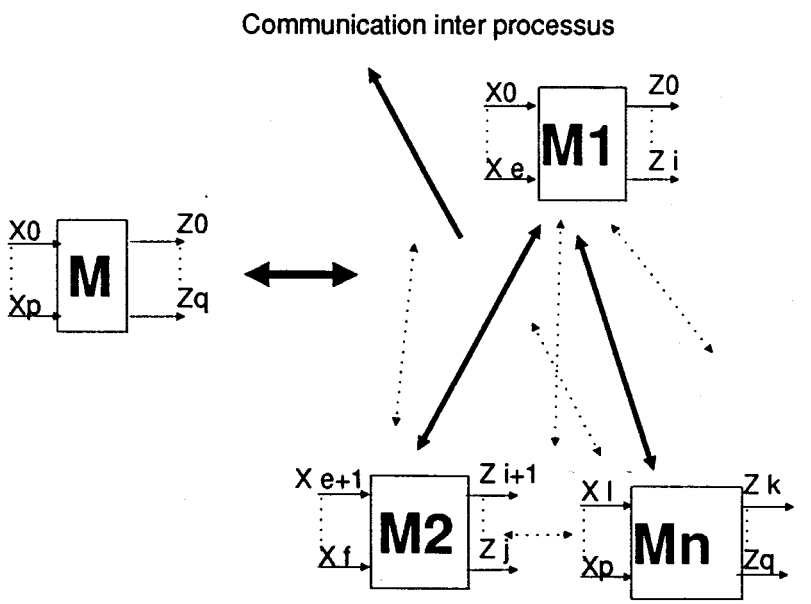

FIG. II-8 : Sous machines dépendantes

II-4-3-2) Définitions.

Forme décomposée [ZAH 80]: On dit qu'un ensemble de sous machines M1,M2,...,Mn constitue une forme décomposée de la machine M lorsque cette ensemble commandé par les mêmes entrées que M se comporte comme M.

Chaque sous machine Mi réalise une fraction $\mathbf{Q^i}$ de l'état **Y**(K) associé à M. L'interconnexion des sous machines Mi est réalisée de la façon suivante :

Chaque Mi reçoit deux types d'entrées :

* un ensemble $\mathbf{E_i}$ appartenant à l'ensemble des entrées **E** de M ;

* un ensemble $\mathbf{F_i}$ constitué par des informations provenant d'autres sous machines.

On peut alors écrire :

$$\mathbf{Q^i}_{K+1} = \mathbf{G^i}[\mathbf{Q^i}_K, \mathbf{Ei}_K, \mathbf{Fi}_K] \quad \text{"Eq-7"}$$

Remarque : On montre que chaque élément de **Fi** est défini sur l'ensemble des conditions d'évolution, d'où :

$$\mathbf{Q^i}_{K+1} = \mathbf{G^i}[\mathbf{Q^i}_K, \mathbf{Ei}_K, \mathbf{CE}_K] \quad \text{"Eq-8"}$$

II-4-4*) Utilisation des conditions d'évolution pour décrire le comportement inter - processus.*

II-4-4-1) Equation de récurrence.

En se basant sur une décomposition [TAN 88] par choix de regroupement des événements et des opérations, l'équation "Eq-6" se décompose de la façon suivante :

$$\mathbf{Y}_{K+1} = \mathbf{H1}.\mathbf{CE}_K + \mathbf{H2}.\overline{\mathbf{CE}}_K = (\mathbf{H1^1} + \ldots + \mathbf{H1^n}).\mathbf{CE}_K + (\mathbf{H2^1} + \ldots + \mathbf{H2^n}).\overline{\mathbf{CE}}_K$$

avec $\mathbf{H1^i}$ et $\mathbf{H2^i}$ comprenant uniquement des données relatives à la sous machine Mi et où chaque termes n'appartenant pas à cette sous machine est remplacé par un zéro logique.

Pour chaque sous machine on peut alors écrire :

$$\mathbf{Q}^{\mathbf{h}}{}_{K+1} = \mathbf{H1}^{\mathbf{h}}.\mathbf{CE}_K + \mathbf{H2}^{\mathbf{h}}.\overline{\mathbf{CE}}_K \qquad \text{"Eq-9"}$$

pour l'exemple précédent, cela donne :

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 0&0&0&0&1 \\ 1&0&0&0&0 \\ 1&0&0&0&0 \\ 0&1&0&0&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \end{pmatrix} \cdot \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \end{pmatrix}_K + \begin{pmatrix} y_0&0&0&0&0 \\ 0&y_1&0&0&0 \\ 0&0&y_2&0&0 \\ 0&0&0&y_3&0 \\ 0&0&0&y_4&0 \\ 0&0&0&0&y_5 \end{pmatrix}_K \cdot \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \end{pmatrix}_K$$

pour une partition $\mathbf{Q}(Y) = \{\mathbf{Q}^0, \mathbf{Q}^1, \mathbf{Q}^2\}$ telle que :

$$\mathbf{Q}^0 = \{y0, y5\}$$
$$\mathbf{Q}^1 = \{y1, y3\}$$
$$\mathbf{Q}^2 = \{y2, y4\}$$

il vient en supprimant dans les matrices $H1^i$ et $H2^i$ les lignes n'appartenants pas à la sous machine Mi:

$$\mathbf{Q}^0 = \begin{pmatrix} y_0 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 0&0&0&0&1 \\ 0&0&0&1&0 \end{pmatrix} \mathbf{CE}_K + \begin{pmatrix} y_0&0&0&0&0 \\ 0&0&0&0&y_5 \end{pmatrix}_K . \overline{\mathbf{CE}_K}$$

$$\begin{pmatrix} CE0 \\ CE4 \end{pmatrix}_K = \begin{pmatrix} y_0&0 \\ 0&y_5 \end{pmatrix}_K \cdot \begin{pmatrix} x_0 \\ x_4 \end{pmatrix}_K$$

$$\mathbf{Q}^1 = \begin{pmatrix} y_1 \\ y_3 \end{pmatrix}_{K+1} = \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \end{pmatrix} \mathbf{CE}_K + \begin{pmatrix} 0&y_1&0&0&0 \\ 0&0&0&y_3&0 \end{pmatrix}_K . \overline{\mathbf{CE}_K}$$

$$\begin{pmatrix} CE1 \\ CE3^1 \end{pmatrix}_K = \begin{pmatrix} y_1&0 \\ 0&y_3 \end{pmatrix}_K \cdot \begin{pmatrix} x_1 \\ x_3 \end{pmatrix}_K$$

$$\mathbf{Q}^2 = \begin{pmatrix} y_2 \\ y_4 \end{pmatrix}_{K+1} = \begin{pmatrix} 1&0&0&0&0 \\ 0&0&1&0&0 \end{pmatrix} \mathbf{CE}_K + \begin{pmatrix} 0&0&y_2&0&0 \\ 0&0&0&y_4&0 \end{pmatrix}_K . \overline{\mathbf{CE}_K}$$

$$\begin{pmatrix} CE2 \\ CE3^2 \end{pmatrix}_K = \begin{pmatrix} y_2&0 \\ 0&y_4 \end{pmatrix}_K \cdot \begin{pmatrix} x_2 \\ x_3 \end{pmatrix}_K$$

Chaque sous machine élabore une partie du vecteur $\mathbf{CE}_K$. Le regroupement de ces différentes parties et l'application du produit des conditions d'évolution locales (ex :$CE3 = CE3^1.CE3^2$) pour calculer les conditions d'évolution globales, reconstituent le vecteur $\mathbf{CE}_K$

Remarque : l'évolution globale de l'activation des différentes étapes du Grafcet dépend d'un vecteur commun à toutes les sous machines. Les changements d'états de ce vecteur sont synchronisés sur le temps dit "événementiel". L'évolution de l'ensemble des états du procédé peut donc être considérée comme synchrone.

II-4-4-2) Minimisation des communications.

En observant les conditions d'évolution au niveau du comportement d'une sous machine i, on distingue trois types :

- les conditions d'évolution internes à la sous machine i ($CE^i int_K$). Elles ne font varier que l'état interne propre à la sous machine i ;

- les conditions d'évolution d'interaction entre la sous machine i considérée ($CE^i ips_K$) et le reste du processus ;

- les conditions d'évolution externes à la sous machine i ($CE^i ext_K$), qui n'influencent pas son propre comportement.

Le vecteur CE_K peut donc s'écrire, pour une sous machine i, sous la forme d'une somme logique de vecteurs :

$$CE_K = CE^i int_K + CE^i ips_K + CE^i ext_K$$

Vecteurs dans lesquels les conditions d'évolution n'appartenant pas au type du vecteur considéré son remplacées par une valeur logique fausse (zéro logique)

En fonction de ces trois types de conditions d'évolution, l'équation de récurrence "Eq-9" peut se décomposer sous la forme suivante:

$$\begin{aligned} Q^i_{K+1} &= H1^i.(CE^i int_K + CE^i ips_K + CE^i ext_K) + H2^i.(\overline{CE^i} int_K + \overline{CE^i} ips_K + \overline{CE^i} ext_K) \\ &= H1^i.CE^i int_K + H1^i.CE^i ips_K + H1^i.CE^i ext_K + H2^i.\overline{CE^i} int_K + H2^i.\overline{CE^i} ips_K \\ &\quad + H2^i.\overline{CE^i} ext_K \end{aligned}$$

où les produits :

- $H1^i.CE^i int_K$ et $H2^i.\overline{CE^i} int_K$ décrivent le comportement interne au sous processus considéré ;

- $H1^i.CE^i ips_K$ et $H2^i.\overline{CE^i} ips_K$ décrivent le comportement entre le sous processus considéré et le reste du processus ;

- $H1^i.CE^i ext_K$ et $H2^i.\overline{CE^i} ext_K$ décrivent un comportement externe au sous processus considéré et qui donc n'influence pas son comportement. Par conséquent, ces produits ne peuvent être que nuls.

autrement dit

$$Q^i_{K+1} = H1^i.CE^i int_K + H1^i.CE^i ips_K + H2^i.\overline{CE^i} int_K + H2^i.\overline{CE^i} ips_K$$

Ceci peut s'exprimer en éliminant dans les matrices H et les vecteurs CE les lignes et les colonnes n'intervenant pas dans le produit :

$$\mathbf{Q}^{i}_{K+1} = \mathbf{H'1}^{i}.\mathbf{CE}^{i}\mathrm{int}_K + \mathbf{H'2}^{i}.\overline{\mathbf{CE}}^{i}\mathrm{int}_K + \mathbf{H"1}^{i}.\mathbf{CE}^{i}\mathrm{ips}_K + \mathbf{H"2}^{i}.\overline{\mathbf{CE}}^{i}\mathrm{ips}_K \quad \text{"Eq-10"}$$

Comportement interne — Comportement inter-processus

Toujours pour l'exemple précédent :

* $\mathbf{Q}^0$

$$\begin{pmatrix} y_0 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} CE4(K) + \begin{pmatrix} 0 \\ y_5 \end{pmatrix}_K . \overline{CE4(K)} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} CE0 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} y_0 & 0 \\ 0 & 0 \end{pmatrix}_K . \begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

* $\mathbf{Q}^1$

$$\begin{pmatrix} y_1 \\ y_3 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} CE1(K) + \begin{pmatrix} y_1 \\ 0 \end{pmatrix}_K . \overline{CE1(K)} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} CE0 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} 0 & 0 \\ 0 & y_4 \end{pmatrix}_K . \begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

* $\mathbf{Q}^2$

$$\begin{pmatrix} y_2 \\ y_4 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} CE2(K) + \begin{pmatrix} y_2 \\ 0 \end{pmatrix}_K . \overline{CE2(K)} + \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} CE0 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} 0 & 0 \\ 0 & y_4 \end{pmatrix}_K . \begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

De cette manière on fait bien apparaître un vecteur commun à toutes les sous machines qui est : $\begin{pmatrix} CE0 \\ CE3 \end{pmatrix}$

Remarque : Ce vecteur représente la quantité minimale d'informations à communiquer à tous les sous processus pour garantir une évolution globale cohérente.
Dans le cas d'un processus plus complexe, le vecteur $\mathbf{CE}^{i}\mathrm{ips}_K$ n'est pas forcement identique pour chaque sous machine. La quantité minimale d'information $\mathbf{CE}\mathrm{ips}_K$ a faire circuler sur le réseau est décrite par l'ensemble des vecteurs $\mathbf{CE}^{i}\mathrm{ips}_K$. Pour un processus comprenant n sous machines :

$$\mathbf{CE}\mathrm{ips}_K = \bigcup_{i=1}^{n} \mathbf{CE}^{i}\mathrm{ips}_K \quad \text{"Eq-11"}$$

où U représente l'opération d'union entre les ensembles d'informations comprises dans chaque $\mathbf{CE}^{i}\mathrm{ips}_K$

Discussion : Ce type de représentation d'un Grafcet sous forme de sous machines séquentielles, met en évidence pour chaque sous processus, deux comportements :

* l'un "local" ou "interne" au sous processus considéré et dont les variations n'intérressent pas les autres sous processus ;

* l'autre "externe" ou "inter-processus" qui lie l'évolution d'un sous processus considéré aux autres sous processus par l'intermédiaire d'un vecteur commun à tous les sous processus.

II-5) Modélisation du comportement externe d'un processus de commande par les R.D.P.I.C.

Comme pour le Grafcet, il est possible de définir pour un R.D.P.I.C. un ensemble de variables associées aux places et aux transitions. L'évolution du marquage des places du R.D.P.I.C. est décrite sous la forme d'une équation d'état.

La suite de l'exposé montre comment l'équation d'état associée au modèle global peut se décomposer en un ensemble de sous équations d'état associées à un sous processus. Cette décomposition permet comme pour le Grafcet de mettre en évidence un vecteur minimum d'informations communes à communiquer entre chaque sous processus.

II-5-1) *Définitions et représentation graphique.*

II-5-1-1) Définitions [BRA 83], [THE 78].

Un **réseau de Petri** est un graphe orienté, défini par un quadruplet R = <P, T, Pré, Post> où :

P = (p1, p2, ..., pq) est un ensemble fini de places représentées par des cercles ;

T = (t1, t2, ..., tp) est un ensemble fini de transitions représentées par des tirets ;

Pré(p,t) est l'application d'incidence avant qui associe un poid à un arc orienté d'une place p vers une transition t ;

Post(p,t) est l'application d'incidence arrière qui associe un poid à un arc orienté d'une transition t vers une place p (Cf. FIG. II-9).

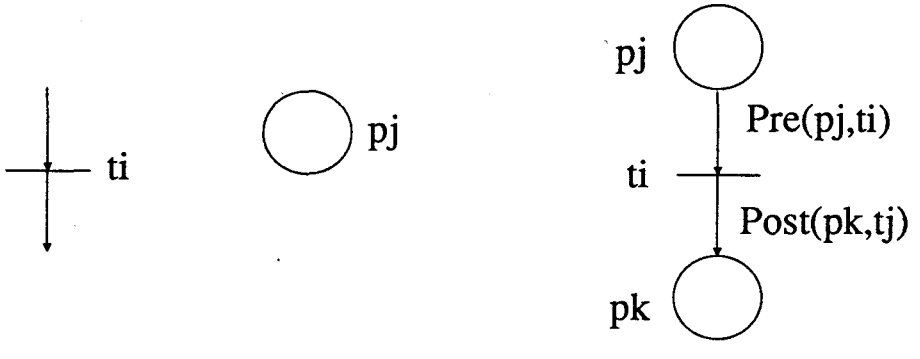

FIG. II-9 : Place et transition

Marquage d'un réseau : il est défini par la présence à l'intérieur des cercles représentant les places, d'un certain nombre de marqueurs (Cf. FIG. II-10).

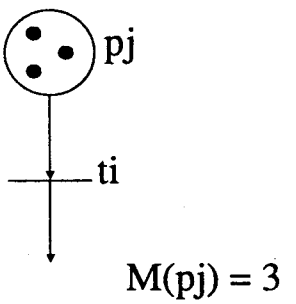

FIG. II-10 : Place marquée

Une place est donc vide ou marquée. Son marquage est noté M(p).
Le marquage M d'un réseau représente le marquage de l'ensemble des places du réseau (Cf. FIG. II-11).

II-5-1-2) Représentation graphique

Un réseau de Petri peut être considéré comme un graphe biparti. Les places et les transitions constituent les sommets (ou noeuds) du graphes qui sont reliés entre eux par des arcs valués (Pre(p,t),Post(p,t)).

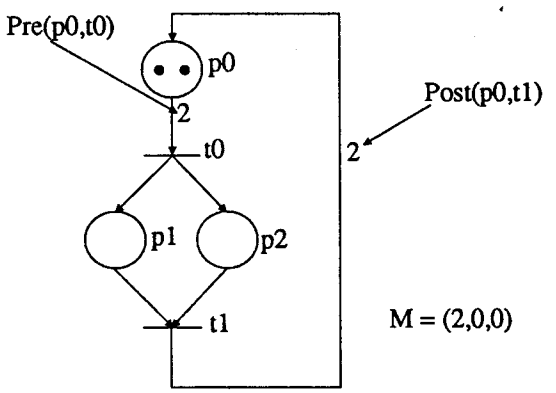

FIG. II-11

Les réseaux de valuation 1 sont des réseaux où chaque arc a comme valuation la valeur 1.
Dans la suite de l'exposé, le type de réseau de Petri utilisé est appelé réseau de Petri interprété et de commande [DAV 89]. Pour plus de précision concernant les R.D.P.I.C. se reporter aux ouvrages cités en référence. Ce type de réseau est de valuation 1.

Modéliser le comportement externe d'un processus par les R.D.P.I.C. consiste, comme pour le Grafcet, à synchroniser son évolution aux interactions que le processus doit avoir avec son environnement.

II-5-1-3) Régles d'arbitrage

Pour disposer d'un modèle qui permette de construire une spécification, il convient de définir l'ensemble des régles d'arbitrage suivantes :

- tous les événements associés aux transitions sont disjoints ;

- pour un marquage accessible M et un événement e, il ne peut y avoir plusieurs transitions simultanément franchissables. Dans le cas contraire, il faut définir des priorités qui rendent le R.D.P.I.C. déterministe ;

- entre chaque évolution de marquage du R.D.P.I.C., une transition ne peut avoir au plus qu'une occurrence ;

- les réseaux utilisés ne possèdent pas de place à la fois d'entrée et de sortie d'une même transition.

II-5-2) *Représentation des machines séquentielles à l'aide des R.D.P.I.C..*

Les machines séquentielles peuvent être représentées [THE 78] par des graphes de Petri interprétés et de commande tels que :

- à chaque transition ti du graphe est associée une variable booléenne xi fonction des entrées. Une transition ti validée est tirée si l'événement correspondant à cette transition se produit. L'ensemble des xi est noté X ;

- à toute place pj du graphe est associée une variable yj représentant le marquage de cette place. L'ensemble des yj est noté Y.

L'équation fondamentale des réseaux de Pétri peut être écrite sous la forme d'une équation contenant des variables logiques, des variables arithmétiques et des opérateurs arithmétiques de la façon suivante :

$$\begin{pmatrix} y1 \\ y2 \\ . \\ yq \end{pmatrix}_{K+1} = \begin{pmatrix} y1 \\ y2 \\ . \\ yq \end{pmatrix}_{K} + C(p,t) . \begin{pmatrix} x1 \\ x2 \\ . \\ xp \end{pmatrix}_{K} \qquad \text{"Eq 12"}$$

Dans cette expression C(p,t) est une matrice appelée matrice d'incidence dont les éléments Cji sont définis sur l'ensemble {-1, 0, 1}. L'indice K caractérise l'instant d'évaluation.
Pour une place pj donnée, on peut donc écrire :

$$yj(K+1) = yj(K) + \sum_{i=1}^{p} Cji \, . \, xi \qquad \text{"Eq-13"}$$

II-5-2-1) Description des règles de fonctionnement à l'aide d'équations algébriques.

De manière analogue à celle du Grafcet, l'évolution du marquage des places peut être décrite par une équation contenant des termes logiques et arithmétiques. Le franchissement d'une transition xi est tributaire du marquage des places amont décrites dans l'équation "Eq-12" par la variable logique yj.

Dans le cas d'un R.D.P. généralisé, une transition xi est tirable pour un marquage M si et seulement si [BRA 83] : $\forall$ yj, $M(yj) \geq Pre(yj,xi)$.
Pour un R.D.P.I.C., la condition de tir s'écrit : $\forall$ yj,$M(yj) = 1$

Règle 1 : Le tir d'une transition xi s'exprime par une fonction combinatoire des états du réseau. Cette fonction booléenne est notée fi(Y). La transition xi est tirable, si et seulement si, fi(Y) est vraie,ce qui s'écrit:
$\forall$ xi, si $fi(Y) = 1$ alors xi est tirable.
Le tir d'une transition provoque l'évolution du marquage M du réseau [BRA 83] tel que le nouveau marquage obtenu M' par le tir de cette transition soit:

$$M'(yj) = M(yj) - Pré(yj,xi) + Post(yj,xi)$$

Compte tenu de la règle 1, le tir d'une transition xi s'exprime par une expression booléenne associant la franchissabilité de cette transition, décrite par fi(Y), et sa validité. Cette expression booléenne est appelée "condition d'évolution" de la transition xi. Elle est notée CEi(K) de la même manière que pour le Grafcet :

$$CEi(K) = fi(Y).xi \quad \text{"Eq-14"}$$

L'ensemble des CEi(K) est noté CE_K.

A partir de cette expression, les règles de franchissement d'une transition deviennent:

Règle 2 : Une transition xi est tirable à l'instant K si sa condition d'évolution CEi(K) associée est vraie à l'instant K.

Règle 3 : Le tir d'une transition provoque l'évolution du marquage des places du réseau. Le nouveau marquage obtenu est tel que:

$\forall$ yj place amont à xi, $yj(K+1) = yj(K) - CEi(K)$

$\forall$ yl place aval à xi, $yl(K+1) = yl(K) + CEi(K)$

De manière générale:

$$yj(K+1) = yj(K) + \sum_{i=1}^{p} Cji \,.\, CEi(K) \quad \text{"Eq-15"}$$

II-5-2-2) Expression matricielle.

Comme pour le Grafcet, les expressions "Eq-14" et "Eq-15" peuvent être exprimées sous forme matricielle:

$$CE_K = G(Y_K).\ X_K \quad \text{"Eq-16"}$$

où $G(Y_K)$ est une matrice dont les termes gij sont des variables représentant les conditions de franchissabilité d'une transition.

$G(Y_K)$ est telle que: $g_{ii} = f_i(Y_K)$

et

$g_{ij} = 0$ si $i \neq j$

- de même :

$$Y_{K+1} = Y_K + C.\ CE_K \quad \text{"Eq-17"}$$

où C est la matrice d'incidence

$$\begin{pmatrix} y1 \\ y2 \\ . \\ yq \end{pmatrix}_{K+1} = \begin{pmatrix} y1 \\ y2 \\ . \\ yq \end{pmatrix}_K + C.\begin{pmatrix} CE\ 1 \\ CE\ 2 \\ . \\ CE\ p \end{pmatrix}_K \quad \text{"Eq-18"}$$

Exemple : Soit le R.D.P.I.C. suivant

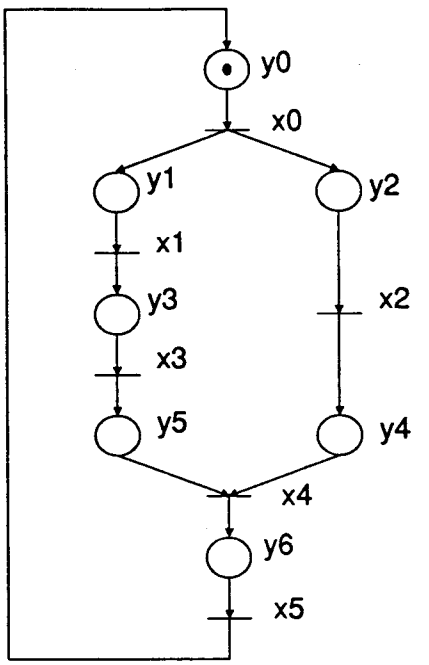

FIG. II-12

Sa matrice d'incidence sera :

$$C = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

$$\begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,2 \\ CE\,3 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & y_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & y_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & y_4.y_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & y_6 \end{pmatrix}_K \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix}_K$$

L'évolution du R.D.P.I.C. se traduit par l'expression:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}_{K+1} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \end{pmatrix}_K + \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,2 \\ CE\,3 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K$$

II-5-3) *Partition du modèle de description*

L'ensemble des équations de récurrence précédemment défini peut être partitionné sur la base de choix de regroupement des événements et opérations [TAN 88] en associant à chaque sous machine, les états et les entrées correspondants.

