Protocoles multipoint et interconnexion de réseaux FIP

Ghassan Saba

Institut National Polytechnique de Lorraine

Département de Formation Doctorale en Informatique — École Doctorale IAE + M

Protocoles multipoint et interconnexion de réseaux FIP

THÈSE

présentée et soutenue publiquement le 26 Janvier 1996

pour l'obtention du

Doctorat de l'Institut National Polytechnique de Lorraine

(Spécialité Informatique)

par

Ghassan SABA

Composition du jury

Président :	Jean-Jacques PANSIOT, Université Louis Pasteur, Strasbourg
Rapporteurs :	Abd-El-Kader SAHRAOUI, LAAS, Toulouse André SCHAFF, CRIN, Nancy
Examinateurs :	Zoubir MAMMERI, ENSAM, Châlone-sur-Marne Jean-Pierre THOMESSE, directeur de thèse, CRIN-ENSEM, Nancy

Centre de Recherche en Informatique de Nancy

Remerciements

Le travail présenté dans ce mémoire a été effectué au sein de l'équipe Informatique Industrielle du CRIN-INPL.

Je tiens à exprimer ma profonde gratitude à M. Jean-Pierre Thomesse, directeur de thèse, pour m'avoir accueilli dans son équipe et pour l'aide qu'il m'a apportée.

Je remercie vivement M. Zoubir Mammeri pour ses conseils et ses efforts qui m'ont permis de valoriser mes résultats .

Je remercie également Mme Françoise Simonot-Lion, M. Ye-qiong Song et M. Michel Dufner pour leur assistance sans faille.

Je remercie MM. Jean-Jacques Pansiot (Université Louis Pasteur, Strasbourg), Abd-El-Kader Sahraoui (LAAS, Toulouse) et André Schaff (CRIN, Nancy) qui m'ont honoré en acceptant de participer au jury de ma thèse et qui m'ont aidé à améliorer ce document.

Je remercie tous mes collègues ainsi que Mme. Micheline Dalbourg, secrétaire de l'équipe, pour leurs encouragements, leur collaboration ainsi que pour la bonne ambiance qu'ils ont su maintenir.

Un remerciement est adressé enfin à M. Sami Al-Khiyami pour son aide indirecte mais indispensable.

Table des matières

Liste des figures

Liste des tableaux

Introduction générale

Nous présentons dans ce mémoire une contribution au problème de routage multipoint ainsi que des solutions pour l'interconnexion de réseaux de terrain FIP (Factory Instrumentation Protocol) par l'intermédiaire de ponts.

FIP est un réseau de terrain qui est conçu principalement pour faire communiquer ou assurer l'échange de données entre capteurs, actionneurs et les différents équipements de commande et de contrôle tel que les automates programmables et les régulateurs. Les réseaux de terrain sont généralement caractérisés par la connaissance préalable, lors de l'installation de l'application, du trafic échangé. Cette connaissance concerne non seulement la source et le(s) destinataire(s) du trafic mais aussi des paramètres temporels qui peuvent concerner la fréquence d'échange et la cohérence des données issues de la répartition de l'application sur plusieurs sites. Les données échangées sont en général courtes (quelques octets), ont des durées de validité assez faibles (quelques dizaines de milli-secondes), utilisent des mécanismes d'adressage logique par objets permettant un échange multipoint indépendamment de la localisation physique. Nous y retrouvons par ailleurs des services traditionnels de messages, avec ou sans acquittement, point à point ou multipoint qui fournissent un moyen pour le diagnostic ou la gestion de configuration.

Il se trouve que l'installation d'un seul bus FIP est devenue insuffisante pour répondre à tous les besoins de communication. Ces besoins peuvent concerner (i) l'évolution de l'application en terme de : nombre maximal des identifieurs échangés sur un bus, nombre maximal de stations connectées à un bus et l'étendue géographique (ii) la performance d'un bus en terme de trafic variables, messages et liste de variables apériodiques. Pour faire face à ces exigences, nous avons besoin des équipements spéciaux d'interconnexion qui offrent en plus un moyen pour maintenir la connectivité sur l'ensemble du réseau et pour permettre l'échange de données entre entités qui n'appartiennent pas au même bus. Par ailleurs, ces équipements d'interconnexion que nous allons appeler ponts doivent considérer quelques problèmes inhérents à la spécificité du réseau FIP, notamment :