En suivant une démarche analogue à celle utilisée pour le Grafcet (Cf. II-5-3-2), la matrice d'incidence C peut se décomposer en un ensemble de n sous matrices telle que :

$$C = C^1 + C^2 + C^3 + ... + C^n$$

L'équation d'état du système $Y_{K+1} = Y_K + C.CE_K$ devient pour un processus décomposer en n sous machines:

$$Q^1{}_{K+1} + Q^2{}_{K+1} + ... + Q^n{}_{K+1} = Q^1{}_K + Q^2{}_K + ... + Q^n{}_K + (C^1 + C^2 + ... + C^n).CE_K$$

Chaque vecteur $Q^i{}_K$ est composé des variables associées à l'ensemble des places appartenant à la sous machine Mi, les données relatives aux autres sous machines sont remplacées par un zéro logique.

Pour chaque sous machine, l'équation d'état prend la forme suivante :

$$Q^i{}_{K+1} = Q^i{}_K + C^i.CE_K$$

Remarque : L'évolution de chaque sous machine dépend du vecteur condition d'évolution CE_K associé aux processus global.

Chaque sous machine élabore une partie $CE^i{}_K$du vecteur CE_K.
Le vecteur condition d'évolution s'écrit alors :

$$CE_K = CE^1{}_K + CE^2{}_K + ... + CE^n{}_K$$

Comme pour le Grafcet, la notion de condition d'évolution locale est également utilisée.

Pour un système partitionné en n sous machines, une condition d'évolution globale associée à une transition xi est égale au produit des conditions d'évolution locale de l'ensemble des sous machines associées à cette même transition

$$CEi(K) = \prod_{m=1}^{n} CEi^m(K)$$

L'équation d'état d'une sous machine i s'écrit alors :

$$Q^i_{K+1} = Q^i_K + C^i.CE_K$$

$$CE^i_K = G^i[Q^i_K].X^i_K \qquad \text{"Eq-19"}$$

Pour l'exemple précédent, le partitionnement donne, en réduisant la taille de la matrice C^i par suppression des lignes qui correspondent aux places n'appartenant pas au sous processus Mi :

* Q^0 :

$$Q^0{}_{K+1} = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_{K+1} = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_K + \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,2 \\ CE\,3 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_K + \begin{pmatrix} -1 & 0 & 1 \\ 0 & 1 & -1 \end{pmatrix} \begin{pmatrix} CE\,0 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,0 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y0 & 0 \\ 0 & y6 \end{pmatrix}_K \cdot \begin{pmatrix} x0 \\ x5 \end{pmatrix}_K$$

* Q^1 :

$$Q^1{}_{K+1} = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_{K+1} = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_K + \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,2 \\ CE\,3 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_K + \begin{pmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,3 \\ CE\,4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,1 \\ CE\,3 \\ CE\,4^1 \end{pmatrix}_K = \begin{pmatrix} y1 & 0 & 0 \\ 0 & y3 & 0 \\ 0 & 0 & y5 \end{pmatrix}_K \cdot \begin{pmatrix} x1 \\ x3 \\ x4 \end{pmatrix}_K$$

* Q^2 :

$$Q^2{}_{K+1} = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_{K+1} = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_K + \begin{pmatrix} 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,1 \\ CE\,2 \\ CE\,3 \\ CE\,4 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_K + \begin{pmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,2 \\ CE\,4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,2 \\ CE\,4^1 \end{pmatrix}_K = \begin{pmatrix} y2 & 0 \\ 0 & y4 \end{pmatrix}_K \cdot \begin{pmatrix} x2 \\ x4 \end{pmatrix}_K$$

où $CE4(K) = CE4^1(K)\ .\ CE4^2(K)$

II-5-3-1) Minimisation

En observant les conditions d'évolution au niveau d'une sous machine, on distingue trois types :

- les conditions d'évolution internes à la sous machine i considérée ($CE^i int_K$). Elles ne font varier que l'état interne propre à la sous machine i ;

- les conditions d'évolution d'interaction entre la sous machine i et les autres sous machines ($CE^i ips_K$) ;

- Les conditions d'évolution externes à la sous machine i ($CE^i ext_K$), qui n'influencent pas son comportement propre.

Le vecteur CE_K peut donc s'écrire, pour une sous machine i, sous la forme d'une somme logique de vecteurs :

$$CE_K = CE^i int_K + CE^i ips_K + CE^i ext_K$$

En fonction de ces trois types de conditions d'évolution, l'équation de récurrence "Eq-19" peut se décomposer sous la forme suivante:

$$Q^i_{K+1} = Q^i_K + C^i.CE^i int_K + C^i.CE^i ips_K + C^i.CE^i ext_K$$

où le produit $C^i.CE^i ext_K$ décrit un comportement externe au sous processus considéré qui n'influence pas son propre comportement. Ce produit est donc nécessairement nul.

Autrement dit :

$$Q^i_{K+1} = Q^i_K + C^i.CE^i int_K + C^i.CE^i ips_K$$

ce qui peut s'exprimer en éliminant dans les matrices C^i et les vecteurs CE les lignes et les colonnes n'intervenant pas dans le produit :

$$Q^i_{K+1} = Q^i_K + C^i Int.CE^i int_K + C^i Ips.CE^i ips_K \qquad \text{"Eq-20"}$$

Comportement interne — Comportement inter-processus

L'exemple précédent donne :

* Q^0 :

$$Q^0{}_{K+1} = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_{K+1} = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_K + \begin{pmatrix} 1 \\ -1 \end{pmatrix} . CE5(K) + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,0 \\ CE\,5 \end{pmatrix}_K = \begin{pmatrix} y0 & 0 \\ 0 & y6 \end{pmatrix}_K \cdot \begin{pmatrix} x0 \\ x5 \end{pmatrix}_K$$

* Q^1 :

$$Q^1{}_{K+1} = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_{K+1} = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_K + \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} CE1 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,1 \\ CE\,3 \\ CE\,4^1 \end{pmatrix}_K = \begin{pmatrix} y1 & 0 & 0 \\ 0 & y3 & 0 \\ 0 & 0 & y5 \end{pmatrix}_K \cdot \begin{pmatrix} x1 \\ x3 \\ x4 \end{pmatrix}_K$$

* Q^2 :

$$Q^2{}_{K+1} = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_{K+1} = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_K + \begin{pmatrix} -1 \\ 1 \end{pmatrix} . CE2(K) + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} CE\,0 \\ CE\,4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE\,2 \\ CE\,4^1 \end{pmatrix}_K = \begin{pmatrix} y2 & 0 \\ 0 & y4 \end{pmatrix}_K \cdot \begin{pmatrix} x2 \\ x4 \end{pmatrix}_K$$

où $CE4_{(K)} = CE4^1{}_{(K)} \,.\, CE4^2(K)$

Pour le cas d'un processus plus complexe, la quantité minimale d'information **CE**ips$_K$ à faire circuler sur le réseau est décrite par l'ensemble des vecteurs **CE**^{i}ips$_K$. Pour un processus comprenant n sous machines :

$$\mathbf{CE}ips_K = \bigcup_{i=1}^{n} \mathbf{CE}^{i}ips_K \qquad \text{"Eq-21"}$$

Discussion : Comme pour le Grafcet, ce type de représentation du comportement d'un processus global met en évidence deux comportements distincts :

* le comportement "interne" propre à l'évolution du sous processus considéré et dont les variations n'intéressent pas les autres sous processus ;

* le comportement "inter-processus" qui lie l'évolution du sous processus considéré aux autres sous processus par l'intermédiaire d'un vecteur distribué à tous les sous processus.

Le modèle de comportement décrivant le fonctionnement du processus global peut donc être défini par un réseau de Petri tel que :

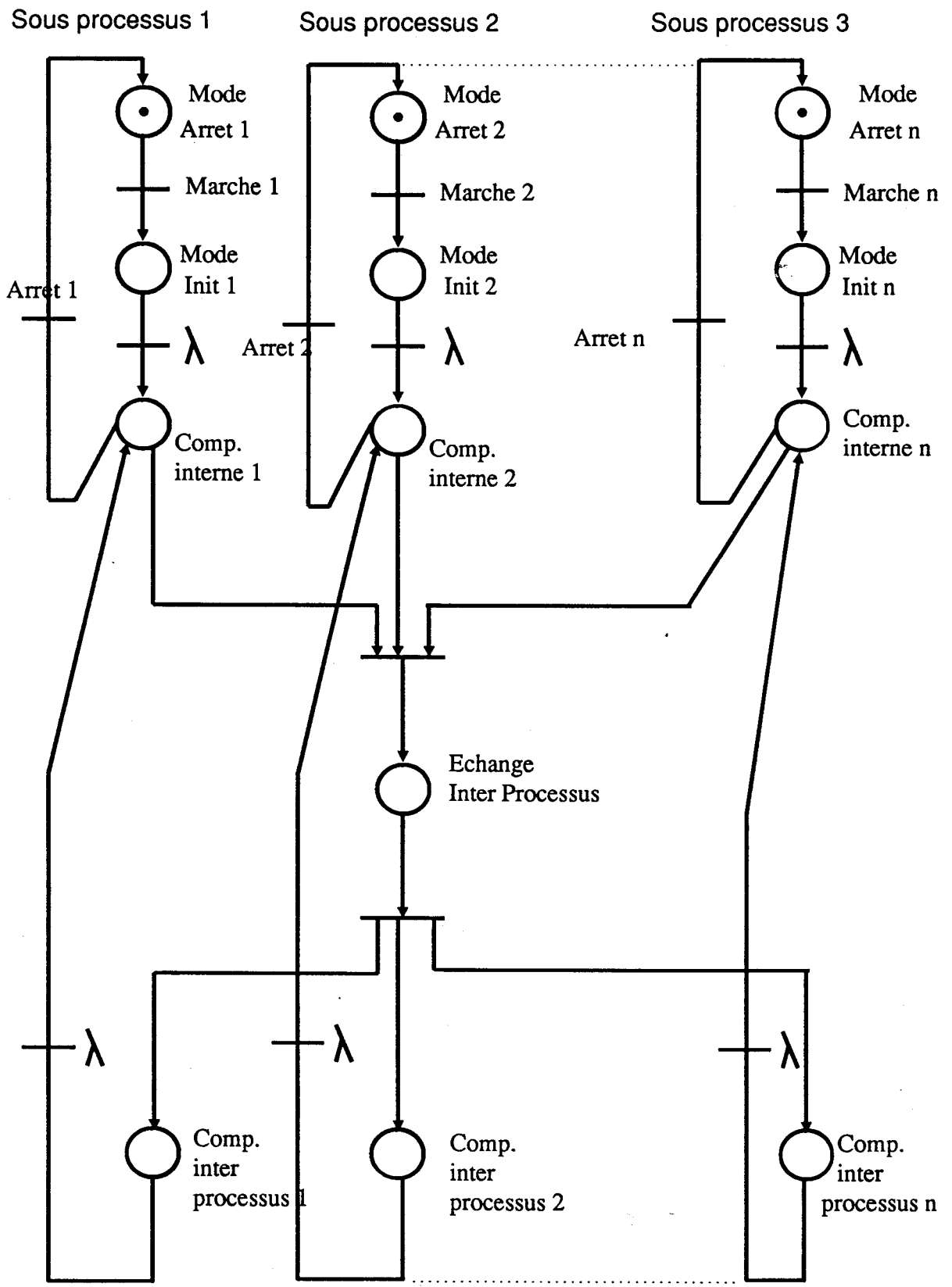

FIG. II-13 : Processus décomposé en n sous machines

II-6) Evolution du graphe de commande

II-6-1) *Evolution interne à un sous processus.*

L'absence de variations sur les ports d'entrées/sorties virtuels d'un sous processus ne nécessite aucune communication avec l'extérieur. Dans ce cas, l'évolution interne est conforme à celle d'un graphe de commande global sur un système monoprocesseur et peut donc être traitée de manière asynchrone des autres sous processus.

II-6-2) *Evolution inter processus*

Lorsque les entrées/sorties virtuelles d'un sous processus subissent des variations, le mécanisme régissant l'évolution interne ne suffit plus à décrire le fonctionnement complet de l'ensemble. Il faut prévoir dans ce cas la gestion du passage de l'activité d'un sous processus vers d'autres sous processus. Le transfert de l'activité repose sur la diffusion entre les différents sous processus reliés par un moyen de communication, de la franchissabilité de la transition frontière à franchir.

Sans préjuger de la nature physique du réseau de communication, il apparaît que le protocole de transfert de l'activité entre sous processus se décompose en deux phases :

- diffusion des conditions d'évolution liées aux transitions frontières à franchir ;
- évolution du marquage au niveau de chaque sous processus.

A priori, la mise en oeuvre de ces deux phases est dépendante de la structure logique du réseau. Dans la suite de l'étude (Cf. chap. III et IV), le protocole adopté pour le transfert de l'activité entre sous processus utilise une technique de communication en anneau logique avec passage de jeton.
Dans cette hypothèse, le jeton qui tourne en permanence sur l'anneau constitue un vecteur d'information entre les sous processus qui peuvent le charger ou le décharger des conditions d'évolution frontières.
Pour permettre à chaque sous processus d'avoir une connaissance globale de l'état de toutes les transitions frontières franchissables à un instant donné, un transfert complet des conditions d'évolution frontières s'effectue sur deux tours du vecteur d'information :

- un tour de **collecte** des conditions d'évolution frontières actives à cet instant donné ;
- un tour de **distribution** de l'ensemble des conditions d'évolution frontières locales ou globales collectées au niveau de chaque sous processus.

Cette forme de protocole offre l'avantage de permettre à chaque sous processus de reconstituer les conditions d'évolution globales partitionnées en conditions d'évolution locales, en lisant ces dernières dans le même vecteur d'information transmis lors de la distribution.

Pour garder cohérente et déterministe l'évolution globale du modèle, le transfert des conditions d'évolution doit se faire en synchronisant les sous processus et en figeant l'état de leur activité pendant cet échange. Dans ces conditions, le comportement inter processus est traité de manière synchrone par l'ensemble des sous processus.
Pour réaliser la synchronisation des sous processus, l'évolution du comportement de chaque sous processus est arrêtée pendant l'échange d'informations.

CHAPITRE III :

Implantation distribuée

III-1) Introduction

Après la présentation d'une méthode d'implantation centralisée, ce chapitre présente une implantation distribuée utilisant la description du comportement faite au chapitre précédent. Cette implantation nécessite la mise en oeuvre d'un moyen de communication entre les différentes parties du processus distribué pour supporter le comportement "interprocessus".

L'implantation distribuée est dérivée de méthodes d'implantation centralisée existantes [DAV 89], [THE 80] sur lesquelles est greffée la gestion des communications interprocessus.

Hypothèses :

Le développement des raisonnements qui vont suivre, nécessite les hypothèses suivantes :

- l'implantation du modèle partitionné est réalisée sur un ensemble de systèmes monoprocesseurs ;

- les sous processus composant le système à réaliser sont reliés entre eux par un réseau local dont le comportement est supposé parfait (pas d'erreur de transmission).

- l'intervalle de temps séparant deux événements est supposé grand par rapport au temps de cycle de l'automate décrivant le comportement d'un sous processus ;

- la prise en compte des entrées se fait à un instant fixe par rapport à l'exécution du programme ;

- la variation de l'état d'une entrée se fait dans un temps supérieur au temps de cycle de l'automate associé ;

- l'affectation des sorties se fait en fin de cycle programme ;

- la durée d'un échange entre sous processus est courte vis à vis du temps de cycle automate ;

- les algorithmes présentés par la suite ne prennent pas en compte les actions conditionnelles et impulsionnelles.

III-2) Description de l'implantation sur processus centralisé

L'implantation d'un modèle Grafcet ou R.D.P.I.C. sur une structure monoprocesseur consiste à définir un ensemble de deux parties :

* une structure de données (Interprétation algébrique) représentant les spécifications fonctionnelles définies par le concepteur de l'automatisme ;

* un interpréteur de cette structure de données (Interprétation algorithmique). De cet interpréteur dépend le respect des règles d'évolution du modèle.

III-2-1) *Interprétation algébrique.*

Elle consiste à définir une structure de données qui décrit le comportement du modèle à implanter. Cette structure a été définie au chapitre précédent. Elle se présente sous la forme d'un ensemble d'équations :

- équations de récurrence de type mémoires pour le grafcet : $Y_{k+1} = S + R.Y_k$;
- équations d'états pour le R.D.P.I.C. : $Y_{k+1} = Y_k + C.CE$;
- conditions d'évolution.

III-2-2) *Interprétation algorithmique.*

Dans un premier temps, un algorithme simplifié, tiré de la littérature [DAV 89], [THE 80], est utilisé pour effectuer l'implantation. L'exécution d'un modèle par un système programmé se décompose alors en six parties :

- Initialisation
- Acquisition des entrées
- Calcul des conditions d'évolution
- Evolution du marquage
- Calcul des combinatoire local et général
- Affectation des sorties

L'initialisation définit l'état que prennent les variables de l'algorithme au début d'exécution de ce dernier.

L'acquisition des entrées effectue une copie de l'état des entrées dans une zone mémoire du système programmé. On appelle X_K le vecteur image des entrées à l'instant K, où K est le temps dit "événementiel".

Le calcul des conditions d'évolution permet de savoir si le système va évoluer. Si il y a évolution alors on fait évoluer l'activité du modèle et on incrémente le temps événementiel.

L'évolution du marquage permet de faire évoluer l'activité du modèle. L'ensemble des états du modèle est matérialisé dans le système programmé par le vecteur d'état Y_K.

Le combinatoire local calcule l'état de l'image des sorties Z_K en fonction de l'état du système Y_K. **Le combinatoire général** calcule des fonctions logiques particulières.

L'affectation des sorties associe à chaque sortie du système la valeur prise par son image mémoire. Cette image des sorties est représentée par le vecteur Z_K.

Remarque : Le temps événementiel évolue à chaque changement d'état du système, c'est à dire si celui ci est réceptif à au moins une entrée xi_K.
En utilisant la définition de la condition d'évolution donnée au chapitre précédent, un système est dit réceptif à une entrée xi_K si sa condition d'évolution associée est vraie à l'instant K. Il y a évolution si au moins une condition d'évolution est vraie. Ceci est généralement matérialisé en testant la véracité de la somme logique de l'ensemble des conditions d'évolution [THE 80].
La structure de l'algorithme est alors la suivante :

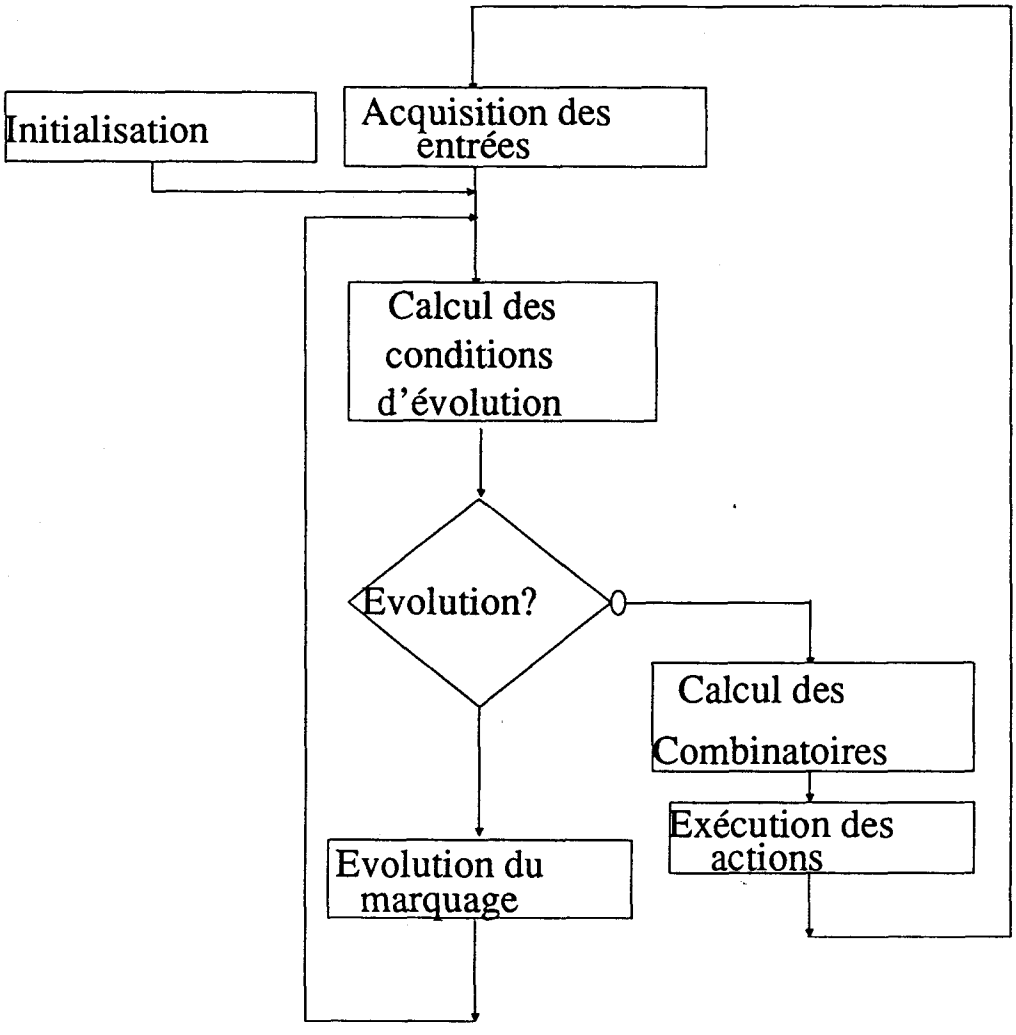

FIG. III-1 : Algorithme de traitement centralisé

Suivant le type de modèle implanté (Grafcet ou R.D.P.I.C.), seul le bloc "évolution du marquage" varie. Ce bloc se résume à évaluer un ensemble d'équations algébriques représentant la structure de données associée au modèle.