- la gestion d'accès au bus n'est pas séparée du contrôle logique de liaison,
- la communication est en mode multipoint,

- la cohabitation de deux modes d'adressage complètement différents (l'adressage objet et l'adressage station),
- la coexistence de plusieurs types de trames (objets identifiés, messages, listes d'objets identifiés, ...) sur le médium; ces trames pouvant être adressées à un ou plusieurs destinataires explicitement ou implicitement désignés,
- l'existence de deux types de trafics, l'un périodique, l'autre apériodique, tous les deux étant applicables aux objets identifiés et aux messages,
- l'accès au médium est centralisé et géré par une station dite arbitre de bus qui doit connaître, avant le démarrage de l'application, les périodes de tous les trafics périodiques,
- l'absence de routage de bout en bout.

Les raisons précédentes ont rendu les solutions classiques des ponts inadaptées et nous ont conduits, par conséquent, à reconsidérer le problème en entier.

Puisque l'objectif principal du travail mené dans ce mémoire est l'interconnexion de réseaux FIP par l'intermédiaire de ponts agissant habituellement au niveau de la sous-couche MAC (Medium Access Control), nous avons été amenés à structurer, dans un premier temps, la couche Liaison de FIP en deux sous-couches : MAC et LLC (Logical Link Control). De cette structuration, nous avons déduit des solutions d'interconnexion de réseaux FIP. Ces solutions tiennent compte de la spécificité de ce réseau et définissent les ressources nécessaires et les fonctionnalités minimales pour la mise en œuvre d'un pont.

Les mécanismes décrits informellement peuvent comporter des erreurs fonctionnelles et/ou des incohérences. Aussi, nous avons utilisé des techniques de spécification formelle et de vérification pour valider notre proposition de structure de la couche Liaison de FIP ainsi que le comportement du pont.

Il convient de noter enfin que les solutions définies utilisent des tables de routage qui ont été mises à la disposition de chaque pont. Ces tables peuvent être construites à la main lorsque la topologie d'interconnexion est assez simple. Mais, par contre, des problèmes se posent lorsque la topologie d'interconnexion devient maillée et fait apparaître des chemins multiples, par exemple le problème de prévention debouclage et d'optimalité de chemins. Pour ce faire, nous avons développé un algorithme de routage multipoint bien adapté au trafic objet multi-destinataires. Cet algorithme nous permet d'établir les tables de routage dans les différents ponts de l'interconnexion de façon à garantir le non-bouclage et à optimiser les délais et le nombre de ponts traversés.

Organisation du mémoire :

La structuration du reste de ce mémoire est réalisée en deux parties :

PARTIE I

Chapitre 1 : *Interconnexion des réseaux.* Nous présentons un état de l'art consacré aux solutions classiques d'interconnexion des réseaux. L'accent est mis sur les différents niveaux d'interconnexion, les terminologies utilisées, les solutions OSI, les stratégies d'adressage et de routage, les algorithmes d'extension des réseaux locaux (utilisés souvent dans les ponts) et les algorithmes de routage. Le reste de ce chapitre concerne l'adressage de groupe et les algorithmes existants de routage multipoint.

Chapitre 2 : *Présentation du réseau de terrain FIP.* Nous introduisons l'architecture générale, les principaux services rendus par la couche application et le modèle PDC. Nous concentrons notre présentation sur la couche liaison de données qui relève du fonctionnement du pont, notamment la gestion d'accès, les services offerts et l'adressage.

PARTIE II

Chapitre 3 : *Structuration de la couche liaison de FIP.* Nous nous intéressons à l'architecture de SAP/CEPi dans FIP et l'identification de LLC/MAC. L'architecture de SAP/CEPi est importante pour identifier les différentes entités LLC d'une façon unique et non ambiguë et pour définir les extrémités de connexions à travers ces SAPs. Tandis que l'identification de LLC/MAC nous permet de séparer l'accès au bus du contrôle logique de liaison pour pouvoir spécifier correctement la structure d'un pont.

Chapitre 4 : *Interconnexion de réseaux FIP.* Nous proposons deux types de solutions qui permettent la résolution du problème d'adressage ainsi que la définition des ressources nécessaires et du fonctionnement interne du pont.