Exemple : Implantation d'un Grafcet.
Cet exemple reprend le Grafcet du chapitre précédent :

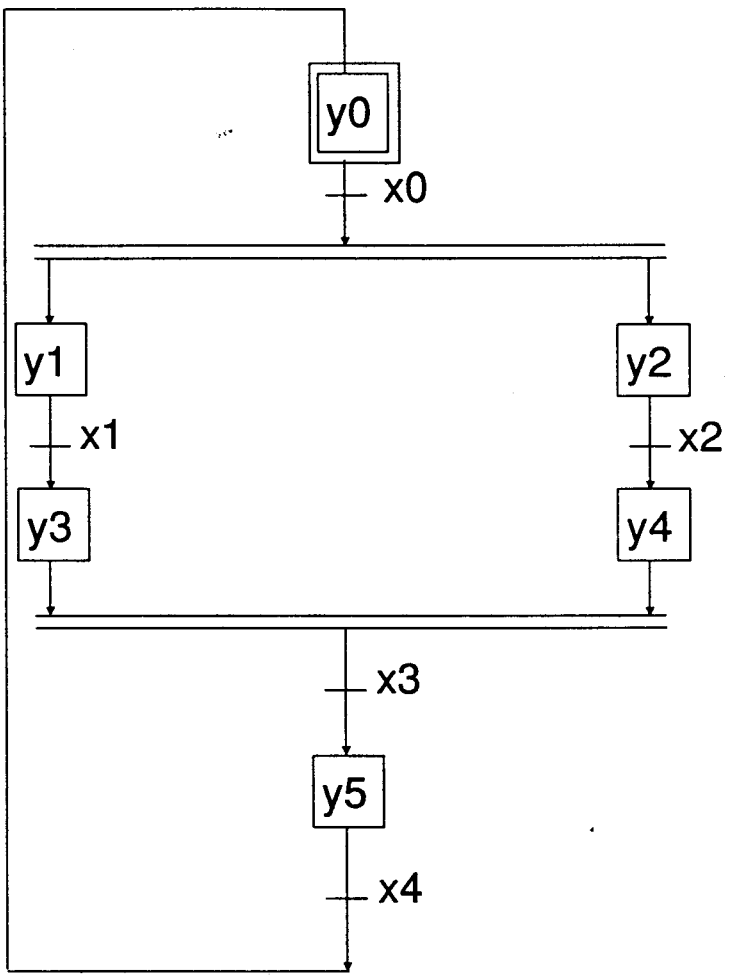

FIG. III-2

Les équations de récurrences de ce Grafcet sont :

$$Y_{K+1} = \begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{pmatrix}_{K+1} = \begin{pmatrix} CE4 \\ CE0 \\ CE0 \\ CE1 \\ CE2 \\ CE3 \end{pmatrix}_{K} + \begin{pmatrix} \overline{CE0} & 0 & 0 & 0 & 0 & 0 \\ 0 & \overline{CE1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \overline{CE2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \overline{CE3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \overline{CE3} & 0 \\ 0 & 0 & 0 & 0 & 0 & \overline{CE4} \end{pmatrix}_{K} \cdot \begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \end{pmatrix}_{K} = S + R \cdot Y_K$$

avec comme équation d'évolution :

$$CEK = \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \end{pmatrix}_{K} = \begin{pmatrix} y0 & 0 & 0 & 0 & 0 \\ 0 & y1 & 0 & 0 & 0 \\ 0 & 0 & y2 & 0 & 0 \\ 0 & 0 & 0 & y3.y4 & 0 \\ 0 & 0 & 0 & 0 & y5 \end{pmatrix}_{K} \cdot \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \end{pmatrix}_{K} = G \cdot X_K$$

d'où l'algorithme d'exécution :

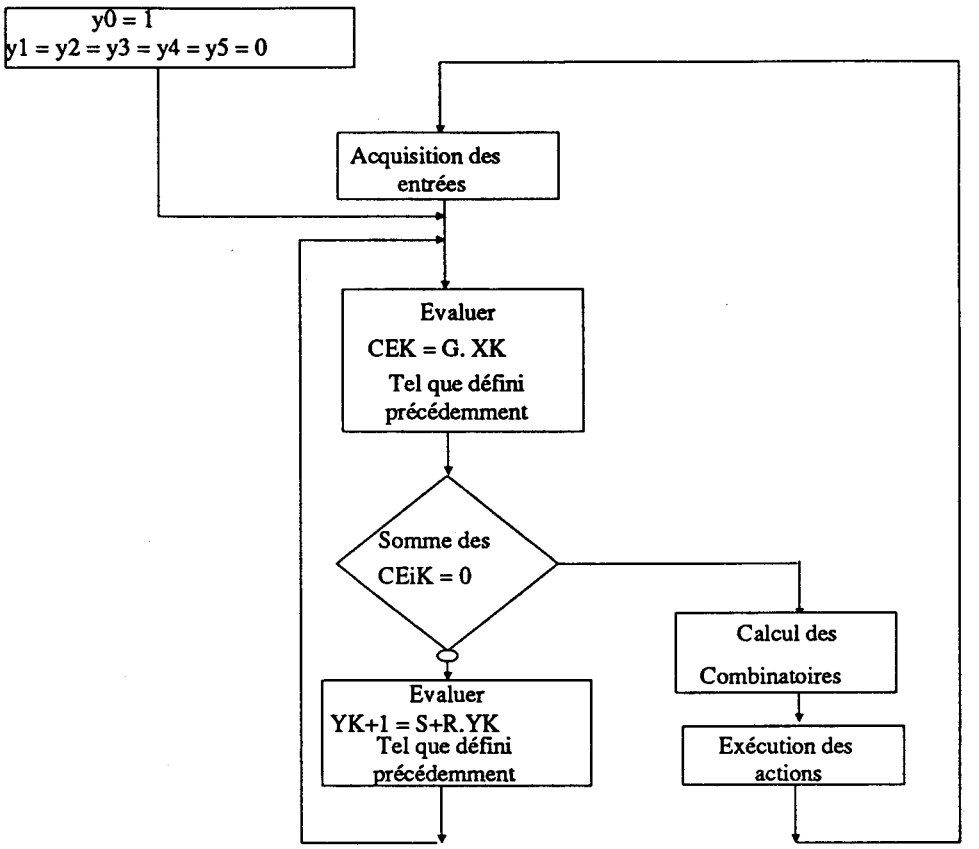

FIG. III-3

Pour un R.D.P.I.C., soit l'exemple suivant :

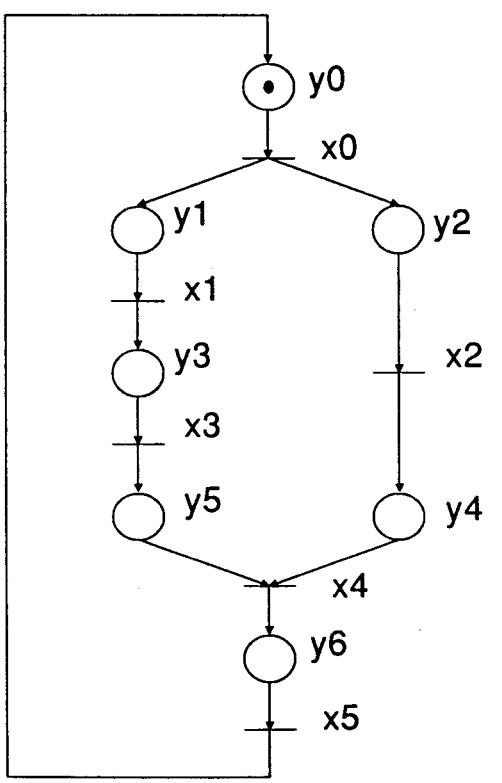

FIG. III-4

D'après le chapitre II, l'équation d'état est la suivante :

$$Y_{K+1} = \begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \end{pmatrix}_{K+1} = \begin{pmatrix} y0 \\ y1 \\ y2 \\ y3 \\ y4 \\ y5 \\ y6 \end{pmatrix}_{K} + \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}_{K} \cdot \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \\ CE5 \end{pmatrix}_{K} = Y_K + C \,.\, CEK$$

avec comme équation d'évolution :

$$CEK = \begin{pmatrix} CE0 \\ CE1 \\ CE2 \\ CE3 \\ CE4 \\ CE5 \end{pmatrix}_{K} = \begin{pmatrix} y0 & 0 & 0 & 0 & 0 & 0 \\ 0 & y1 & 0 & 0 & 0 & 0 \\ 0 & 0 & y2 & 0 & 0 & 0 \\ 0 & 0 & 0 & y3 & 0 & 0 \\ 0 & 0 & 0 & 0 & y4.y5 & 0 \\ 0 & 0 & 0 & 0 & 0 & y6 \end{pmatrix}_{K} \cdot \begin{pmatrix} x0 \\ x1 \\ x2 \\ x3 \\ x4 \\ x5 \end{pmatrix}_{K} = G \,.\, X_K$$

L'algorithme d'exécution est identique au précédent et utilise les équations propres aux R.D.P.I.C.

III-3) Implantation répartie.

L'implantation répartie d'un modèle consiste à définir un ensemble de deux parties pour chaque sous processus :

- une structure de données représentant les spécifications fonctionnelles de la partie du modèle à implanter ;

- un interpréteur de cette structure de données prenant en compte l'aspect coopération entre sous processus.

III-3-1) *Interprétation algébrique.*

Au chapitre précédent, la partition de l'ensemble des équations de récurrence associées au modèle global, a conduit à des sous ensembles d'équations (Cf. Eq 10 et Eq 20). Un sous ensemble représente la structure de données associée à un sous processus. Cette structure se décompose en deux parties :

- l'une décrivant le comportement interne du sous processus ;

- l'autre décrivant son comportement externe qui le lie aux autres sous processus.

Pour prendre en compte ces deux types de comportement, une modification de l'algorithme de traitement global s'impose.

III-3-2) *Interprétation algorithmique.*

Le comportement interne d'une partie de système distribué est indépendant du comportement des autres parties. Il peut donc être traité par un algorithme tel que celui précédemment défini, qui est dupliqué sur chaque site du processus réparti. La gestion des communications entre processus distribués est greffée sur cet algorithme (Cf. FIG. III-5).

L'algorithme traitant le comportement interne d'une partie de processus distribué, a un fonctionnement asynchrone des autres algorithmes.

Le comportement externe d'une partie de processus distribué est dépendant du comportement externe des autres parties. Il consiste à distribuer à chaque sous processus l'état des conditions d'évolution inter processus et doit donc être traité de manière synchrone des autres parties.

De plus, la dépendance entre sous processus se résume à un ensemble de données commun aux différentes parties du processus distribué (Cf. chap. II). Cet ensemble de données prend la forme d'un vecteur d'information qui est échangé entre chaque partie par l'intermédiaire d'un réseau de communication.

L'algorithme de traitement associé à chaque sous processus est alors le suivant :

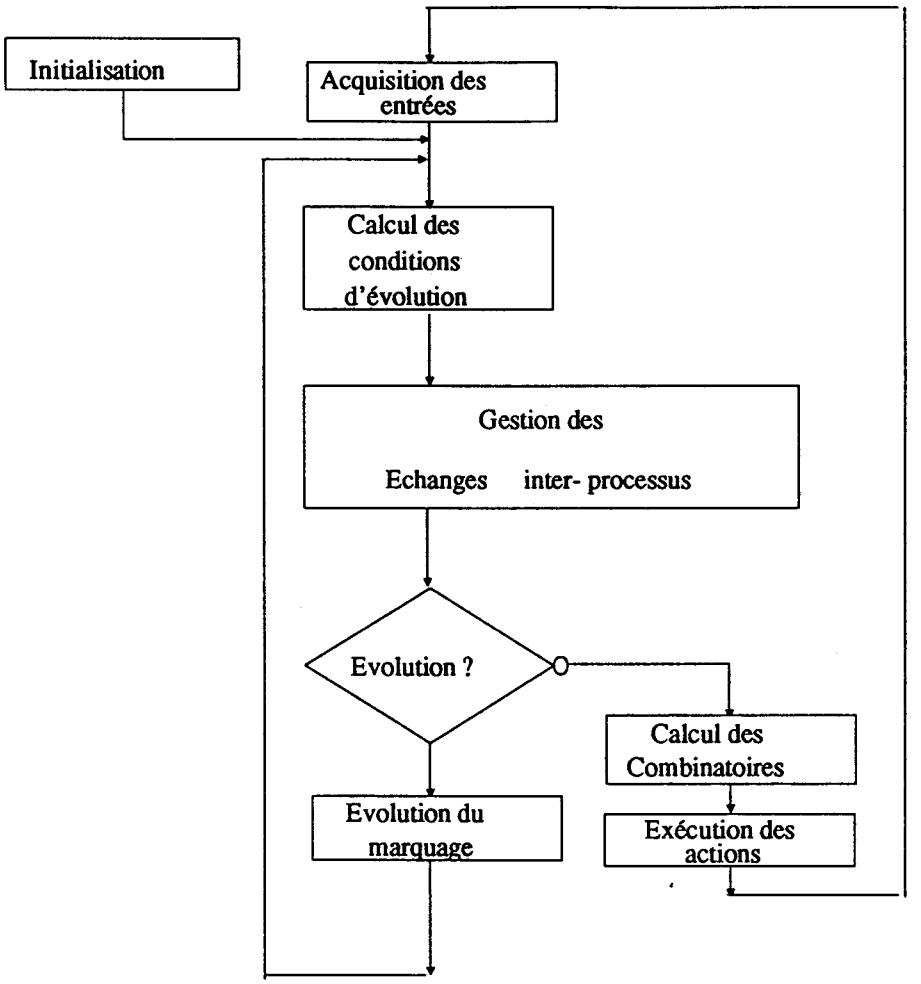

FIG. III-5 : Algorithme de traitement distribué

L'algorithme d'interprétation ci-dessus conserve la même structure que celui précédemment défini, pour un traitement centralisé, sur lequel est greffé la gestion des échanges inter processus.

On peut remarquer que la gestion des échanges inter processus est réalisée juste après le calcul des conditions d'évolution. Ceci se justifie par le fait que les conditions d'évolution permettent d'évaluer si une transition est franchissable et en particulier les transitions frontières..

Dans le cas ou une transition frontière est franchissable, il faut en informer les autres sous processus avant de continuer l'évolution du cycle décrit par l'algorithme de traitement distribué. Pour cela, pendant la diffusion des conditions d'évolution inter processus, l'exécution de l'algorithme de traitement distribué est arrêtée. On dit alors qu'il y a blocage du cycle d'évolution..

Ce blocage permet de garantir qu'après la diffusion des conditions d'évolution inter processus, chaque sous processus reprend son évolution avec la même image des informations échangées.

La suite du développement précise le contenu du bloc "Gestion des échanges Inter processus".

III-3-3) *Echange inter processus*

Les échanges inter processus peuvent être observés sous deux aspects :

- le premier aspect est relatif à un sous processus qui franchit une transition frontière et qui doit donc transmettre l'activité ou le marquage à d'autres sous processus. Ce sous processus se comporte alors comme un "Maître" ;

- le second aspect est relatif aux sous processus attendant ou pas le transfert du marquage. Ces sous processus se comportent comme des "esclaves" dès réception d'une demande de blocage.

III-3-3-1) Comportement "Maître".

Il se traduit par la succession des requêtes réseau suivantes :

- Blocage du cycle d'évolution de l'ensemble des sous processus par transmission d'un ordre de blocage ;

- Collecte sur chaque sous processus de l'ensemble des conditions d'évolution frontières validées (y compris celles du sous processus considéré) ;

- Distribution à chaque sous processus de l'ensemble des conditions d'évolution collectées ;

- Déblocage du cycle d'évolution de l'ensemble des sous processus par transmission d'un ordre de déblocage ;

III-3-3-2) Comportement "Esclave".

C'est un comportement qui peut être qualifié de passif et qui consiste à exécuter les requêtes émises par le "Maître":

- blocage du cycle d'évolution sur réception de l'ordre blocage ;

- insertion des conditions d'évolution frontières dans la trame de collecte ;

- lecture de l'ensemble des conditions d'évolution dans la trame de distribution ;

- déblocage du cycle d'évolution sur réception de l'ordre de déblocage.

L'évolution du cycle décrit par l'algorithme traitement distribué d'un sous processus qui reçoit une demande de blocage, se bloque dés qu'il exécute la gestion des échanges inter processus.

Remarque : Cette succession de requêtes permet de garantir la synchronisation entre les différents sous processus du système distribué. De plus, la suite de requêtes "Collecte" et "Distribution" offre à chaque sous processus la même information sur l'état de toutes les conditions d'évolution frontières à un instant K donné.

On aboutit à la structure détaillée suivante :

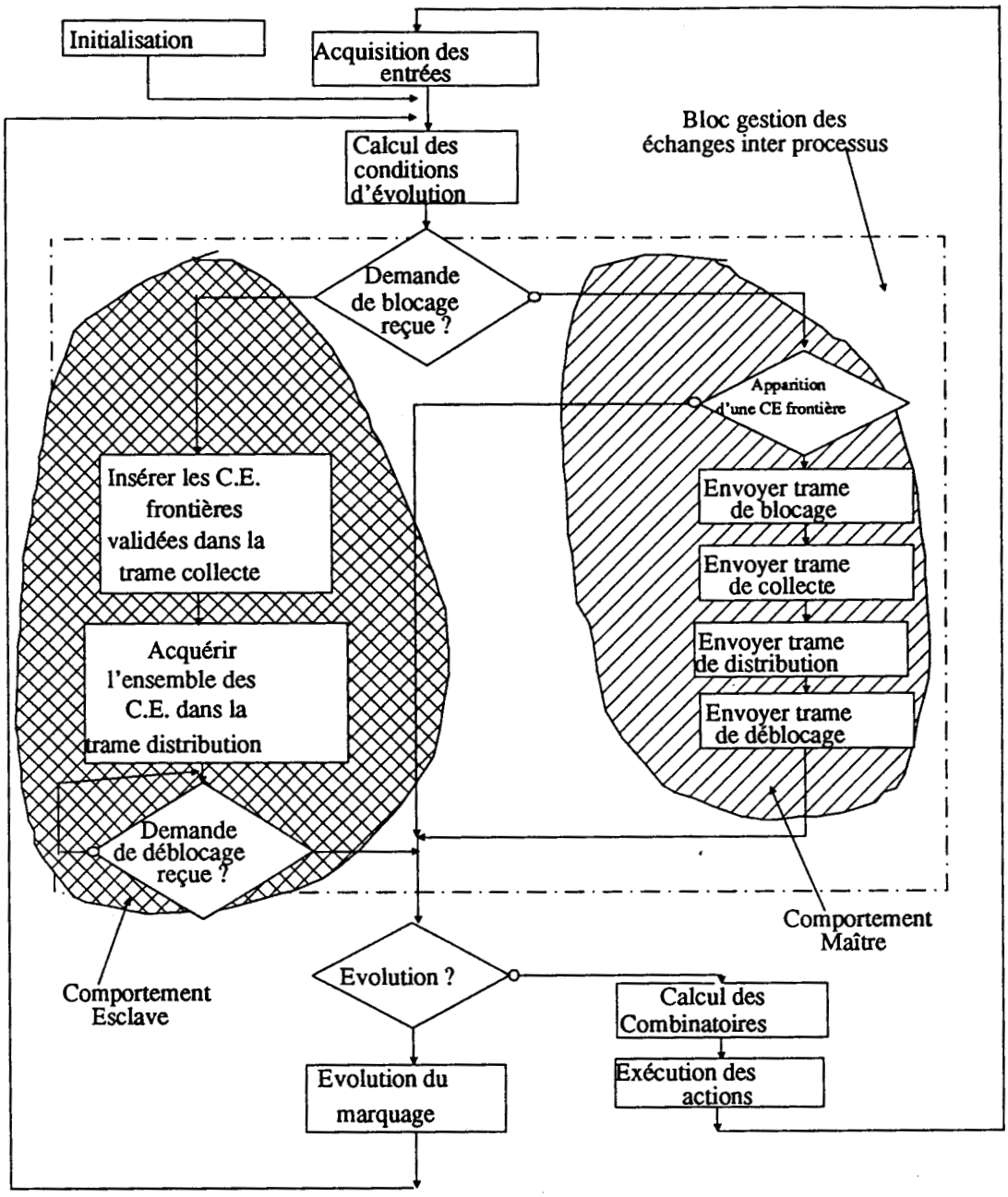

FIG. III-6 : Algorithme et primitives de communication

Remarque :

- la structure de la figure III-6 fait bien apparaître le fait qu'une partie donnée se comporte tantôt comme un Maître tantôt comme un Esclave selon l'évolution du système global ;

- l'avantage de cette structure est qu'elle ne préjuge en rien de la nature du réseau utilisé ;

- l'inconvénient majeur est qu'elle ne permet pas de résoudre le cas ou deux sous processus franchissent simultanément une transition frontière.

Cet inconvénient est résolu, dans la suite de l'exposé, en dédiant au réseau le soin d'attribuer à un sous processus un comportement Maître ou Esclave.

Exemple : Pour le Grafcet précédent, les équations de récurrence du processus partitionné sont :

* Q^0

$$\begin{pmatrix} y_0 \\ y_5 \end{pmatrix}_{K+1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}.CE4(K) + \begin{pmatrix} 0 \\ y_5 \end{pmatrix}_K.\overline{CE4(K)} + \begin{pmatrix} 0\,0 \\ 0\,1 \end{pmatrix}\begin{pmatrix} CE0 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} y_0\,0 \\ 0\,0 \end{pmatrix}_K.\begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

$$\begin{pmatrix} CE0 \\ CE4 \end{pmatrix}_K = \begin{pmatrix} y_0 & 0 \\ 0 & y_5 \end{pmatrix}_K.\begin{pmatrix} x_0 \\ x_4 \end{pmatrix}_K$$

* Q^1

$$\begin{pmatrix} y_1 \\ y_3 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.CE1(K) + \begin{pmatrix} y_1 \\ 0 \end{pmatrix}_K.\overline{CE1(K)} + \begin{pmatrix} 1\,0 \\ 0\,0 \end{pmatrix}\begin{pmatrix} CE_0 \\ CE_3 \end{pmatrix}_K + \begin{pmatrix} 0\,0 \\ 0\,y_4 \end{pmatrix}_K.\begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

$$\begin{pmatrix} CE1 \\ CE3^1 \end{pmatrix}_K = \begin{pmatrix} y_1 & 0 \\ 0 & y_3 \end{pmatrix}_K.\begin{pmatrix} x_1 \\ x_3 \end{pmatrix}_K$$

* Q^2

$$\begin{pmatrix} y_2 \\ y_4 \end{pmatrix}_{K+1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.CE2(K) + \begin{pmatrix} y_2 \\ 0 \end{pmatrix}_K.\overline{CE2(K)} + \begin{pmatrix} 1\,0 \\ 0\,0 \end{pmatrix}\begin{pmatrix} CE0 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} 0\,0 \\ 0\,y_4 \end{pmatrix}_K.\begin{pmatrix} \overline{CE0} \\ \overline{CE3} \end{pmatrix}_K$$

$$\begin{pmatrix} CE2 \\ CE3^2 \end{pmatrix}_K = \begin{pmatrix} y_2 & 0 \\ 0 & y_4 \end{pmatrix}_K.\begin{pmatrix} x_2 \\ x_3 \end{pmatrix}_K$$

où $CE3 = CE3^1.CE3^2$

La partie Q^0 est associée à l'algorithme d'exécution suivant :

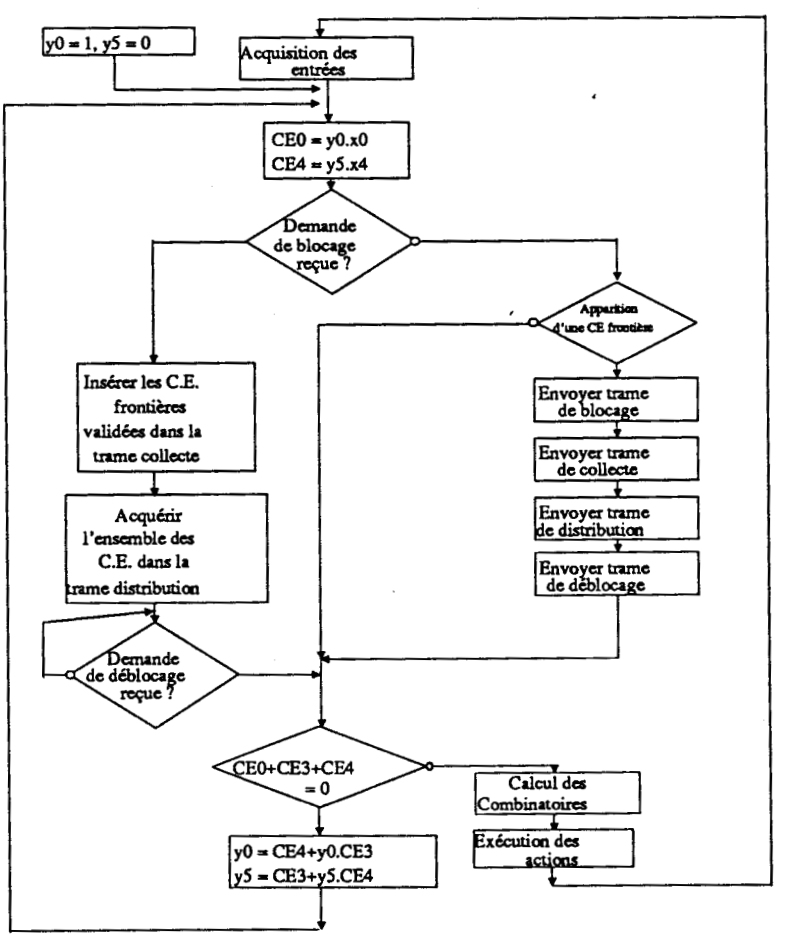

FIG. III-7

Pour les parties Q^1 et Q^0, les structures d'algorithme sont en tout points analogues.

Pour le R.D.P.I.C. précédent, la structure de données partitionnée en trois parties est :

* Q^0 :

$$\begin{pmatrix} y0 \\ y6 \end{pmatrix}_{K+1} = \begin{pmatrix} y0 \\ y6 \end{pmatrix}_K + \begin{pmatrix} 1 \\ -1 \end{pmatrix} . CE5(K) + \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} CE0 \\ CE4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE0 \\ CE5 \end{pmatrix}_K = \begin{pmatrix} y0 & 0 \\ 0 & y6 \end{pmatrix}_K . \begin{pmatrix} x0 \\ x5 \end{pmatrix}_K$$

* Q^1 :

$$\begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_{K+1} = \begin{pmatrix} y1 \\ y3 \\ y5 \end{pmatrix}_K + \begin{pmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} CE1 \\ CE3 \end{pmatrix}_K + \begin{pmatrix} 1 & 0 \\ 0 & 0 \\ 0 & -1 \end{pmatrix} . \begin{pmatrix} CE0 \\ CE4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE1 \\ CE3 \\ CE4^1 \end{pmatrix}_K = \begin{pmatrix} y1 & 0 & 0 \\ 0 & y3 & 0 \\ 0 & 0 & y5 \end{pmatrix}_K . \begin{pmatrix} x1 \\ x3 \\ x4 \end{pmatrix}_K$$

* Q^2 :

$$\begin{pmatrix} y2 \\ y4 \end{pmatrix}_{K+1} = \begin{pmatrix} y2 \\ y4 \end{pmatrix}_K + \begin{pmatrix} -1 \\ 1 \end{pmatrix} . CE2(K) + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \begin{pmatrix} CE0 \\ CE4 \end{pmatrix}_K$$

$$\begin{pmatrix} CE2 \\ CE4^1 \end{pmatrix}_K = \begin{pmatrix} y2 & 0 \\ 0 & y4 \end{pmatrix}_K . \begin{pmatrix} x2 \\ x4 \end{pmatrix}_K$$

où $CE4(K) = CE4^1(K) . CE4^2(K)$

L'algorithme d'exécution est analogue à celui du Grafcet de l'exemple précédent.