Chapitre 5 : *Spécification formelle d'un réseau FIP connecté à un pont.* Une spécification formelle par l'intermédiaire des réseaux de Petri Étiquetés Prédicat-Transition fournie par l'outil EVAL nous a permis de vérifier (qualitativement) le comportement attendu de notre pont en interaction avec la nouvelle structuration du réseau FIP. Nous avons tenu compte dans cette spécification de différents SAPs qui ont été définis à l'interface entre le MAC et LLC évoquée précédemment.

Chapitre 6 : *Algorithme de routage multipoint de plus courts délais et de coût minimal.* Nous proposons un algorithme de routage multipoint que nous avons développé et qui cherche à optimiser le nombre de ponts intermédiaires tout en conservant les plus courts chemins entre une source spécifique et un groupe de destinataires. Cet algorithme nous permet de construire un arbre d'acheminement par objet identifié de FIP où la source est le producteur, et le groupe de destinataires est l'ensemble des consommateurs. Il permet aussi de construire un arbre d'acheminement par couple (source, groupe) pour le trafic de message multipoint.

La conclusion et les perspectives du travail concluent ce mémoire.

PARTIE I

CHAPITRE 1 : Interconnexion des réseaux

CHAPITRE 2 : Présentation du réseau de terrain FIP

Chapitre 1

Interconnexion des réseaux

1.1. Introduction

La prolifération des réseaux et l'impossibilité de construire un réseau universel à partir d'une technologie unique qui répond à tous les besoins utilisateurs, font que l'interconnexion des réseaux gagne une importance primordiale. Nous ajoutons ainsi qu'une seule installation d'un réseau devient, tôt ou tard, insuffisante parce qu'elle impose des limitations sur :

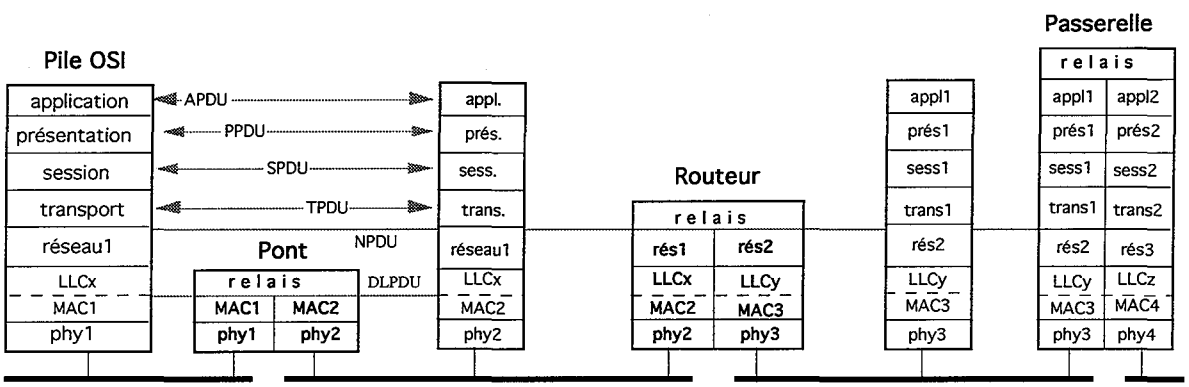
- le nombre de stations qui peuvent être connectées à un segment
- la longueur physique et l'étendue géographique d'un tel support
- la performance d'un réseau
- l'espace d'adressage d'un réseau

Pour ce faire, nous avons besoin d'équipements spéciaux pour maintenir la connectivité entre les réseaux de façon à préserver tant que possible les propriétés essentielles de chaque réseau (ou chaque segment appartenant à un réseau). Par conséquent, cette interconnexion ayant pour objectif de former un inter-réseau peut être achevée par plusieurs méthodes.

Dans tous les cas, le succès d'une telle stratégie d'interconnexion est fonction de son adaptation de l'autonomie et de la différence de chaque réseau individuel ainsi que de sa tolérance d'un partage maximal des données [Sunshine 90]. A ce moment là, il faudrait bien choisir la localisation des entités relais (soit centralisée dans une artère "backbone" soit répartie dans les sous réseaux [Roffinella 87]) et de sa fonctionnalité (toutes les fonctions nécessaires pour assurer l'interconnexion doivent être rassemblées dans les entités relais ou une participation des stations terminales peut être envisagée).