III-4) Architecture générale du système de communication

Pour simplifier la mise en oeuvre de la partie commande et compte tenu du caractère séquentiel de l'exécution des primitives de communication, ces dernières peuvent être implantées sur le réseau lui même.
Dans la suite, l'ensemble des primitives de communication réseau introduites dans l'algorithme d'interprétation a été intégré dans la carte "communicateur" du réseau (Cf. FIG. III-8). Le communicateur se charge d'adopter un comportement Maître ou esclave.
Le comportement Maître consiste alors, pour un communicateur, à contrôler l'enchaînement des primitives de communication réseau et le comportement Esclave à répondre aux requêtes reçues du communicateur Maître.
La solution choisie repose sur l'utilisation d'un réseau de type anneau physique avec passage de jeton :

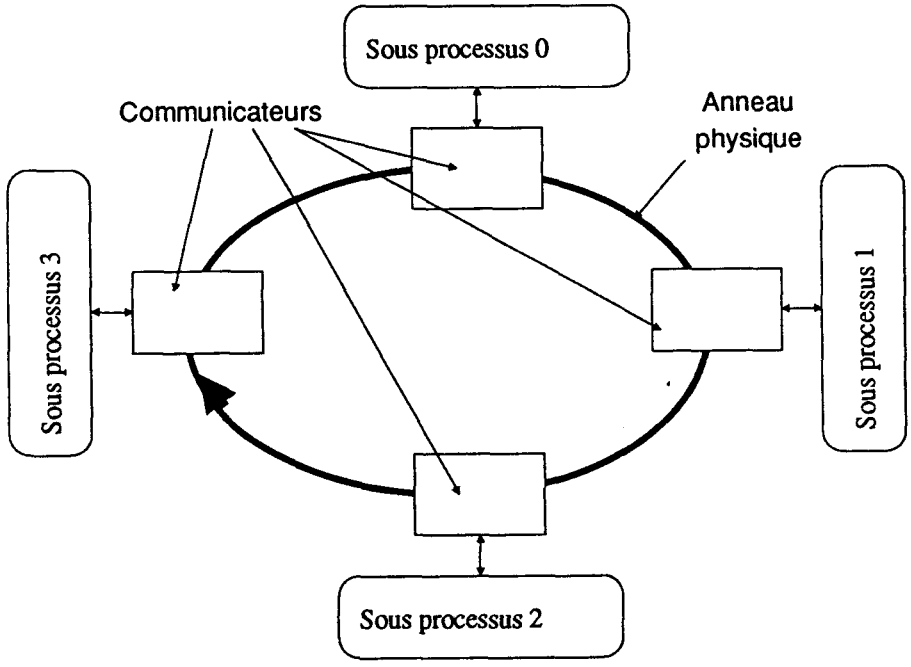

FIG. III-8 : Réseau en anneau

Le communicateur fonctionne selon les principes suivants :

- lorsqu'une condition d'évolution frontière apparaît, le sous processus en informe son communicateur qui devient "Maître" après réception du jeton ;

- ce dernier exécute alors une procédure complète de dialogue entre sous processus (Blocage - Collecte - Distribution - Déblocage) où il réunit en un seul vecteur l'ensemble des conditions d'évolution frontières validées et déposées par chaque sous processus dans son communicateur ;

- à la fin de cette procédure, chaque communicateur transmet à son sous processus le vecteur condition d'évolution reçu du communicateur ;

- ceci permet au sous processus de reprendre son évolution, après réception de la trame de déblocage, en tenant compte d'états physiquement distribués.

Remarque : L'avantage d'un tel support physique est de permettre la résolution des cas de non déterminisme présents dans la figure III-6 (inconvénient majeur).
En effet, pour devenir Maître, un sous processus a besoin de recevoir le jeton. Ce jeton symbolise le fait qu'aucun autre poste n'est Maître à cet instant.
Dans le cas où un sous processus désirant devenir Maître reçoit à la place d'un jeton une trame de blocage, il passe immédiatement en mode Esclave.

Le modèle de comportement de l'ensemble des sous processus prend alors la forme suivante :

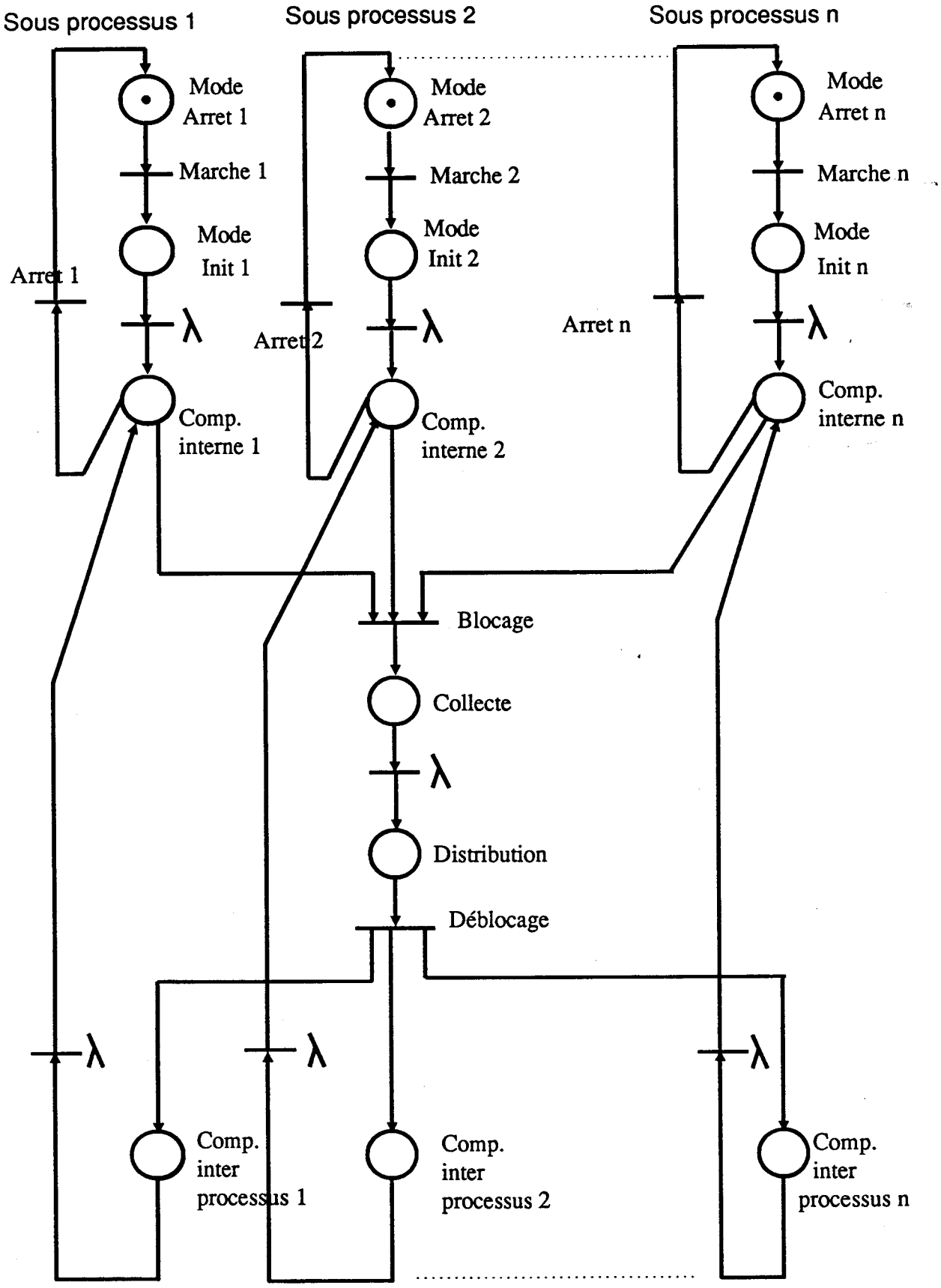

FIG. III-9 : Comportement global des processus

L'adaptation du sous processus à son communicateur est obtenue en définissant un nouvel algorithme d'implantation distribué dont la structure est la suivante :

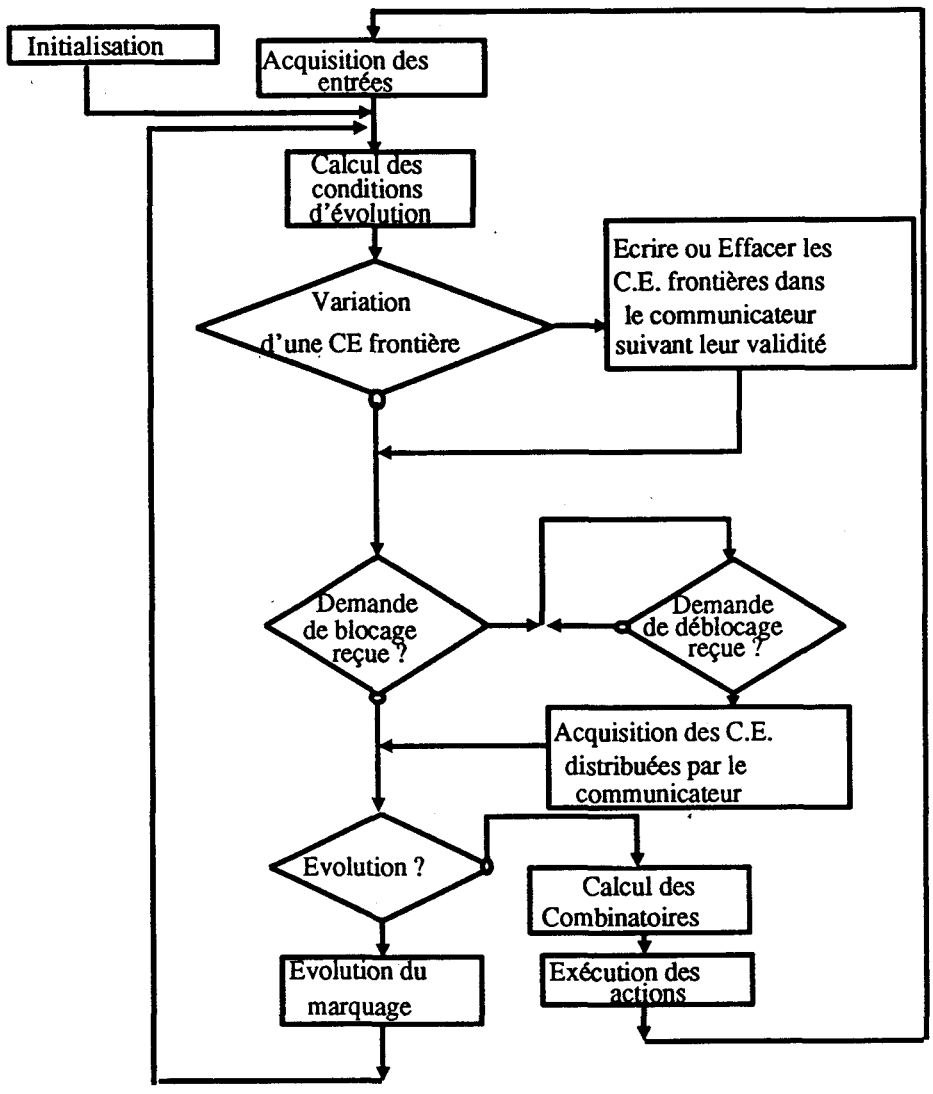

FIG. III-10 : Algorithme d'adaptation au communicateur

Dans cet algorithme on constate que les différents dialogues entre sous processus et communicateur se résument à quatre primitives ou services :

- écriture des conditions d'évolution frontières validées à l'instant K dans le communicateur ;

- effacement des conditions d'évolution frontières qui viennent d'être dévalidées dans le communicateur ;

- transfert du communicateur vers le sous processus, à la fin de chaque procédure de communication réseau, de l'ensemble des conditions d'évolution distribuées ;

- dialogue en appel - réponse pour effectuer un blocage ou un déblocage du cycle d'évolution.

III-4-1) *Discussion.*

La nécessité d'utiliser le "blocage" du cycle d'évolution d'un sous processus lors d'un échange inter processus est discutable :

Le principal avantage du blocage est de rendre parfaitement déterministe l'évolution inter processus, étant donné que chaque sous processus reprend ensuite son évolution avec la même image des conditions d'évolution.

Ses principaux inconvénients sont d'augmenter le temps nécessaire aux échanges et de bloquer l'évolution de sous processus qui ne sont pas concernés par l'échange.

Dans la pratique, les constructeurs de réseaux locaux industriels dédiés au contrôle de processus n'adoptent pas cette technique de blocage (Cf. I-8-4). En général, Ils supposent l'évolution entre sous processus déterministe grâce à une vitesse d'échange très grande (Réseau très rapide) par rapport au temps de cycle automate (Cf. Chap. I).

La solution préconisée pour résoudre ce problème est de ne bloquer l'évolution des sous processus que si ils sont concernés par l'échange en cours. En effet, lors d'un échange, seuls quelques sous processus évoluent suite à cet échange.

Pour ne bloquer que les sous processus concernés par l'échange, on peut définir d'une politique d'échange qui consiste à définir lors de la partition du modèle global, quels sont les liens de coopération entre sous processus. Les liens sont matérialisés par les conditions d'évolution aux quelles ont peut associer une politique donnée.
Le blocage des cycles d'évolution s'effectue alors en fonction de la politique d'échange en cours.

A titre d'exemple, soit le Grafcet suivant décomposé en 6 sous processus :

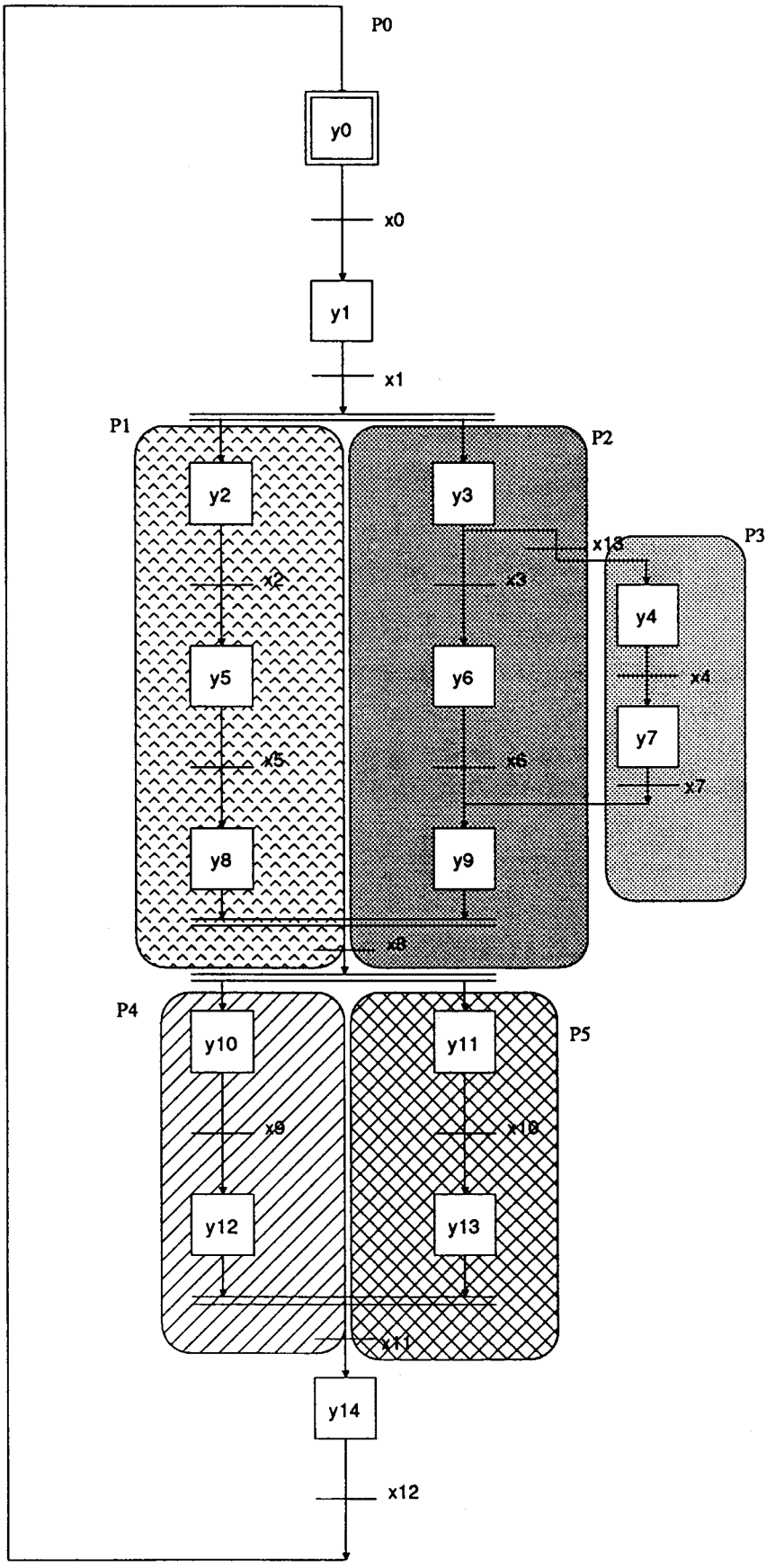

FIG. III-11

L'ensemble des équations d'état pour chaque sous processus est :

$$P_0 = \begin{pmatrix} y0 \\ y1 \\ y14 \end{pmatrix} = \begin{pmatrix} 01 \\ 10 \\ 00 \end{pmatrix} \cdot \begin{pmatrix} CE0 \\ CE12 \end{pmatrix} + \begin{pmatrix} y0 & 0 \\ 0 & 0 \\ 0 & y14 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE0} \\ \overline{CE12} \end{pmatrix} + \begin{pmatrix} 00 \\ 00 \\ 01 \end{pmatrix} \cdot \begin{pmatrix} CE1 \\ CE11 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ y1 & 0 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE1} \\ \overline{CE11} \end{pmatrix}$$

$$P_1 = \begin{pmatrix} y2 \\ y5 \\ y8 \end{pmatrix} = \begin{pmatrix} 00 \\ 10 \\ 01 \end{pmatrix} \cdot \begin{pmatrix} CE2 \\ CE5 \end{pmatrix} + \begin{pmatrix} y2 & 0 \\ 0 & y5 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE2} \\ \overline{CE5} \end{pmatrix} + \begin{pmatrix} 10 \\ 00 \\ 00 \end{pmatrix} \cdot \begin{pmatrix} CE1 \\ CE8 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 0 & y8 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE1} \\ \overline{CE8} \end{pmatrix}$$

$$P_2 = \begin{pmatrix} y3 \\ y6 \\ y9 \end{pmatrix} = \begin{pmatrix} 00 \\ 10 \\ 01 \end{pmatrix} \cdot \begin{pmatrix} CE3 \\ CE6 \end{pmatrix} + \begin{pmatrix} y3 & 0 \\ 0 & y6 \\ 0 & 0 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE3} \\ \overline{CE6} \end{pmatrix} + \begin{pmatrix} 1000 \\ 0000 \\ 0100 \end{pmatrix} \cdot \begin{pmatrix} CE1 \\ CE7 \\ CE8 \\ CE13 \end{pmatrix} + \begin{pmatrix} 00\,0\,y3 \\ 00\,0\,0 \\ 00y9\,0 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE1} \\ \overline{CE7} \\ \overline{CE8} \\ \overline{CE13} \end{pmatrix}$$

$$P_3 = \begin{pmatrix} y4 \\ y7 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} . CE4 + \begin{pmatrix} y4 \\ 0 \end{pmatrix} . \overline{CE4} + \begin{pmatrix} 01 \\ 00 \end{pmatrix} \cdot \begin{pmatrix} CE7 \\ CE13 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ y7 & 0 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE7} \\ \overline{CE13} \end{pmatrix}$$

$$P_4 = \begin{pmatrix} y10 \\ y12 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} . CE9 + \begin{pmatrix} y10 \\ 0 \end{pmatrix} . \overline{CE9} + \begin{pmatrix} 10 \\ 00 \end{pmatrix} \cdot \begin{pmatrix} CE8 \\ CE11 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & y12 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE8} \\ \overline{CE11} \end{pmatrix}$$

$$P_4 = \begin{pmatrix} y11 \\ y13 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} . CE10 + \begin{pmatrix} y11 \\ 0 \end{pmatrix} . \overline{CE10} + \begin{pmatrix} 10 \\ 00 \end{pmatrix} \cdot \begin{pmatrix} CE8 \\ CE11 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & y13 \end{pmatrix} \cdot \begin{pmatrix} \overline{CE8} \\ \overline{CE11} \end{pmatrix}$$

On peut alors définir la matrice des interactions suivantes :

	P0	P1	P2	P3	P4	P5
P0		CE1	CE3		CE11	CE11
P1	CE1		CE3		CE8	CE8
P2	CE1	CE1		CE13 CE7	CE8	CE8
P3			CE13 CE7			
P4	CE11	CE8	CE8			CE8
P5	CE11	CE8	CE8		CE8	

Les politiques de dialogues sont définies en fonctions des interactions :

Politique 1 : P0 $\cap$ P1 $\cap$ P2 = CE1

Politique 2 : P4 $\cap$ P5 $\cap$ P1 $\cap$ P2 = CE8

Politique 3 : P2 $\cap$ P3 = CE7 + CE13

Politique 4 : P0 $\cap$ P4 $\cap$ P5 = CE11

Chaque sous processus est donc concerné par les politiques suivantes :

P0 = 1,4 ; P1 =1,2 ; P2 = 1,2,3 ; P3 = 3 ; P4 = 2,4 ; P5 = 2,4

L'algorithme d'exécution, dont le blocage dépend de la politique en cours, est le suivant :

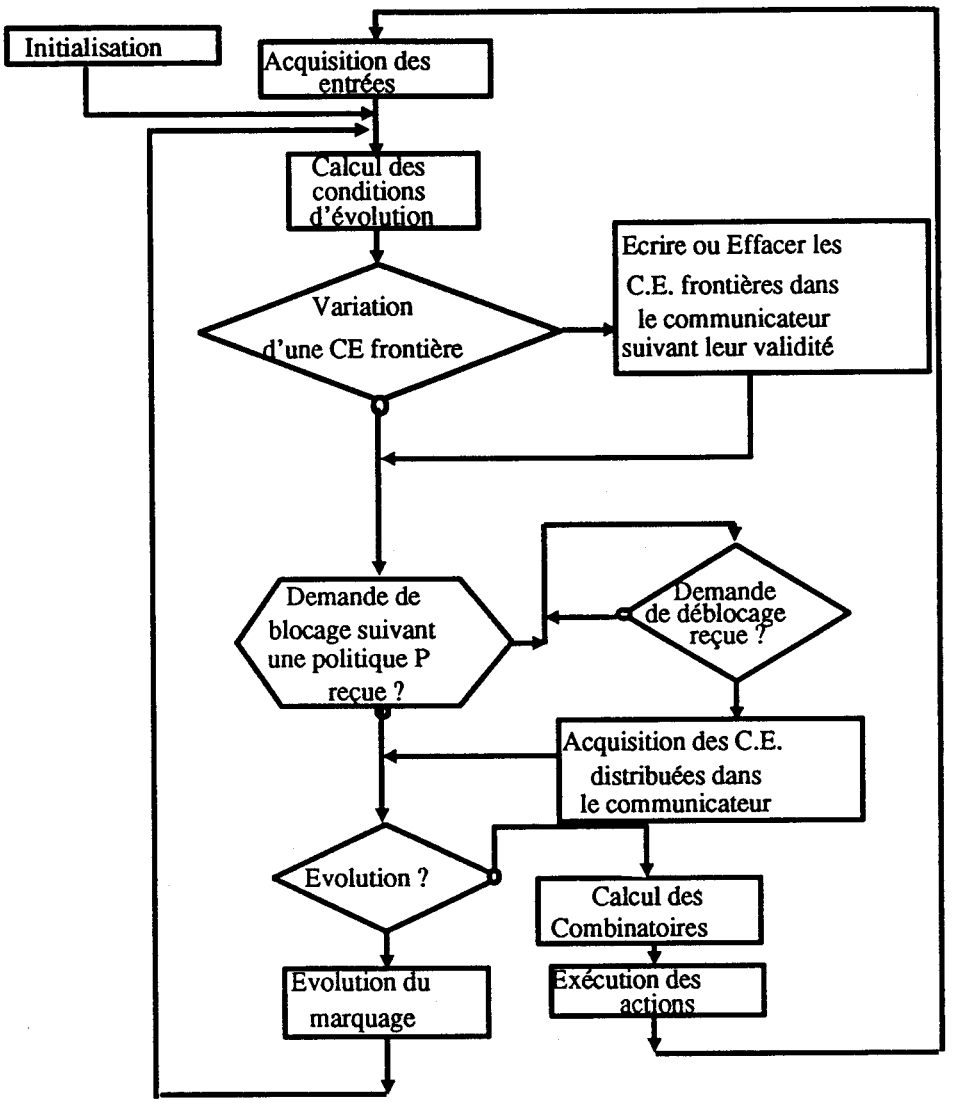

FIG. III-12 :

III-5) **Conclusion.**

Ce chapitre définit la méthode d'implantation retenue pour réaliser une mise en oeuvre distribuée. La solution choisie repose sur la structure de données décrite au chapitre précédent. L'algorithme chargé de traiter la structure de donnée, est défini dans ce chapitre pour supporter deux types de comportements. Il adopte un fonctionnement asynchrone des autres sous processus pour traiter le comportement interne et un fonctionnement synchrone avec les autres sous processus pour échanger les conditions d'évolution inter processus.

Enfin, l'introduction d'un communicateur permet de rendre indépendant le protocole de communication des données échangées et de lui dédier le contrôle du fonctionnement des échanges réseau.

Le chapitre suivant présente l'architecture du réseau utilisé pour mettre en oeuvre la commande distribuée.

CHAPITRE IV :

Le réseau de communication

Ce chapitre est consacré à la réalisation et à la conception du communicateur introduit au chapitre précédent, tant sous l'aspect matériel que sous l'aspect logiciel [SAN 89].

L'objectif de cette réalisation est de dédier le contrôle des échanges entre les différents sous processus à un réseau afin d'alléger l'implantation répartie du processus.

IV-1) **Principe du réseau.**

Tel que précisé au chapitre précédent, la solution adoptée pour permettre la circulation des informations entre communicateurs repose sur l'utilisation d'une topologie en anneau. Chaque communicateur est en relation avec deux autres communicateurs : un en aval, un en amont. L'information transmise se propage circulairement entre tous les communicateurs avant de revenir au communicateur source. Cette topologie permet de garantir à chaque communicateur, l'accès à toutes les informations circulant sur le réseau.

Le protocole d'accès au réseau est basé sur le principe du "droit à la parole" que les communicateurs se passent les uns aux autres sous la forme d'une trame d'un type particulier appelée "Jeton". Un communicateur qui désire envoyer un message vers un autre communicateur ne peut le faire que si il possède ce "droit à la parole" (Jeton)

Dans le cas où aucun communicateur ne désire envoyer de message, les communicateurs se passent séquentiellement le Jeton en attendant que l'un d'eux prenne ce "droit à la parole" pour émettre.

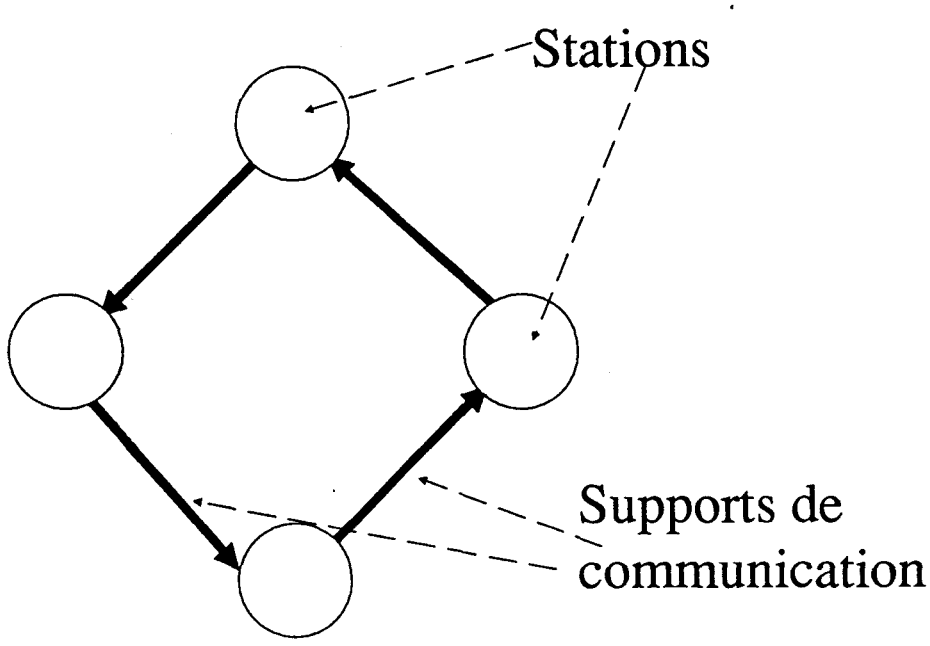

FIG. IV-1 : Aspect du réseau

Etant donné qu'il n'y a qu'un seul Jeton circulant entre les communicateurs, les risques de collisions sont écartés. D'autre part, il convient d'envisager une stratégie de remise en route, notamment dans le cas d'une perte de Jeton ou de toute autre trame.

IV-1-1) *Fonctionnement Maître - Esclave.*

De par la topologie du réseau et le protocole d'accès défini, on peut remarquer que tous les communicateurs sont identiques et qu'il n'y a pas de contrôleur central (Maître) pour le réseau. Il est donc nécessaire que l'un des communicateurs soit chargé de contrôler l'activité du réseau à un instant donné. On dit alors qu'il possède la main et son comportement est qualifié de "Maître". Un communicateur est susceptible de prendre la main s'il désire émettre un message. Il doit alors attendre pour passer en mode Maître de réceptionner un Jeton.

Un communicateur prend la decision d'émettre un message à chaque fois que le sous processus attaché lui transmet de nouvelles conditions d'évolution frontières.

Un communicateur fonctionnant en mode Maître prend en charge l'ensemble des procédures de communication. Il est le seul capable de gérer les erreurs de transmissions et de réinitialiser la procédure le dialogue en cas de perte de trame.

Quand le communicateur Maître a terminé d'envoyer son message, il retransmet le "droit à la parole" aux autres communicateurs.

Un communicateur n'ayant pas la main (mode Esclave) ne peut que réceptionner une trame, y lire ou écrire des informations avant de la retransmettre.

Dans le cas ou un communicateur Esclave détecte une erreur de transmission, il transmet une trame d'un type particulier précisant au communicateur Maître qu'une erreur de transmission s'est produite. Le communicateur Maître prend alors la décision de retransmettre la dernière trame émise.

Dans la suite de ce chapitre, nous présentons, en premier lieu, l'architecture logique du communicateur. Nous donnons, en second lieu, la définition de l'architecture logicielle du communicateur avant de présenter dans la dernière partie du chapitre l'aspect matériel du communicateur.

IV-2) Le communicateur

Le communicateur assure la communication d'une part avec le sous processus et d'autre part avec les autres communicateurs (Cf. FIG. III-8).
Par ailleur, il doit mémoriser un ensemble de données partagées en deux zones :

- une zone destinée à recevoir du sous processus les conditions d'évolution frontières validées ;

- une zone destinée à stocker les informations distribuées sur le réseau.

IV-2-1) *Architecture logique*

La figure IV-2 présente l'architecture logique du communicateur ainsi que son environnement.

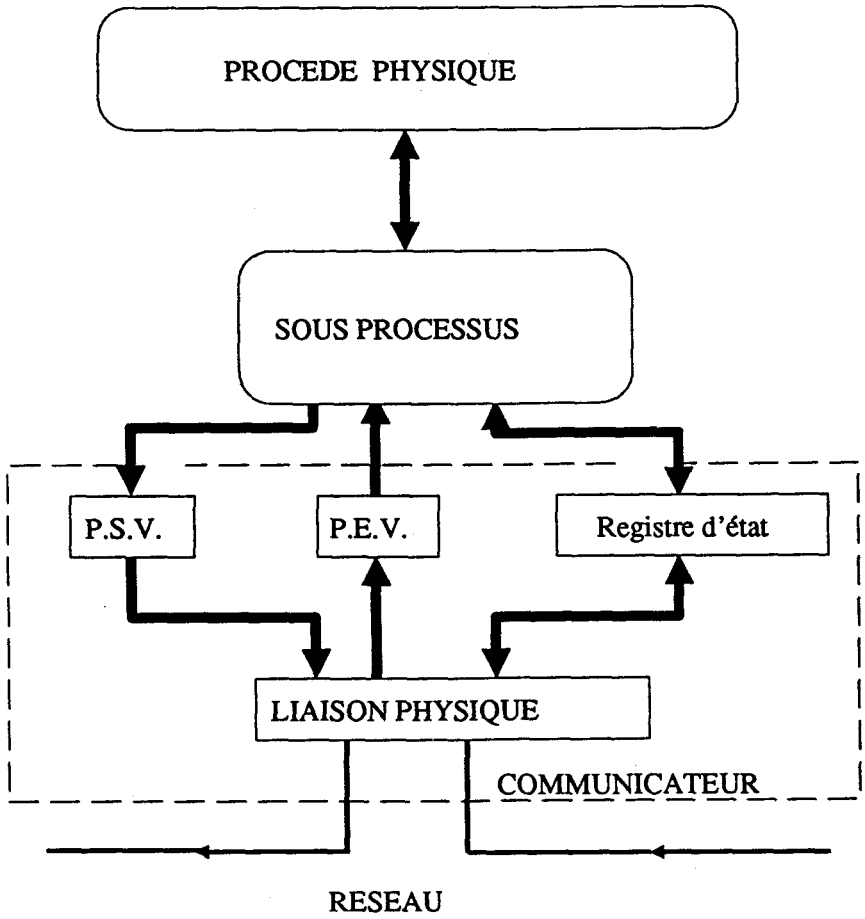

FIG. IV-2 : Le communicateur et son environnement

P.E.V. : Port d'entrées virtuel

P.S.V. : Port de sorties virtuel

Le port d'entrées virtuel (P.E.V.) mémorise les conditions d'évolution émanant d'une communication réseau. Celui ci est mis à jour par le communicateur.

Le port de sortie virtuel (P.S.V.) mémorise les conditions d'évolution frontières d'un sous processus. Celui ci est géré par ce sous processus.

Les variables d'état caractérisent les différents états du communicateur.

IV-2-2) *Les dialogues sous processus - communicateur*

La communication se décompose selon les procédures suivantes :

- le sous processus gère l'écriture ou l'effacement des conditions d'évolution frontières dans le P.S.V. qui est une table située dans la mémoire du communicateur ;

- le communicateur retransmet au sous processus toutes informations qui transitent sur le réseau et qui sont nécessaires à son évolution ;

- le communicateur peut dialoguer en appel-réponse avec le sous processus pour générer un blocage du cycle d'évolution de ce dernier.

IV-2-3) *Les procédures de communication réseau.*

Ces procédures sont déclenchées dès qu'un sous processus transmet la validité de conditions d'évolution frontières à son communicateur. Ce dernier gère alors la séquence suivante :

- synchronisation des sous processus (blocage du cycle d'évolution de chaque sous processus) ;

- collecte dans chaque communicateur du contenu du P.S.V. ;

- écriture dans le P.E.V. de l'ensemble des conditions d'évolution frontières collectées ;

- transfert du contenu des P.E.V. vers les sous processus associés.

IV-3) Présentation de l'aspect logiciel.

L'implantation logicielle relative au communicateur respecte le modèle de l'I.S.O.. Seules trois couches sont utilisées :

* la couche Application ;
* la couche Liaison de données ;
* la couche Physique.

La définition des différentes couches du réseau de communication repose sur les principes de dialogue communicateur - communicateur et sous processus - communicateur définis au chapitre précédent. Dans la suite du chapitre, la description du comportement des différentes couches est décrite à l'aide de réseaux de Petri saufs.

IV-3-1) *La couche Application*

Cette couche est chargée de traiter les différentes procédures d'échanges entre sous processus. Elle gère les ports d'entrées/sorties virtuels et l'ensemble des variables d'état.

La couche Application définit l'ordonnancement des primitives de communication entre communicateurs et le dialogue avec le sous processus associé.

Ceci est décrit par le modèle suivant :

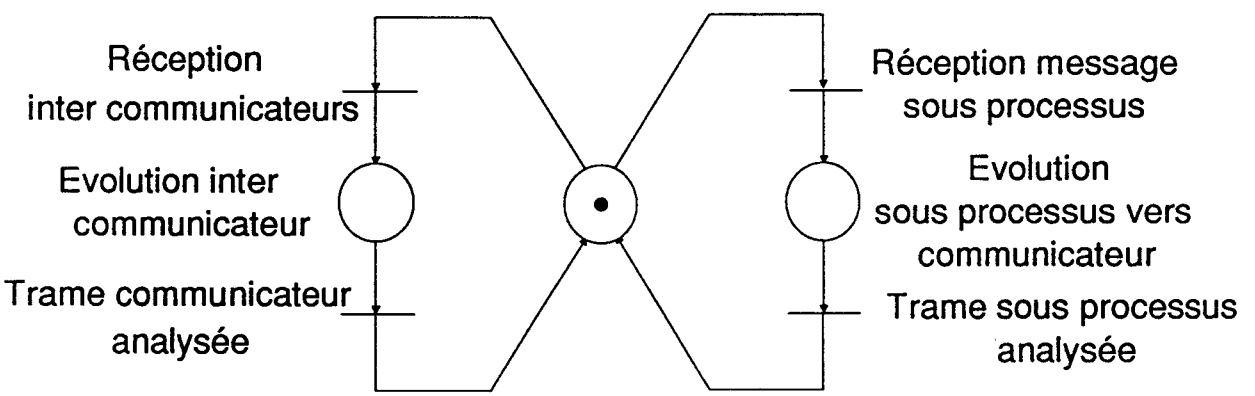

FIG. IV-3 : Analyse réception

IV-3-1-1) Evolution sous processus vers communicateur

Ces dialogues se résument à un ensemble de trois primitives :

- écriture des conditions d'évolution frontières validées à l'instant K dans le P.S.V. ;

- effacement des conditions d'évolution frontières qui viennent juste d'être dévalidées. Ceci consiste à recopier à nouveau dans le P.S.V. l'image de l'ensemble des conditions d'évolution frontières validées ;

- envoi d'un accusé de réception : cycle sous processus bloqué.

D'où le modèle associé suivant :

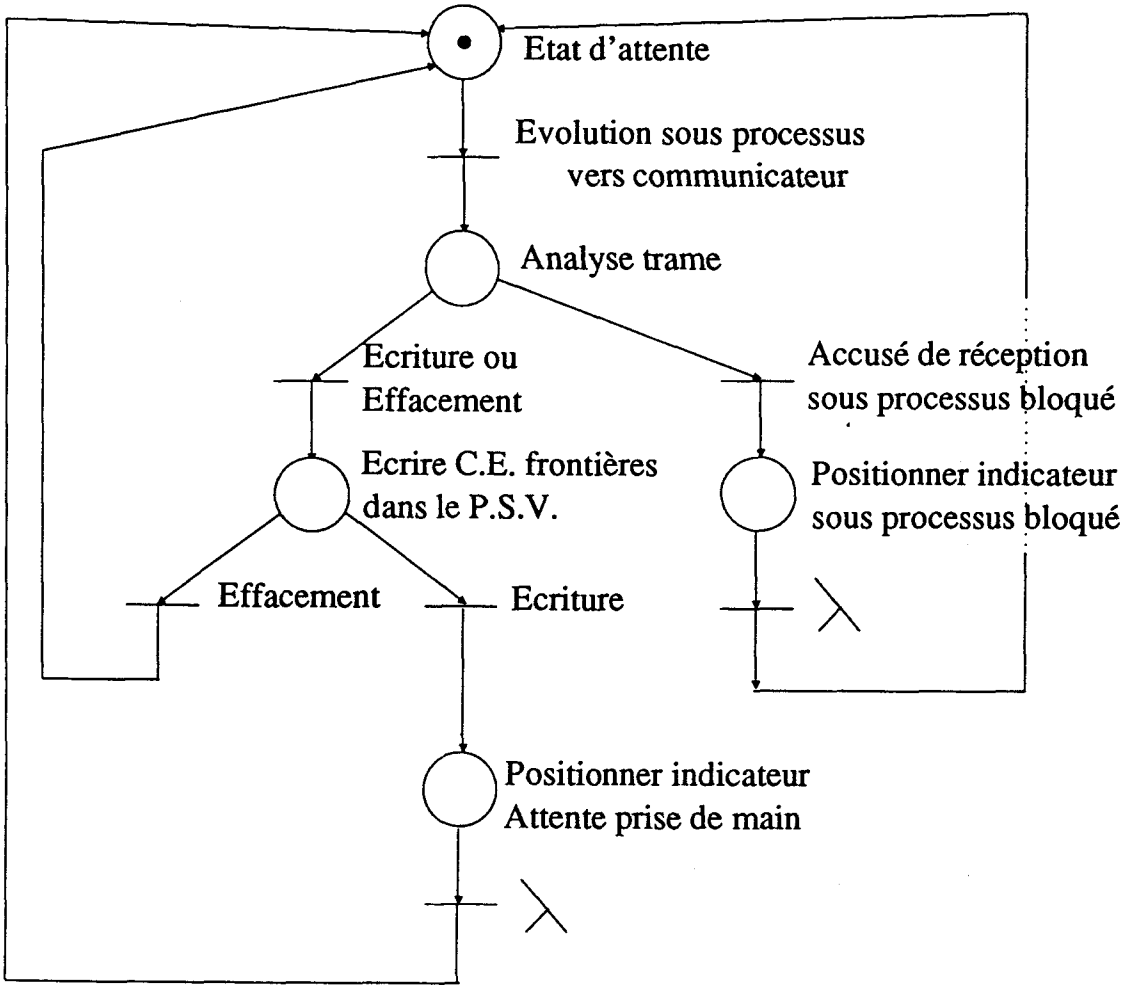

FIG. IV-4 : Evolution sous processus vers communicateur

IV-3-1-2) Evolution entre communicateurs

La gestion des communications est à ce niveau dédiée au communicateur ayant un comportement "Maître" qui a reçu de son sous processus associé une demande d'échange. Les autres communicateurs ont alors un comportement "Esclave".
Le modèle décrivant les deux états possibles d'un communicateur est le suivant :

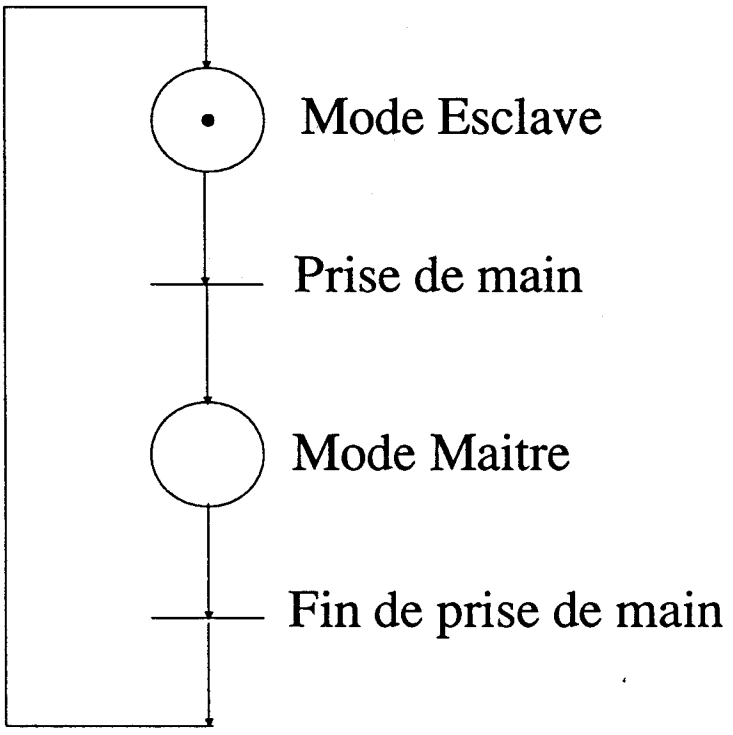

FIG. IV-5 : Mode de fonctionnement

La prise de main est une information indiquant si le communicateur peut prendre en main la gestion du dialogue. Elle est directement liée à la topologie du réseau. Pour notre cas (réseau en anneau), elle consiste en la détection d'une trame réseau d'un type particulier appelée "Jeton" qui est en sorte un droit à l'émission que les communicateurs se passent l'un après l'autre.

L'ensemble des procédures nécessaires à la communication entre deux sous processus se résume aux primitives suivantes :

- réception du jeton, choix d'une éventuelle prise de main ;
- blocage du cycle d'évolution des différents sites ;
- collecte des conditions d'évolution frontières contenues dans le P.S.V. ;
- distribution de l'ensemble des conditions d'évolution collectées ;
- déblocage de l'ensemble des cycles d'évolution.

Le modèle associé à un tel comportement est le suivant :

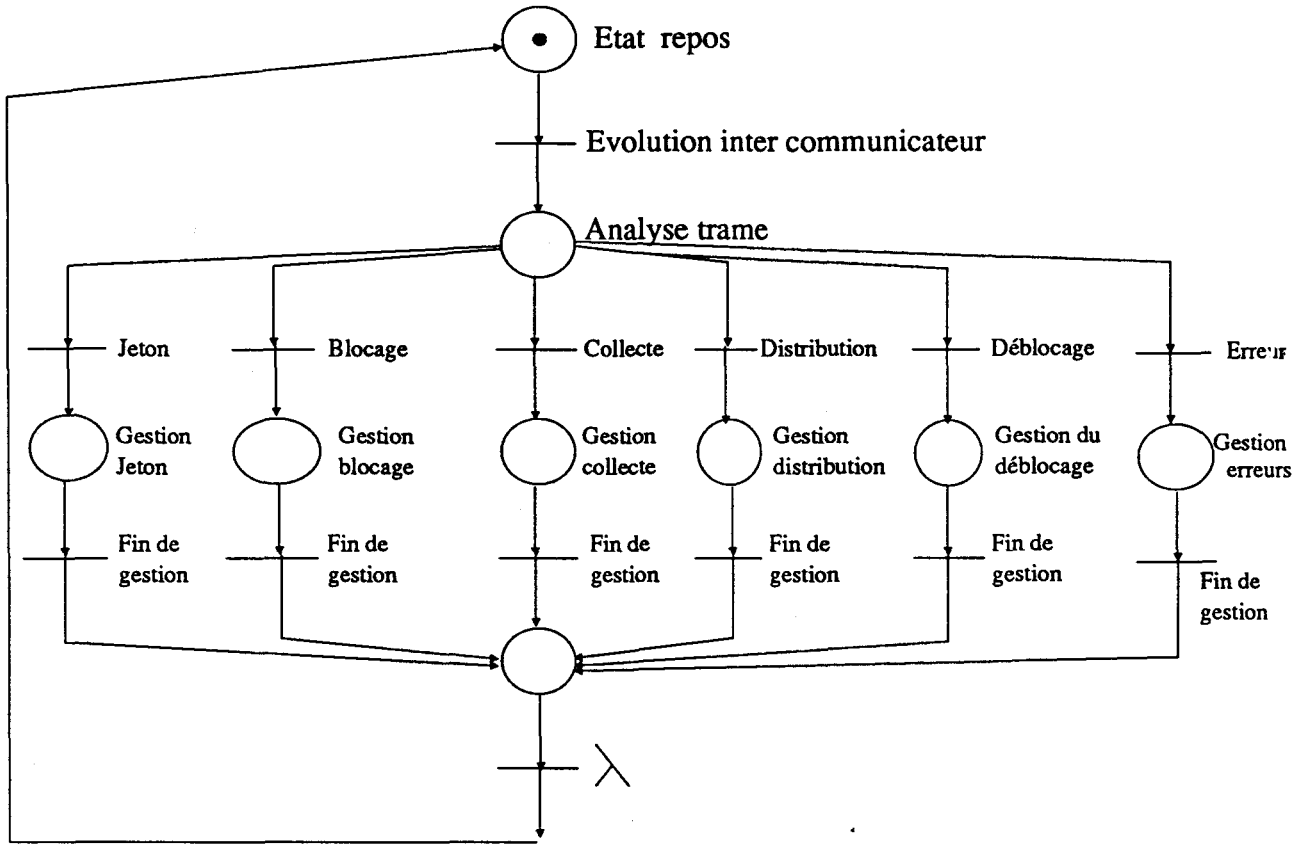

FIG. IV-6 : Evolution inter communicateurs

Dans la suite de l'exposé, nous présentons les modèles de comportement associés à la gestion des différents types de trames en fonction du mode de fonctionnement du processus considéré.

IV-3-1-2-1) *Gestion du jeton*

Deux comportements sont définis à ce niveau :

- soit le sous processus doit effectuer une prise de main pour passer en mode Maître. Il prend alors la gestion du réseau et commence une procédure de communication ;

- soit le sous processus n'a pas de prise de main à effectuer. Il retransmet alors le jeton vers le sous processus suivant.