1.2. Niveau d'interconnexion

La solution la plus adéquate pour remédier au besoin d'interconnexion consiste à remonter dans la pile OSI[1] [Zimmermann 80] jusqu'au premier niveau d'incompatibilité. Par conséquent, on peut recourir à l'utilisation des ponts (niveau 2) dans des réseaux ayant la même structure d'adressage, le même taux d'erreurs et la même bande passante [Perlman 88]. Et utiliser ensuite des routeurs (niveau 3) lorsque des incompatibilités d'adressage ou des besoins de segmentation/assemblage, contrôle de congestion apparaissent indispensables. On peut recourir à utiliser des passerelles (niveau >3) pour convertir des protocoles (e.g., d'un protocole de communication en mode avec connexion vers un autre en mode sans connexion). La figure suivante montre le niveau d'opération de chaque type d'équipement par rapport au modèle OSI. Il convient de noter ici que certaines couches peuvent ne pas être présentes dans la pile comme par exemple si on utilise une passerelle entre un réseau conforme à l'ISO et un autre utilisant TCP/IP[2]; Pour ce dernier, pas de correspondance pour les couches session et présentation.



PDU: Protocol Data Unit
MAC: Medium Access Control
LLC: Logical Link Control
DL: Data Link

Fig. 1.1 — Pont, routeur et passerelle vis à vis du modèle OSI

Nous traitons dans la suite les différents niveaux d'interconnexion ainsi que les caractéristiques principales de chaque type d'équipement.

1.2.1. Relais de niveau couche Physique

Les équipements d'interconnexion opérant à ce niveau sont généralement appelés *répéteurs (repeaters)*. Un répéteur amplifie et redirige les signaux (bits) lorsque des limitations physiques les empêchent de se propager plus loin (e.g., distance maximale) ou lorsque les signaux ont besoin de changer de médium [Sunshine 90, Burg 89]. Dans tous les cas, le réseau résultant se comporte comme un réseau unique.

1 Open Systems Interconnection,
2 Transmission Control Protocol/Internet Protocol,

1.2.2. Relais de niveau couche Liaison de données

Les équipements d'interconnexion de ce niveau, et plus particulièrement du niveau MAC[3], sont connus comme des *Ponts (Bridges)*. Les ponts ont l'aptitude de prendre des décisions intelligentes pour laisser passer le trafic ou non (filtrer) : selon IEEE[4] 802, leurs objectifs consistent à avoir une interconnexion de RL[5] lorsque des limites géographiques se manifestent ou pour augmenter le nombre de stations ou la bande passante.

On rencontre souvent ce type de dispositif dans les protocoles dits non routables. Dans ce type de protocoles, nous ne pouvons pas reconnaître la localisation physique d'une entité terminale à partir de son adresse. C'est le cas des protocoles IEEE 802 utilisant des adresses physiques plates qui sont fixées auparavant par les manufacturiers. Cela a donné naissance aux ponts transparents [IEEE 802.1] ainsi qu'aux ponts utilisant la stratégie de routage par l'émetteur [IEEE 802.5].

Les ponts transparents : ils n'exigent aucune participation de la part des stations terminales que l'on ne modifie pas pour la mise en marche du pont [Backes 88]. Ce dernier a l'aptitude de s'initialiser, de se configurer ainsi que de s'exécuter automatiquement sans l'intervention d'un gestionnaire du réseau *(plug-and-play bridges)*.

Les opérations principales effectuées dans le pont comportent :

- aiguillage des trames,
- apprentissage des adresses des stations,
- utilisation de l'algorithme de l'arbre recouvrant *(Spanning Tree)* [Perlman 85] pour éliminer les éventuelles boucles de la topologie d'interconnexion.

Les ponts utilisant le routage par l'émetteur (source routing) : L'émetteur reconnaît le chemin pour atteindre le destinataire par une procédure d'apprentissage. Cette procédure fait souvent appel à une inondation des trames à travers tous les chemins possibles vers le destinataire pour pouvoir décider le meilleur chemin selon des critères de distance, de charge ou des coûts de liaisons.

L'émetteur doit inclure ce chemin, qui est composé d'une suite ordonnée de paires d'identifieurs (pont, segment), dans chaque trame envoyée. Cela réduit la tâche du pont à une simple comparaison de chaînes de caractères (similaire à celle de reconnaissance d'adresses) lui permettant de prendre une décision pour laisser passer la trame ou l'ignorer.