L'algorithme associé est le suivant :

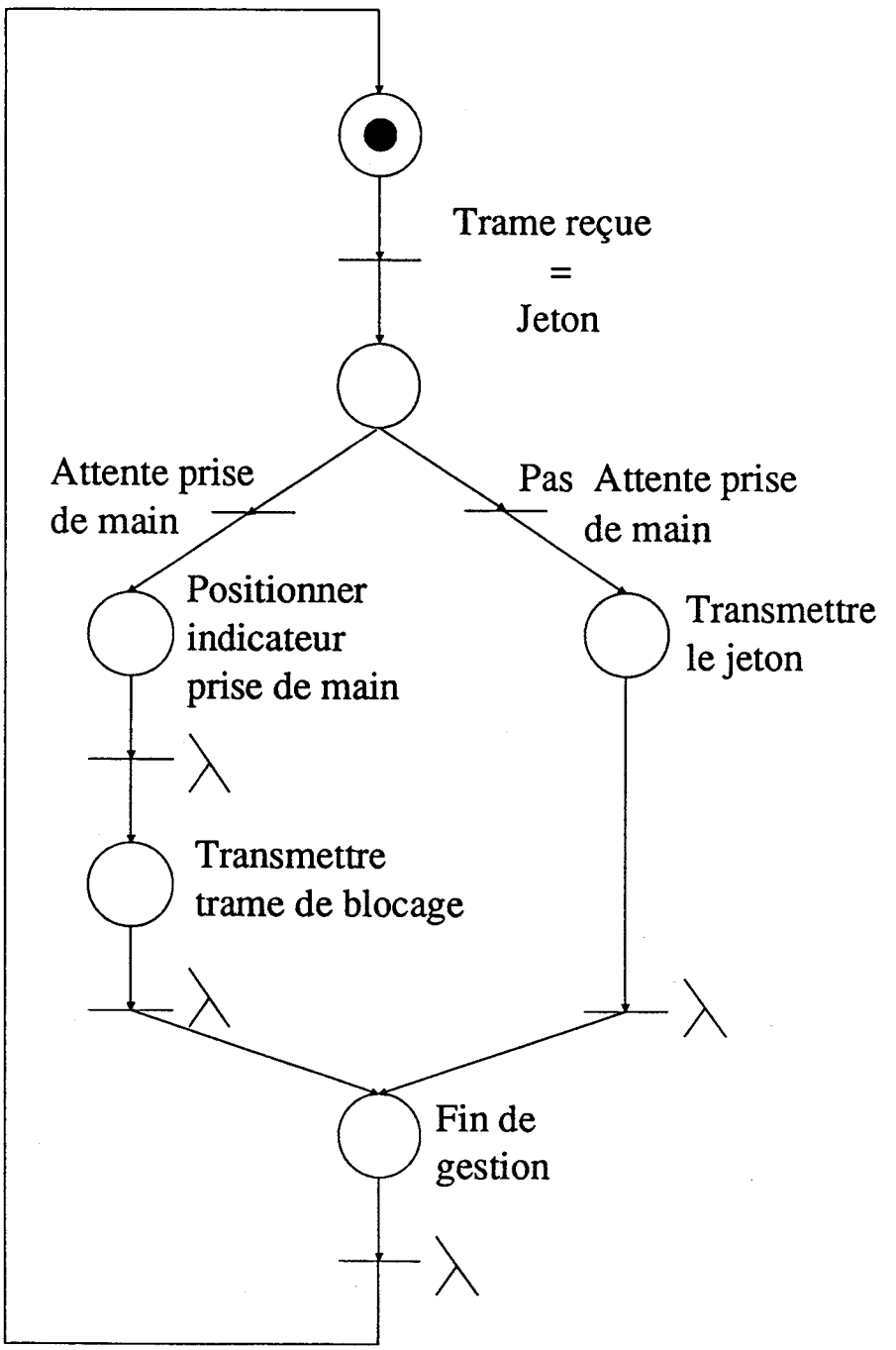

FIG. IV-7 : Traitement du jeton

IV-3-1-2-2) *Gestion du blocage*

Dans ce cas, il faut toujours envisager les deux points de vues (Maître ou Esclave) :

- pour l'état Esclave, le communicateur bloque le cycle d'évolution du sous processus associé et retransmet ensuite la trame de blocage à la station suivante ;

- pour l'état Maître, le communicateur décide après réception de la trame de blocage de lancer la procédure de collecte.

L'algorithme décrivant ce comportement est le suivant :

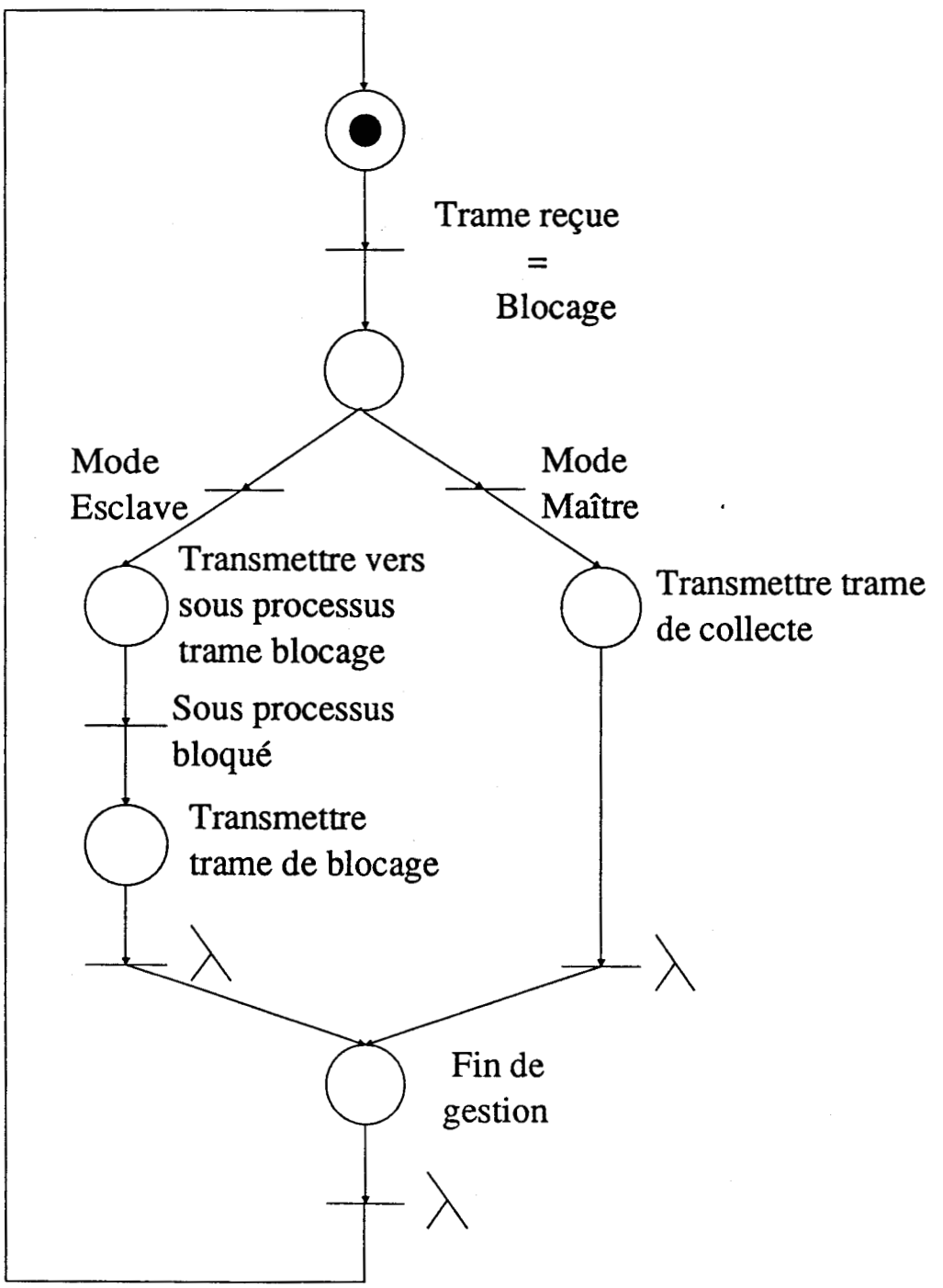

FIG. IV-8 : Traitement du blocage

IV-3-1-2-3) *Gestion de la collecte*

En fonction des deux états possibles (Maître ou Esclave) d'un communicateur, le comportement est le suivant :

- pour l'état Esclave, le communicateur ajoute à l'ensemble des conditions d'évolution contenues dans la trame de collecte, celles contenues dans son P.S.V. et retransmet cette trame vers la station suivante ;

- dans l'état Maître, après réception de la trame de collecte il entame la procédure de distribution en y insèrant le contenu de son P.S.V . ;

L'algorithme décrivant ce comportement est le suivant :

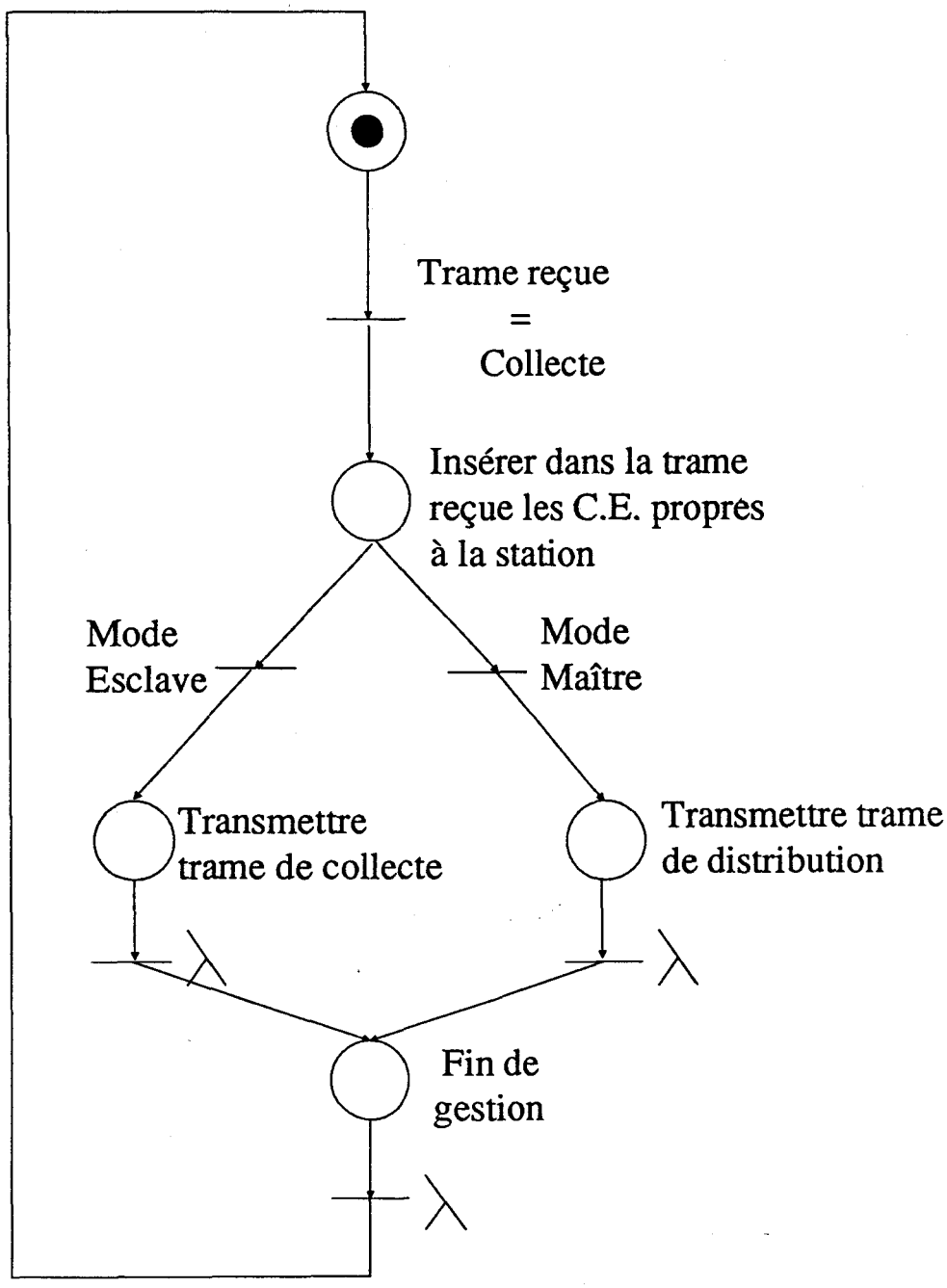

FIG. IV-9 : Traitement de la collecte

IV-3-1-2-4) *Gestion de la distribution*

Les deux comportements qui découlent du mode Maître ou Esclave sont les suivants :

- en mode Esclave, le communicateur écrit dans son P.E.V. l'ensemble des conditions d'évolution contenues dans la trame de distribution et retransmet ensuite cette trame vers la station suivante ;

- en mode Maître, le communicateur écrit aussi dans son P.E.V. l'ensemble des conditions d'évolution distribuées mais transmet après coup la trame de déblocage vers la station suivante.

Ces comportements sont décrit par le R.D.P. suivant :

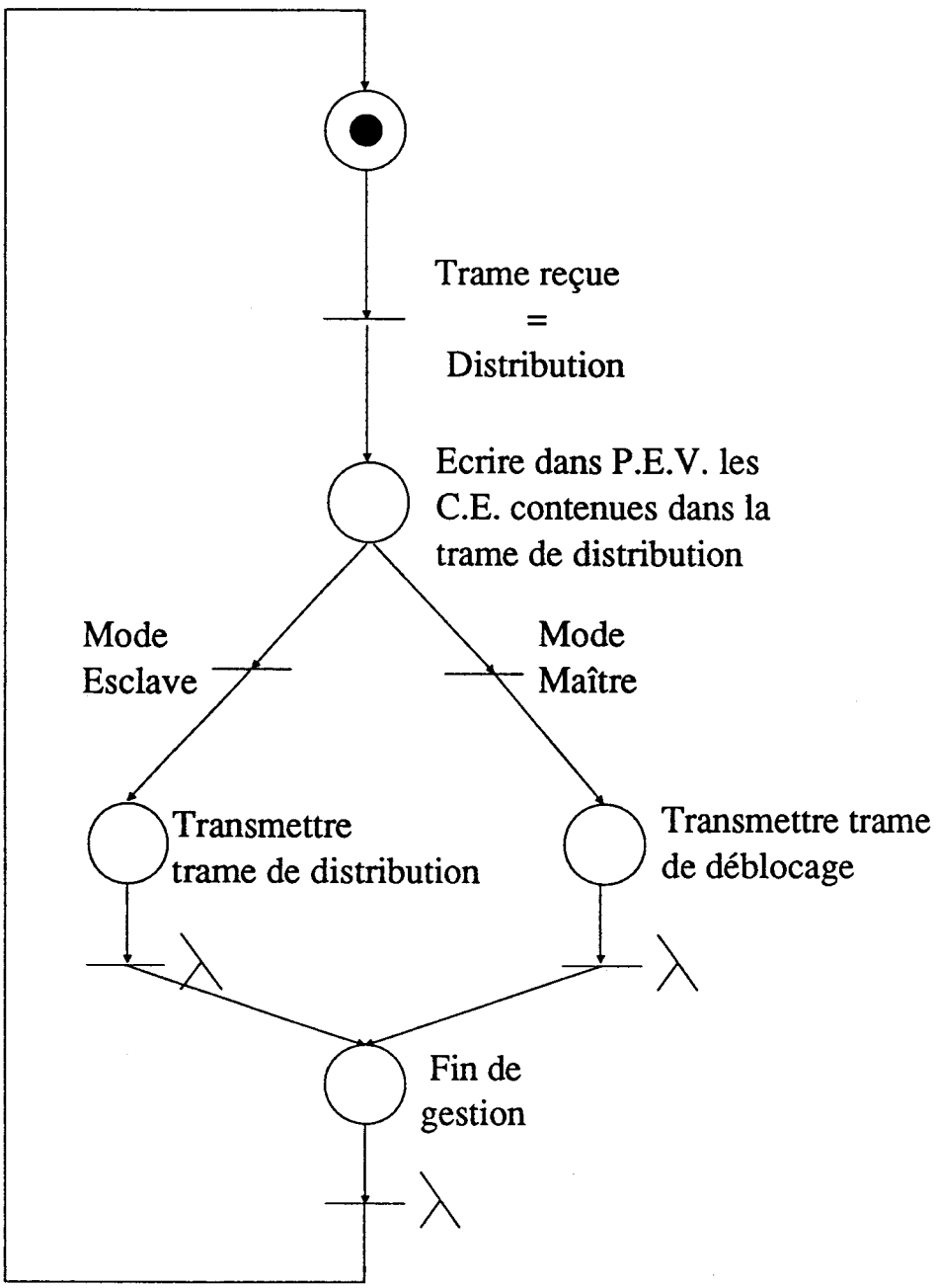

FIG. IV-10 : Traitement de la distribution

IV-3-1-2-5) *Gestion du déblocage*

La trame de déblocage a pour principale fonction de relancer le cycle d'évolution de chaque sous processus. Le communicateur se trouvant Maître doit en plus après réception de cette trame, libérer le Jeton permettant à un autre communicateur de prendre la main. Ceci est décrit par l'algorithme suivant :

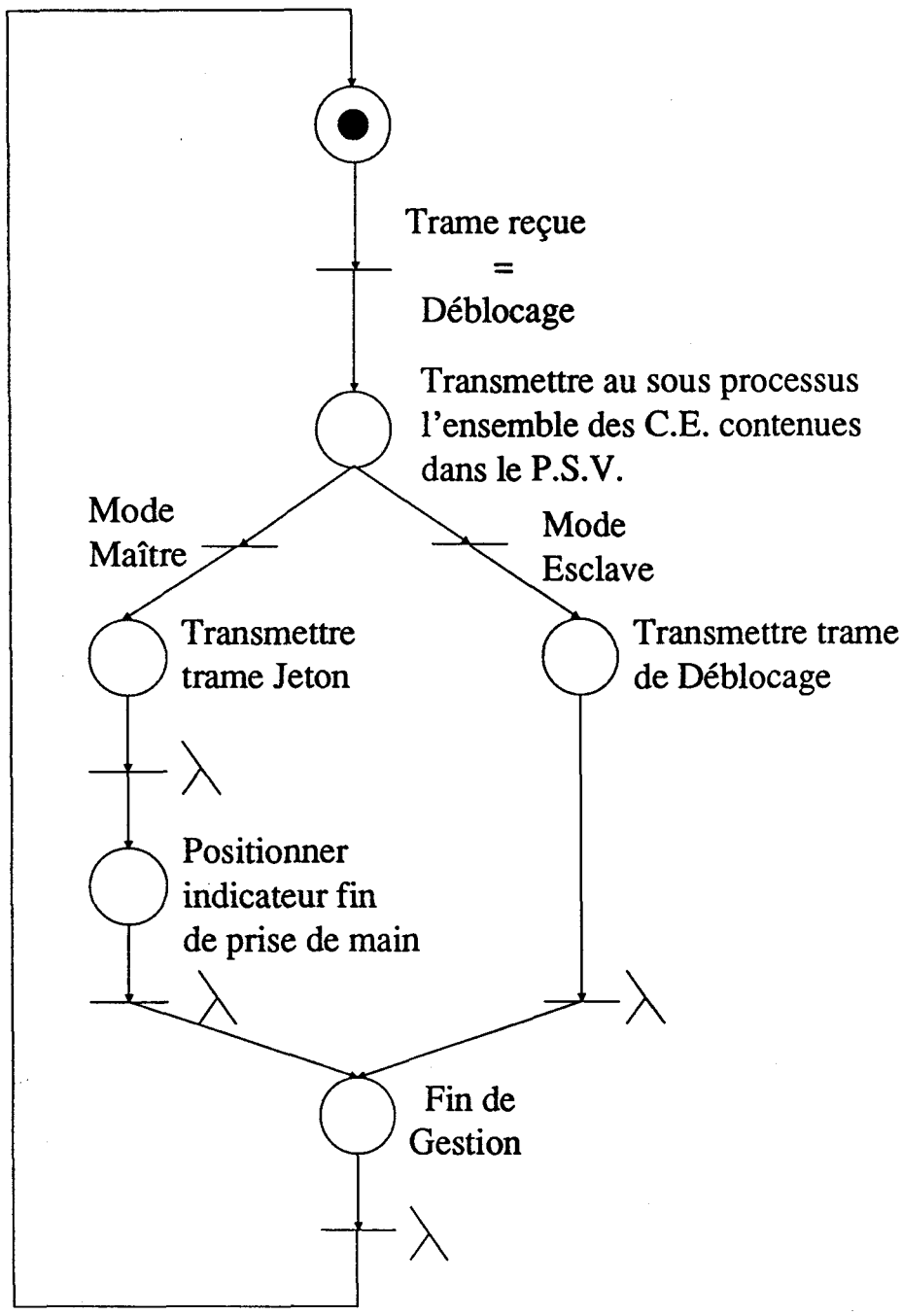

FIG. IV-11 : Traitement du déblocage

Remarque : Le fait qu'un communicateur transmette à son sous processus l'ensemble des conditions d'évolution contenues dans son P.E.V. sert d'ordre de déblocage du cycle d'évolution.

IV-4) La couche liaison de données.

Cette couche est décrite par deux comportements distincts :

* comportement en émission ;

* comportement en réception.

En émission, elle est chargée d'élaborer la trame à émettre. Cela consiste à effectuer deux opérations :

- Encapsuler les données à transmettre en fonction de l'indicateur de type de trame positionné par la couche application ;

- Calculer le checksum et l'introduire dans la trame à transmettre pour une trame à transmettre sur le réseau.

En réception, elle est chargée d'identifier la trame reçue et de vérifier les erreurs de contenu de cette trame. Ceci se résume en trois opérations :

- détection d'erreurs (longueur trame, checksum) pour une trame réseau ;

- identification de la trame reçue

- information de la couche application par positionnement d'un indicateur ;

Ces deux comportements sont mis en oeuvre pour les communications entre sous processus et entre sous processus et communicateur.

Le modèle décrivant le traitement de réception est le suivant :

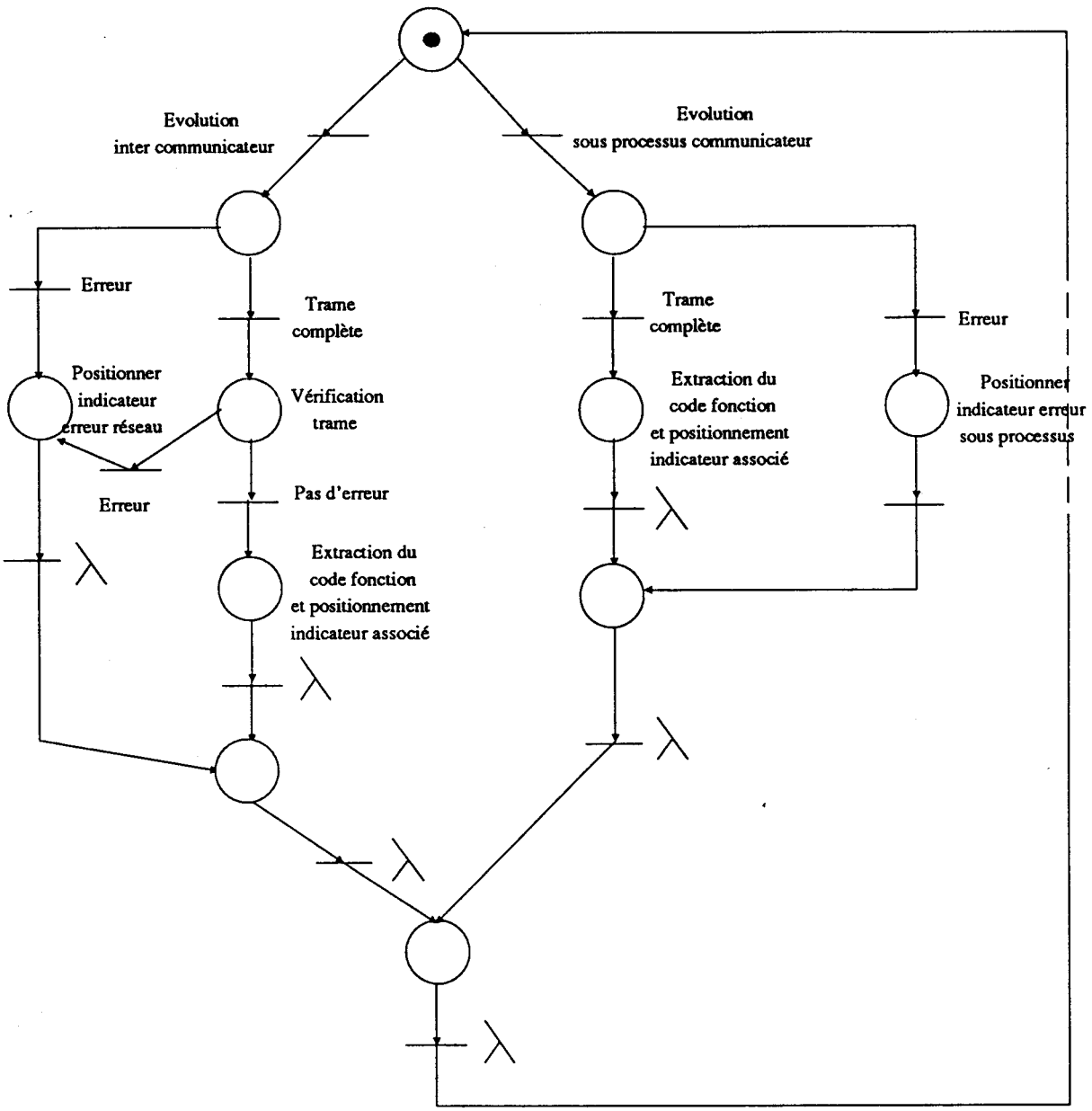

FIG. IV-12 : Traitement réception

Le modèle décrivant le traitement d'émission est le suivant :

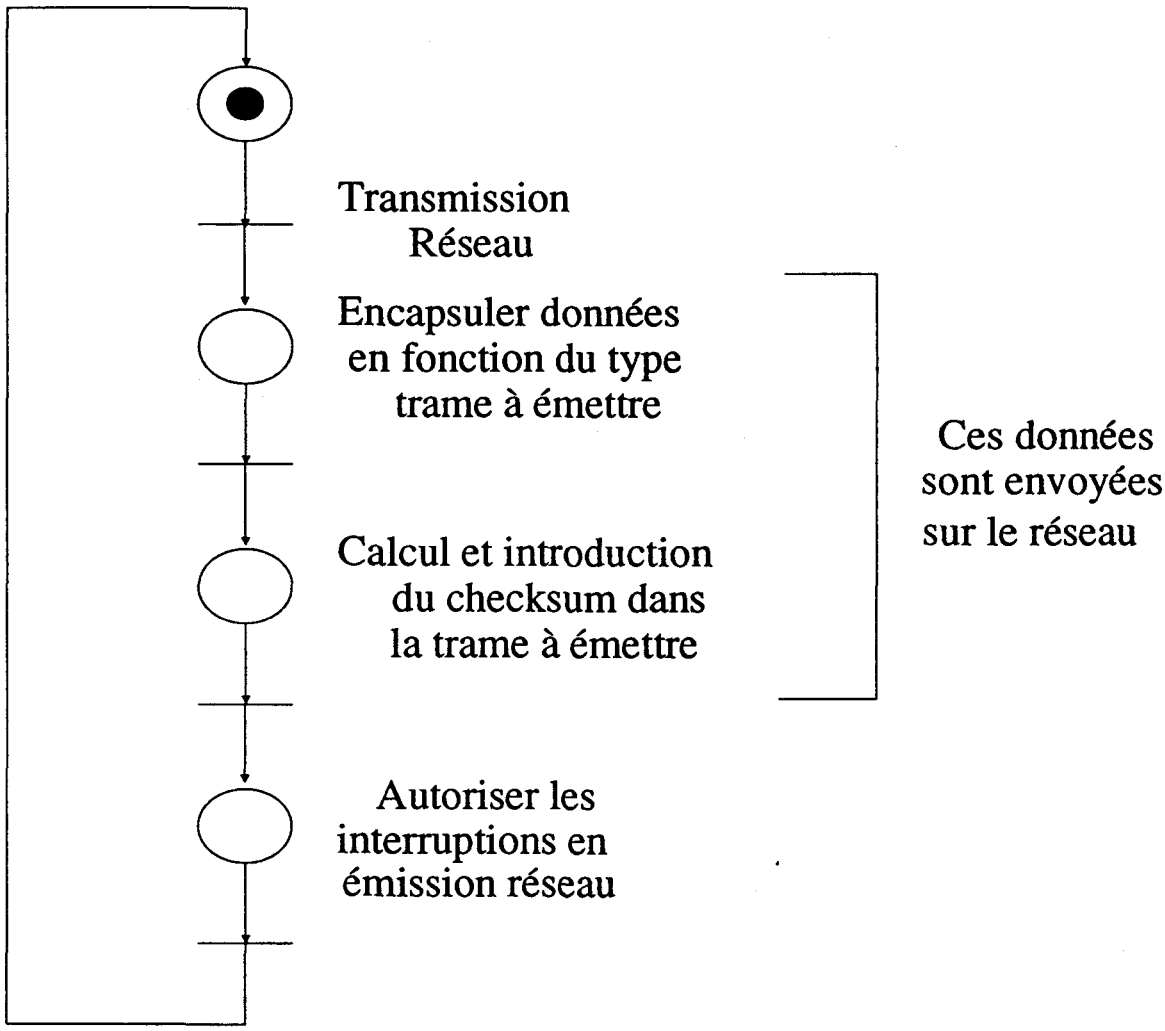

FIG. IV-13 : Traitement émission

IV-5) La couche physique

Cette couche définit la structure physique de la liaison entre deux entités qui communiquent. A cette structure physique, il faut ajouter un ensemble logiciel capable de piloter les éléments physiques qui ne sont pas capables à eux seuls d'effectuer la transmission.

La définition de la couche physique permet d'assurer :

- l'adaptation des émetteurs et des récepteurs au type de signaux reçus ;

- le contrôle de la communication entre émetteur et récepteur pour la détection des erreurs de transmissions.

L'aspect logiciel se résume essentiellement à l'exécution d'un sous programme traitant :

- la réception de données venant du réseau ;

- l'émission de données sur le réseau ;

- la réception de données venant du sous processus ;

- l'émission de données vers le sous processus ;

IV-5-1) *Structure de données.*

Pour permettre ces différents traitements, chaque communicateur utilise une structure de données facilitant le stockage des données et leur utilisation par les différentes couches. Cette structure est définie par l'utilisation de tampons de stockage gérés par des pointeurs tournants.

Comme nous l'avons vu dans le chapitre précédent, les données qui transitent entre un communicateur et un sous processus sont stockées dans des ports d'entrées/sorties virtuels : le P.E.V. et le P.S.V.. Ces derniers sont considérés par chaque sous processus comme une image de l'état des conditions d'évolution frontières du processus global. Ils ne peuvent donc être utilisés comme structure de données sur lesquelles sont effectués des traitements de transmission.

Pour résoudre le problème de stockage et faciliter le traitement des données par le communicateur, un ensemble de quatre tampons de communication a été défini.

Pour interfacer le réseau de communication avec le communicateur, deux de ces tampons permettent :

* l'un, de stocker les données reçues du réseau. Il est appelé P.E.R. (Port d'Entrée Réseau) ;

* l'autre, de transmettre sur le réseau des données stockées dans ce tampon par des couches supérieures. Ce tampon est appelé P.S.R. (Port de Sortie Réseau).

Pour l'interfaçage entre le communicateur et le sous processus, les deux derniers tampons permettent :

* l'un, de stocker les données en provenance du sous processus. Il est appelé P.E.C. (Port d'Entrée Communicateur) ;

* l'autre, de transmettre des données au sous processus. Il est appelé P.S.C. (Port de Sortie Communicateur).

IV-5-2) *Gestion des tampons de transmission par la couche physique.*

La gestion des différents tampons par la couche physique se fait par interruption. Par détection du type d'interruption, la couche physique se charge de stocker ou transmettre des données entre les communicateurs ou entre un communicateur et un sous processus.

Le modèle décrivant ce comportement est le suivant :

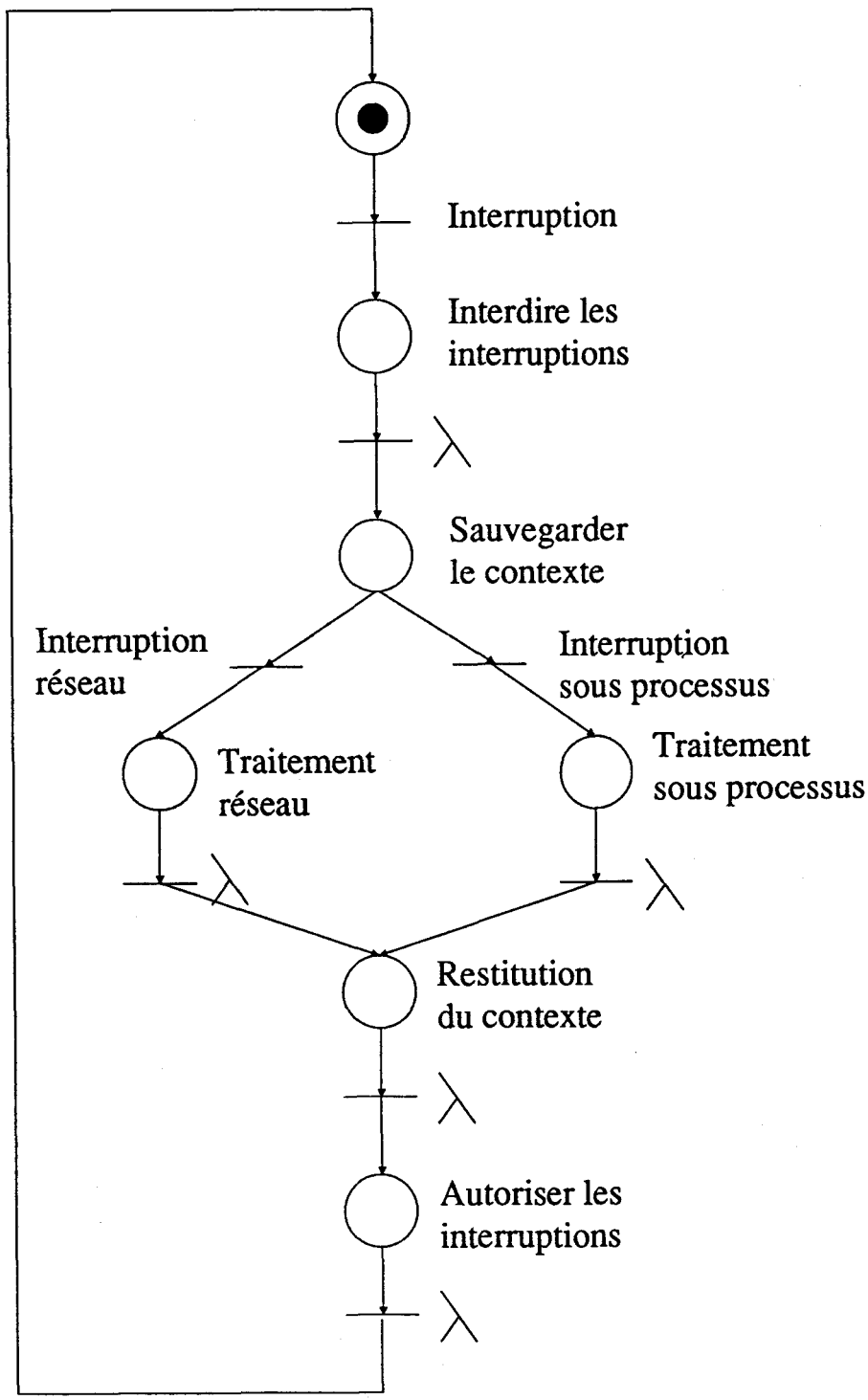

FIG. IV-14 : Traitement des interruptions

avec

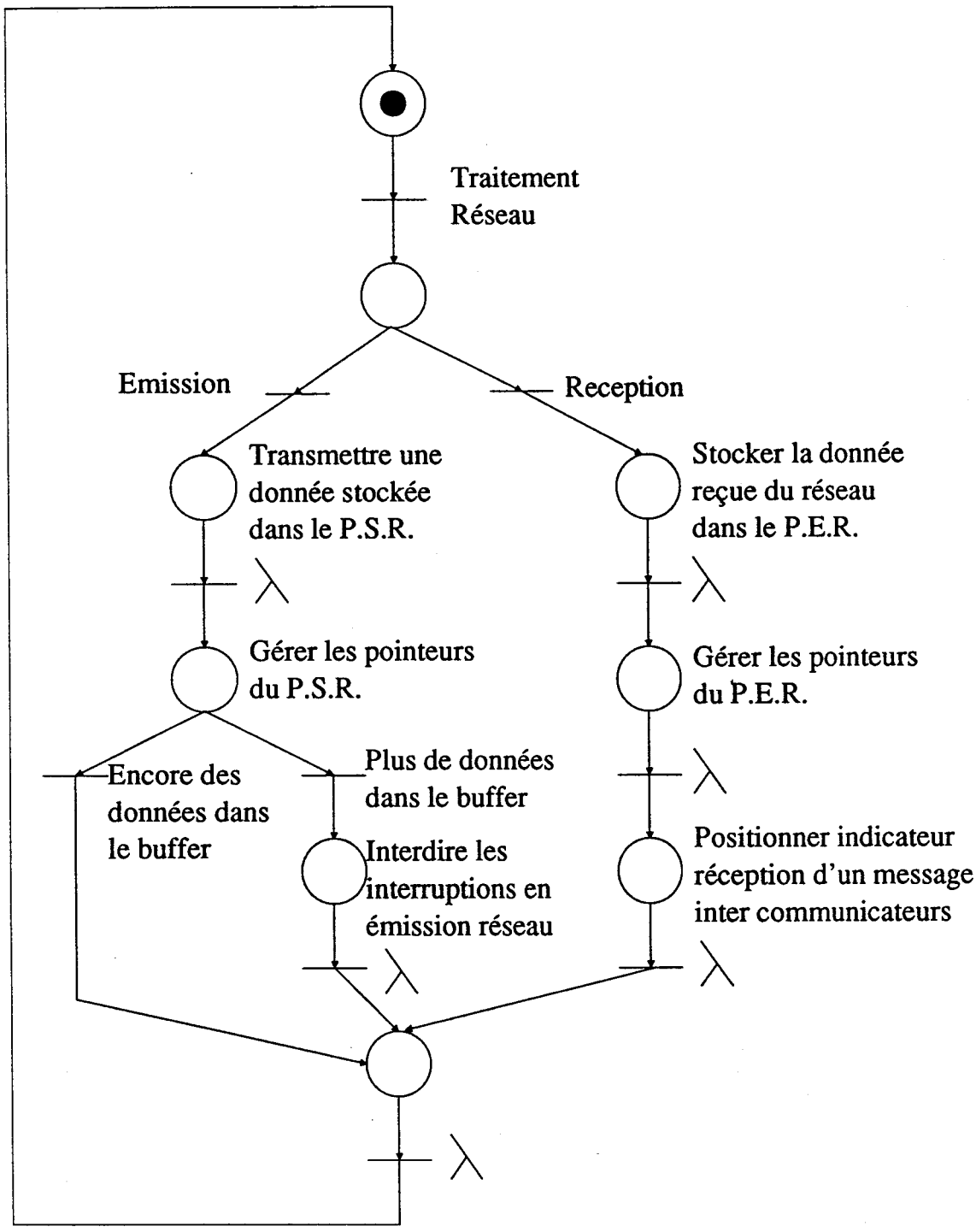

FIG. IV-15 : Traitement réseau

et

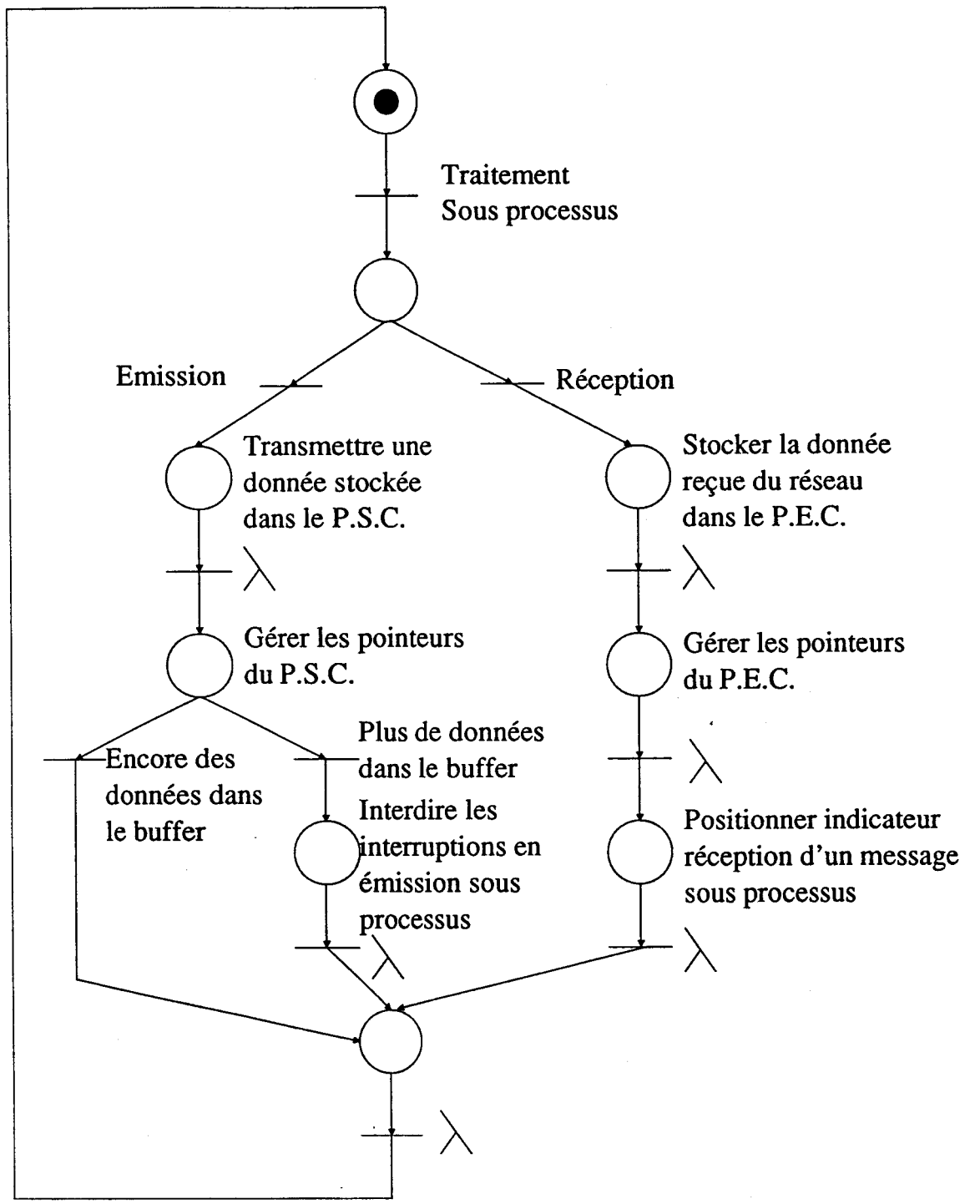

FIG. IV-16 : Traitement sous processus

IV-6) **Aspect matériel.**

Le communicateur est conçu à partir d'un micro contrôleur 8 bits de chez Intel, le 8052 AH Basic. Ce circuit offre la particularité d'être facile à programmer. En effet, il possède dans une R.O.M. interne au circuit un interpréteur basic permettant un développement logiciel souple.

Autour de ce circuit viennent se greffer un ensemble de circuits permettant le stockage d'informations (Mémoires) et le transfert d'information (ports d'Entrées/Sorties).

Les ressources mémoire de la carte sont de 16 K d'E.P.R.O.M. et de 24 K de R.A.M..

Pour le transfert d'information, nous avons fait appel à deux UART programmables.

L'architecture matérielle du circuit est la suivante :

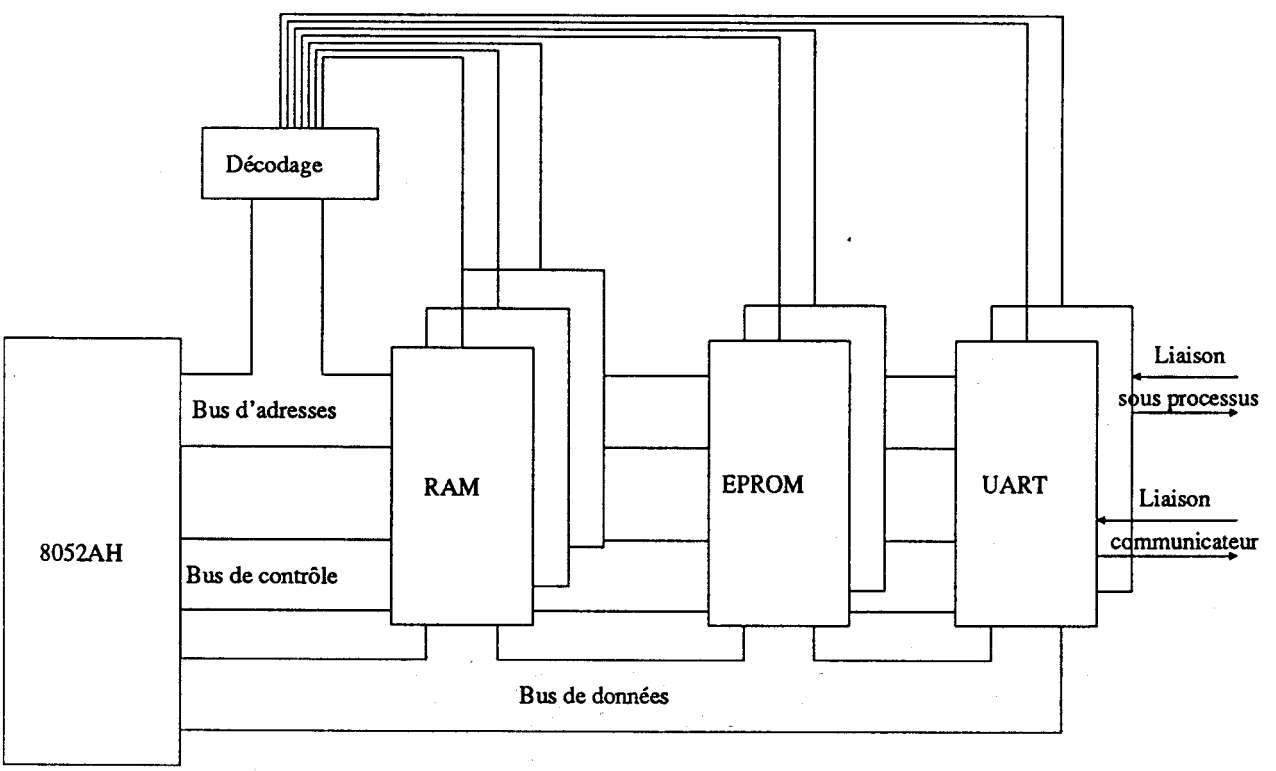

FIG. IV-17 : Architecture materielle du communicateur

Conclusion : La mise en oeuvre d'une implantation distribuée a été effectuée dans le but de montrer la faisabilité de la méthode d'implantation présentée dans ce mémoire.
Pour cela, on a utilisé un réseau fonctionnant sur le principe de l'anneau à jeton.
Lors des essais, nous avons pu constater la bonne marche de l'ensemble. Le comportement global des processus demeure parfaitement cohérent avec celui décrit par le modèle global qui met en évidence un grand nombre de cas de figures types : noeuds ET, noeuds OU, etc....

Il est à noter qu'avec le communicateur développé, l'ensemble des processus communique de manière relativement lente. Ceci est essentiellement dû au langage utilisé pour développer l'application (Basic interprété).

Il est claire que dans une réalisation industrielle, l'utilisation de processeur rapides et/ou spécialisés permettrait d'accroître les vitesses d'échanges dans des proportions considérables (1000 fois plus rapide). Dès lors, le contrôle distribué peut s'envisager dans des conditions tout à fait compatible avec la dynamique des systèmes pilotés.

Conclusion générale.

L'objet de ce mémoire est de définir une méthodologie de conception des processus distribués. Pour ceci, nous avons montré qu'un système physiquement réparti peut être étudié de manière global (Modèle global) et mis en oeuvre sous une forme distribuée.

L'avantage d'une telle démarche est de permettre une définition et une analyse du modèle de comportement global sans tenir compte de l'aspect distribué. Les aspects communications intervenant dans les modélisations habituelles des processus communicant (Cf. I-5) sont absents du modèle d'étude, ce qui en facilite la conception et l'analyse.

Cette aspect communication est pris en compte au niveau de la méthode d'implantation. Le modèle de comportement étant global, c'est l'implantation qui est réalisée de manière distribuée.

L'implantation distribuée d'un modèle global se résume aux deux phases suivantes :

* Partition du modèle global sur la base des choix de regroupement des événements et des opérations (Cf. II-3). La méthode choisie pour décrire ce partitionnement doit faire apparaître deux types de comportement : un comportement interne à chaque partie du processus global et un comportement inter processus qui régit la coopérations entre ces différentes parties ;

* Exécution, par chaque partie du système distribué, des deux types de comportement. Les comportements internes de chaque sous processus peuvent être traités de manière asynchrone les uns des autres alors que le comportement inter processus doit être traité de manière synchrone pour garder cohérent et déterministe le comportement du processus global.

Dans la pratique, l'aspect synchrone du comportement inter processus est essentiellement tributaire du support de communication utilisé. Ce support est généralement un réseau local qui n'offre pas de possibilité d'échange synchrone.

Dans ce mémoire, nous proposons une architecture de communication qui tente de résoudre ce problème.

Cette architecture a bien sûr comme effet de ralentir l'évolution des différents sous processus. Toutefois il est probable que dans un proche avenir, les progrès technologiques, notamment en matière de rapidité des composants électroniques, permettront de pallier cette difficulté.

BIBLIOGRAPHIE

[AND 83] André F., Herman D., Verjus J.P., Synchronisation de programmes parallèles. Editions Dunod informatique, 1983.

[BER 85] Berthelot G., Analyse de processus parallèles par transformation de réseaux de Petri, application à un protocole de réseau. T.S.I., vol. 4, n° 1, 1985, p. 73 à 83.

[BOC 77] Bochmann G.V., Gecsei J., A unified method for spécification and vérification of protocols. Congrès I.F.I.P., éditions B. Gilchrist, North Holland pub. compagny, 1977, p. 229 à 234.

[BRA 83] Brams G.W., Réseaux de Petri : Théorie et pratique. Editions Masson, 1983.

[CAL 82] Calvez J.P., Une méthodologie de conception des systèmes multi-micro ordinateurs pour les applications de commande en temps réels. Thèse de docteur Es-Sciences, Nantes, 1982.

[COUR 74] Courvoisier M., Etude des systèmes logiques de commande asynchrone à évolutions simultanées. Thèse de docteur Es-Sciences en Automatique, Toulouse, 1974.

[COUR 83] Courvoisier M., Valette R., Bigou J.M., Esteban P., Réseaux d'automates et ateliers flexibles. Congrès Afcet Automatique : Productique et Robotique intelligente, Besançon, 1983.

[COUV 89] Couvreur M., Fourmaux D., Sannier L., Vasseur C., Partition d'un graphe de commande global et implantation répartie : Application au contrôle distribué de processus industriels. 39th International mini and micro computer and their applications, Zurich, 1989.

[DAV 89] David R., Hassane A., Du Grafcet aux réseaux de Petri. Editions Hermes, Traités des nouvelles technologies, série automatique, 1989.

[DEF1 86] Defrenne J., Modélisation de la partie opérative - Impact sur la sécurité et la maintenance des automatismes à évolution séquentielle. Thèse de docteur Es-Sciences Physique, Lille I, 1986.

[DEF2 86] Defrenne J., Toulotte J.M., Hachemani R., Validating of control software for industrial systems in simultaneous evolution according. Convention automatique productique, Paris, mai 1986.

[DER 84] Derail Y., La commande séquentielle répartie : Synthèse du Grafcet et utilisation des réseaux locaux. Thèse 3^{em} cycle en automatique, Grenoble, 1984.

[DES 82] Deschizeaux P., Temps et événements dans les systèmes distribués de contrôle de procédé, R.A.I.R.O. Automatique/Systems Analysis and Control, vol. 16, n° 3, 1982, p. 259 à 274.

[FDI 89] Fdida S., Pujolle G., Modèles de systèmes et de réseaux. Tome 1: Performance. Editions Eyrolles, 1989.

[GRO 83] Groupe systèmes logiques de l'Afcet, Grafcet : Interprétation algébriques et algorithmes et temporisations. Nouvel automatisme, septembre 1983.

[HOA 78] Hoare C.A.R., Communicating sequentiel processes. Communication of A.C.M., vol.21, n° 8, août 1978.

[HOR 81] Hortait E., Protocoles de communications de bas niveaux et réseaux locaux. Thèse 3^{em} cycle, Paris XI, 1981.

[LAI 88] Laine T., Thomesse J.P., F.I.P.- Un bus de terrain pour la commande de procédés industriels. Nancy I, 1988.

[LAM 78] Lamport L., Time, clocks and the ordering of events in a distributed system, Communication of the A.C.M., juillet 1978, vol. 21, n° 7, p. 558 à 565.

[LAZ 88] Lazarev V.G., Zoreva L.N., Le Moli G., Decompositional approach to the design of protocols for interaction of entities in distributed systems. Congrès I.F.A.C. Distributed intelligence systems, Varna Bulgaria, 1988.

[LEP 89] Lepage F., Afidal F., ..., Les réseaux locaux industriels. Editions Hermes, Traité des nouvelles technologies, série automatique, 1989.

[LES 88] Lesventes G., YAMS : Les systèmes d'automates à compteurs. Rapport de recherches, I.N.R.I.A.-Rennes, mars 1988.

[LES 89] Lesventes G., Systèmes d'automates à compteurs et semi linéarité des ensembles d'états accessibles : "Forlorn hope", Thèse docteur Es-Sciences en Informatique, Rennes I, 1989.

[MEJ 89] Mejia Olvera M.C., Contribution à la conception d'un réseau local temps réel pour la robotique. Thèse 3eme cycle Informatique, Rennes I, 1989.

[MOA 85] Moalla M., réseaux de Petri interprétés et Grafcet. T.S.I., vol. 4, n° 1, 1985, p. 17 à 30.

[PAR 81] Parsy J.P., Dispositif interactif d'aide à la conception des programmes de commandes de processus logiques industriels. Thèse 3eme cycle en automatique, Lille I, 1981.

[PAT 84] Pattavina A., Trigila S., Combined use of finite - state machines and Petri nets for modelling communicating processes. Electronics letters, vol. 20, n° 22, octobre 1984.

[RAY 85] Raynal M., Algorithmes distribués et protocoles. Editions Eyrolles, 1985.

[ROU 85] Roux O., "Electre" : Expression et modélisation du comportement de processus temps réels répartis communicants. Thèse 3em cycle, Nantes, 1985.

[RUD 86] Rudnianski M., Architecture de réseaux : le modèle I.S.O. Editions Editests, 1986.

[SAN 89] Sannier L. Communicateur de réseau pour automates : conception et expérimentation. Rapport de D.E.A. Informatique industrielle et Automatique, Lille I, 1989.

[TAN 89] Tankoano J., Derniame J.C., Réseaux de Petri et applications distribuables. T.S.I., vol. 8, n° 4, 1989.

[TAN 88] Tankoano J., Une approche méthodique pour la conception certifiée des systèmes de commande des automatismes industriels répartis. Thèse d'état en Informatique, Nancy I, 1988.

[THE 78] Thelliez S., Pratique séquentielle et réseaux de Petri. Editions Eyrolles, collection E.E.A., 1978

[THE 80] Thelliez S, Toulotte J.M., Grafcet et logique industrielle programmée. Editions Eyrolles, collection ingénieurs E.E.A., 1980.

[THO 87] Thomesse J.P., Les réseaux locaux industriels , collection Novotique, 1987.

[TOU 79] Toulotte J.M., Parsy J.P., A method for decomposing interpreted Petri Nets and its utilization. Digital Processes, 1979, p. 223 à 234.

[VAL 82] Valette R., Courvoisier M., Bigou J.M., Les réseaux d'automates : analyse de la coopération. Journées d'études S.E.E., Gif sur Yvette, 1982, p. 733 à 740.

[VER 87] Verjus J.P., La recherche publique française sur le parallélisme et la répartition : le programme C^3, T.S.I., vol. 6, n° 2, 1987.

[ZAH 80] Zahnd J., Machines séquentielles "Traité d'électricité", vol. 11, éditions Georgi 1980 ou éditions Dunod 1989.

ANNEXE

LES RESEAUX DE COMMUNICATION.[LEP 89], [RUD 86], [THO 87]

La notion de travail réparti entre plusieurs calculateurs permet de rassembler en plusieurs machines, un ensemble de potentialités. La répartition, pour être efficace par rapport à un traitement centralisé, doit aller de paire avec une circulation d'informations très importante.
Pour qu'un ensemble de machines échangent des informations, il est nécessaire d'utiliser un moyen de communication. Une des techniques actuelles est de regrouper les entités devant communiquer dans une configuration que l'on appelle un réseau de communication. Dans le domaine industriel, les réseaux utilisés se limitent souvent au cadre d'un atelier où d'une usine. De ce fait, ils sont généralement appelés des réseaux "locaux". Dans une utilisation industrielle, les réseaux sont souvent classés en trois catégories [THO 87] :

* les réseaux d'entreprise ;

* les réseaux d'atelier ;

* les réseaux de terrain.

Les réseaux d'entreprise (ex : Ethernet, Decnet) sont situés dans les bureau des sites industriels au niveau de la gestion de production. Ces réseaux servent à raccorder les stations de travail informatiques entre elles et parfois l'ensemble de ces stations à des réseaux publics ou à des liaisons privées longues distances. Ces réseaux ne sont pas utilisés pour faire du contrôle commande de processus et non donc pas à faire face à des contraintes temps réels

Les réseaux d'ateliers (ex: Telway, Map) sont utilisés pour raccorder divers équipements industriels et réaliser des applications de contrôle commande coordonnées. Ces réseaux permettent de garantir aux applications temps réel des temps d'accès rapides ce qui facilite la mise en place d'applications ayant besoin de temps de réponse courts.

Les bus de terrain (ex: Digibus, Fip) sont des réseaux de communications utilisés pour raccorder des capteurs, des actionneurs et des machines de contrôle commande (ex : Automate programmable). Ce type de réseau offre l'avantage de remplacer les traditionnelles liaisons point à point existant entre chaque capteur, actionneur et son organe de contrôle commande.
Un réseau est composé d'entités communicantes et de liaisons entre ces entités :

* les entités communicantes représentent les unités de traitement de l'information (automates, micro-ordinateurs,...) ;

* les liaisons ou canaux de transmission servent à relier les entités entre elles.

Les réseaux locaux peuvent être classés en fonction des critères suivants:

* la topologie, c'est à dire la localisation des entités communicantes et les liaisons entre ces entités ;

* le protocole d'accès au réseau qui permet d'organiser correctement les transmissions entre les différentes entités.

A-1) La topologie

Elle caractérise l'agencement des liaisons entre les différentes entités communicantes. Cinq possibilités fondamentales d'agencement apparaissent :

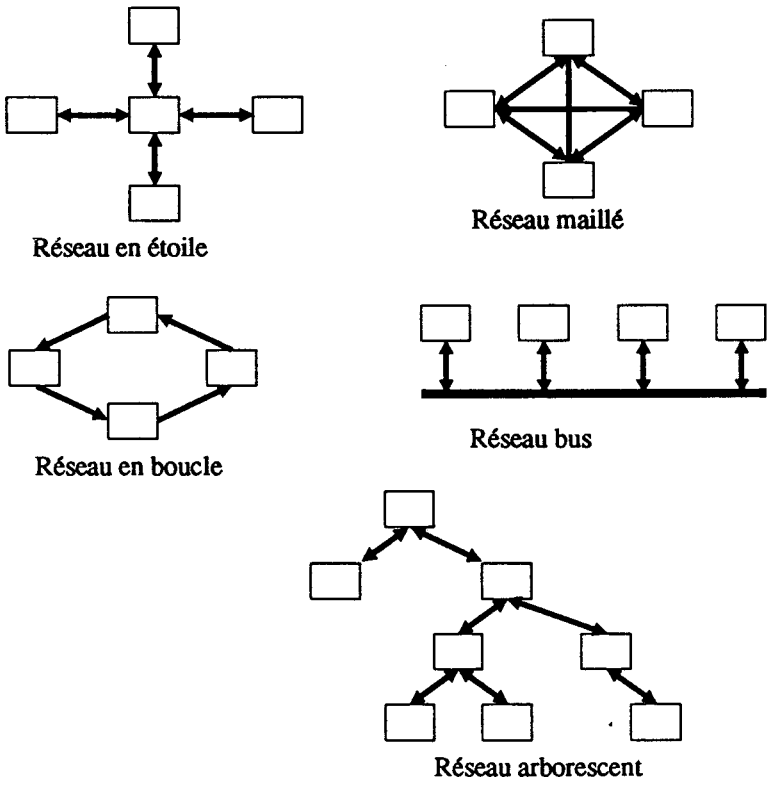

FIG. A-1

* Le réseau en étoile :

Ce type de réseau est un réseau fortement centralisé puisqu'une seule entité est directement reliée à l'ensemble des autres entités.

* Le réseau maillé :

Ce type de réseau à comme particularité que deux entités quelconques sont adjacentes. Cette forme de réseau est appelée "entièrement maillé" en opposition au cas où il existe au moins un couple d'entités non adjacentes.

* Le réseau en boucle ou en anneau :

C'est un réseau où chaque entité a deux entités adjacentes. La boucle est mono-directionnelle si le transfert d'information s'effectue dans un seul sens. Elle est bi-directionnelle dans le cas contraire.

* Le réseau en bus :

C'est un réseau, où chaque entité est reliée à la même voie de transmission appelée "bus". Ce type de réseau est aussi appelé "réseau à voie multipoints".

* Le réseau arborescent :

C'est un compromis des différents types de réseaux précédents. Certaines entités possèdent une seule adjacence tandis que d'autres en ont plusieurs.

Les réseaux précédents peuvent servir de base à la construction de réseaux plus complexes. On parle alors de réseaux interconnectés.

Exemple:

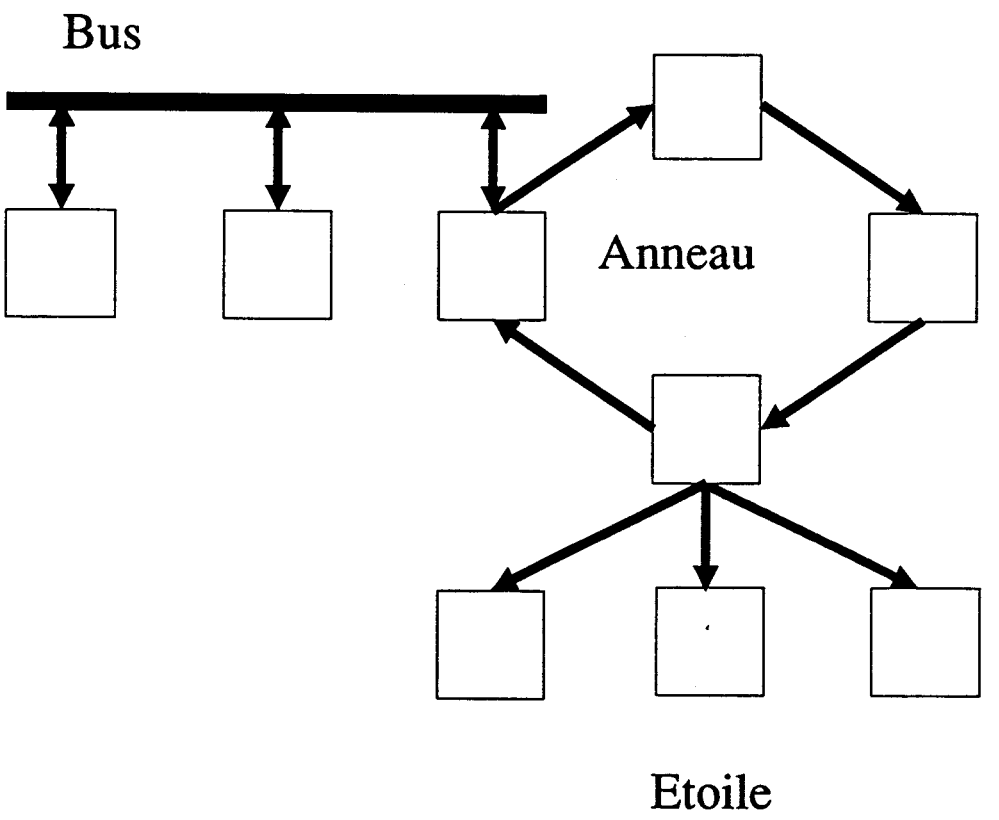

FIG. A-2 : Réseaux interconnectés

A-2) Protocole d'accès au réseau.

Le mode d'accès peut être réalisé de manière statique ou dynamique :

* le mode d'accès statique attribue périodiquement une tranche de temps à chaque entité communicante. Ce type d'accès a l'avantage d'éliminer les transmissions simultanées mais n'est pas utilisé dans les réseaux locaux ;

* le mode d'accès dynamique attribut à chaque entité, un temps d'accès de manière variable suivant les besoins.

La suite de l'exposé précise quelques méthodes parmi les plus utilisées d'accès dynamique à un réseau.

A-2-1) *Accès aléatoire.*

La méthode d'accès est différente suivant la topologie adoptée (bus ou anneau).

A-2-1-1) Réseau à détection de porteuse (topologie bus) :

L'entité communicante écoute l'état du canal de transmission (libre ou occupé) avant d'essayer de transmettre. Il reste cependant des risques de collisions dues aux temps de propagation des informations sur le réseau. Lors d'une détection de conflit, une nouvelle date de transmission pour chaque entité est déterminée de manière probabiliste (C.S.M.A./C.D.) ou déterministe (C.S.M.A./D.C.R.). Une méthode très répandue est la méthode C.S.M.A./C.D. (Carrier Sense Multiple Acces/ Collision Detection) utilisée dans Ethernet qui est un classique des réseaux locaux.

A-2-1-2) Méthode de la tranche vide (topologie en boucle) :

C'est une technique d'accès sur boucle mono-directionnelle consistant à faire tourner en permanence sur la boucle des trames qui possèdent un indicateur "plein / vide".
Pour émettre, une station doit attendre la réception d'une trame vide qu'elle remplit et positionne sur "plein". Le destinataire réceptionne la trame et repositionne l'indicateur sur "vide".
Cette méthode nécessitant un nombre important de stations pour être efficace n'est pas utilisée pour les réseaux locaux.

A-2-2) *Les accès contrôlés.*

A-2-2-1) La gestion centralisée.

L'accès au réseau est géré par une station appelée "maître", les autres stations étant des "esclaves". Cette gestion est effectuée par une interrogation périodique de la part de la station maître, des stations esclaves (Uni telway, Jbus). Cette méthode offre l'avantage de faciliter la réalisation des fonctions à mettre en oeuvre mais a comme inconvénient de rendre chaque station esclave dépendante de la station maître. Pour parer cet inconvénient, la technique du maître flottant (Telway 7) affecte le rôle de maître à la première station initialisée sur le réseau. Ces méthodes de gestion centralisées sont très souvent utilisées dans les réseaux d'automates programmables ou les réseaux dit de "terrain"

A-2-2-2) La gestion décentralisée.

Elle est basée sur la notion "de droit à la parole" que les stations de l'anneau se passent les unes aux autres sous la forme d'une trame particulière appelée "jeton".
Chaque station qui reçoit le jeton :

- émet si besoin est, les informations qu'elle doit transmettre, puis émet à nouveau un jeton vers la station suivante ;

- passe le jeton à la station suivante, si elle ne désire pas émettre d'informations.

De plus, chaque station peut être affectée d'une certaine priorité d'émission et d'un temps maximum d'occupation de la ligne de transmission.
Dans le domaine industrielle, deux méthodes d'accès par jeton sont particulièrement répandues :

* Topologie en bus (le réseau de type Map) :

Le principe de fonctionnement consiste à faire circuler le jeton dans le sens des adresses décroissantes des stations. La station la plus basse transmettant le jeton à celle la plus haute pour boucler l'anneau qui est ici un anneau virtuel, (cf. Fig 1-14). Cette technique est aussi appelée technique du jeton virtuel.

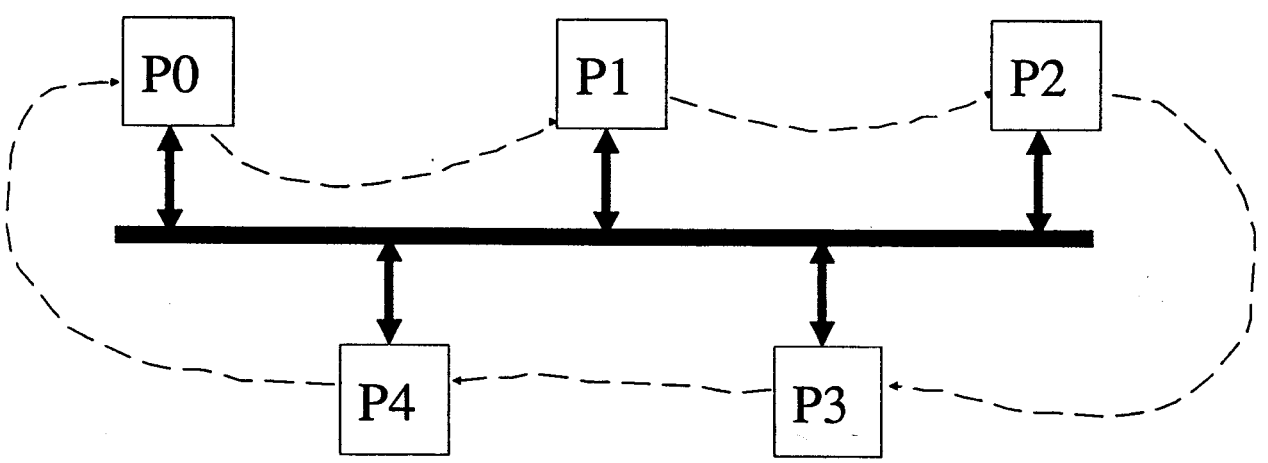

FIG. A-3 : Anneau virtuel

* Topologie en anneau (ou boucle) :

La technique consiste a faire circuler sur l'anneau une trame appelée jeton dans un sens défini (fonctionnement mono-directionnel). Chaque station étant reliée, de part la configuration physique (boucle), à une station qui la précède et à une station qui la suit, l'ordre des stations dans lequel est transmis le jeton étant parfaitement défini.

Exemple :

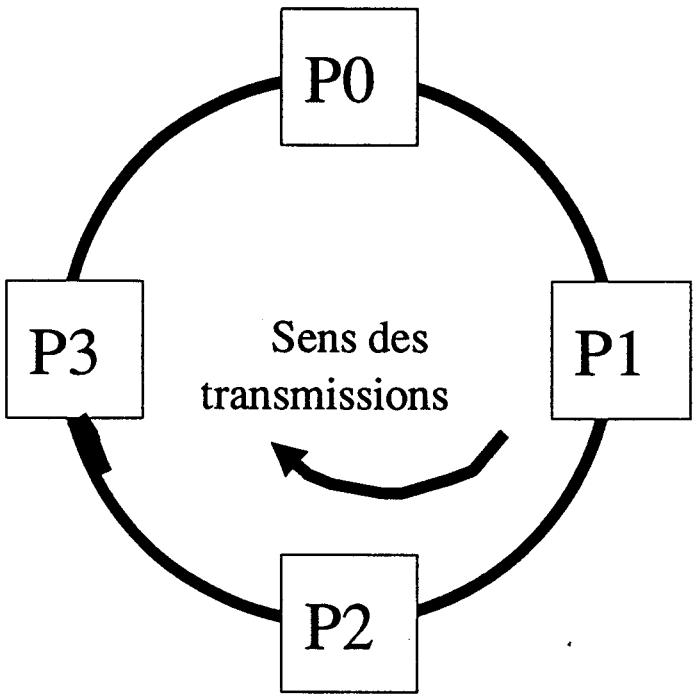

FIG. A-4 : Réseau en anneau

A-3) La normalisation [LEP 89]

Un modèle de structuration du protocole pour l'interconnexion des systèmes ouverts (O.S.I.) a été défini par l'organisme de normalisation internationale (I.S.O.). Ce modèle définit sept niveaux (ou couches) de fonctions devant être assurées pour établir une communication entre utilisateurs. Chaque couche fournit des services à la couche immédiatement supérieure. Pour cela, elle utilise les services de la couche immédiatement inférieure et y ajoute les siens.
Les sept couches sont les suivantes :

* **Application** : cette couche définit l'ensemble des requêtes communes aux applications réparties.

* **Présentation** : cette couche assure l'échange des données entre applications de façon à assurer une certaine compatibilité des équipements vis à vis du réseau.

* **Session** : cette couche permet une gestion des dialogues entre entités communicantes.

* **Transport** : cette couche permet le transport d'information d'un bout à l'autre entre deux entités communicantes. Elle vérifie la présence des paquets et les rassemble pour reconstituer les messages qui ont été envoyés.

* **Réseau** : cette couche a pour fonction d'acheminer les paquets. Elle assure le routage (choix du chemin jusqu'au destinataire) des informations.

* **Liaison** : Cette couche effectue l'acheminement sans erreur des blocs d'informations (ou trames) entre des systèmes interconnectés.

* **Physique** : cette couche représente les interfaces physiques (support de transmission, connecteurs, dérivations,...) utilisés pour la transmission.

Exemple :

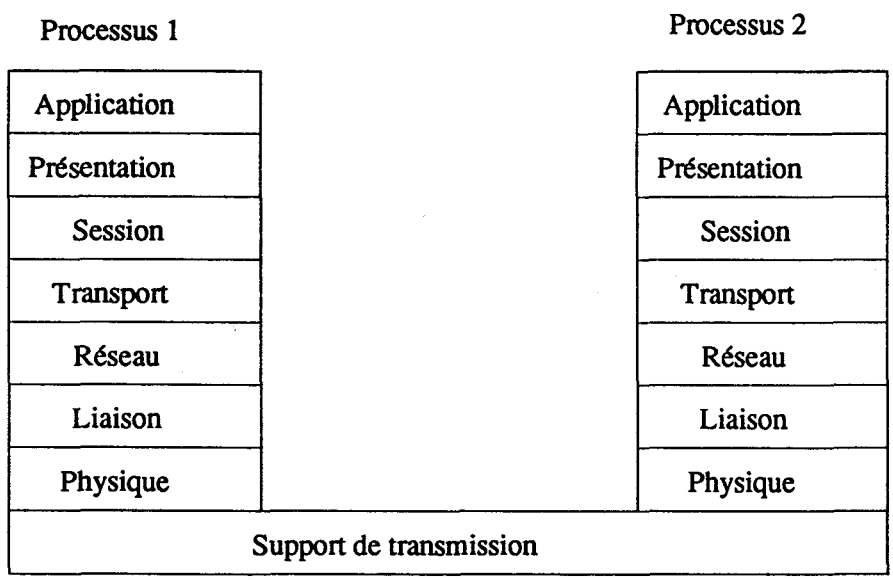

FIG. A-5 : Le modèle O.S.I.

Suivant le type d'application et suivant les performances attendues, l'utilisateur peut n'utiliser que certaines couches prédéfinies.