Les deux types de ponts sont reconnues dans les normes, soit [IEEE 802.1] pour les ponts transparents [Backes 88], soit [IEEE 802.5] comportant le routage par l'émetteur [Dixon 88]. Selon [Soha 88], les ponts transparents offrent une bonne solution en terme de robustesse, stabilité, efficacité, déterminisme et simplicité mais le chemin parcouru n'est pas forcément optimal. Le routage par l'émetteur par contre permet d'avoir un chemin proche de l'optimal et un partage équitable de charge sur les réseaux. Il complique, en revanche, la tâche de l'installation. Il ajoute des coûts supplémentaires à toutes les stations et peut provoquer une saturation des réseaux par des nombres de copies des trames exponentiels au numéro de ponts. Une structure d'un pont interconnectant toutes les combinaisons possibles des MACs de l'IEEE 802, à savoir 802.3, 802.4 et 802.5 a été proposée dans [Berntsen 85] et pour laquelle des conversions portant non seulement sur les

3 Medium Access Control,
4 Institute of Electrical and Electronics Engineers,
5 Réseaux Locaux,

structures des trames mais aussi sur les différents types de contrôle d'accès ainsi que la gestion d'erreurs, ont été élaborées.

Dans tous les cas, les deux types de pont ne favorisent pas la notion de communication multipoint ou l'adressage de groupe. Les ponts transparents tendent vers l'inondation des trames sur tous les ports à chaque fois que l'adresse destinataire est une adresse de groupe parce que l'apprentissage d'adresse n'est plus applicable à moins que le gestionnaire du réseaux ait à définir explicitement les différents groupes. Dans le routage par l'émetteur, le problème devient plus sophistiqué car un problème de bouclage se manifeste dont les solutions possibles sont l'algorithme de l'arbre de recouvrement ou la diffusion restreinte à une route donnée [Dixon 88].

Il faut distinguer enfin deux types de ponts : pont local et pont distant [IEEE 802.6]. Dans le premier, les MACs identiques ou différents sont directement interconnectés. Les deux moitiés de pont se trouvent à la même localisation physique et il n'y a pas de médium d'interconnexion distinct entre eux. Alors que dans le pont distant (remote bridge), les deux moitiés du pont sont séparées et interconnectées par un médium de communication (FDDI, X.25, ISDN, etc.). Ce type de pont est souvent appelé demi-pont. Ceci est bien sûr applicable pour les routeurs et les passerelles.

1.2.3. Relais de niveau couche Réseau

La couche réseau est principalement concernée par le routage de bout en bout mais elle peut également, selon qu'elle est en mode avec ou sans connexion, se préoccuper du contrôle de congestion, de la gestion des erreurs ou de la segmentation/réassemblage des paquets.

Puisque la vocation primaire des équipements d'interconnexion de ce niveau est le routage de bout en bout, un tel équipement est appelé *Routeur (Router)* [Sunshine 90]. Un routeur peut, en plus, sélectionner un chemin parmi plusieurs selon des critères de congestion sur les autres routeurs, de délais de transmission ou tout simplement selon la longueur du chemin [Seifert 88].

Nous signalons ici qu'il existe des équipements du type hybride pont-routeur (Brouter) [Stephenson 91] qui assurent aussi bien les fonctionnalités des ponts que celles de routeurs. Un pont-routeur est capable, à la fois, d'acheminer les paquets utilisant des protocoles routables et de réexpédier, voir diffuser, le reste selon les principes des ponts. Un pont-routeur qui a l'inconvénient d'être complexe, coûteux et difficile à installer peut rendre des services importants dans des réseaux hétérogènes, notamment, la transparence de protocoles.

1.2.4. Relais de niveau couches 4 à 7

Les équipements opérant à ce niveau sont appelés Passerelles (Gateways).

Le rôle principal d'une passerelle est de transformer les paquets d'un protocole vers un autre [Sunshine 90], [Corr 91], [Cantrell 92].

Une passerelle a l'accès à toutes les entêtes engendrées par toutes les couches. Elles se situent souvent soit au niveau transport, par exemple entre TCP et ISO TP4 [Svobodova 90], soit au niveau Application [Zatti 88].

Il existe plusieurs critères qui peuvent influencer le choix d'un tel équipement. Bien que le niveau d'opération peut servir pour en éliminer quelques uns mais il n'est pas suffisant pour effectuer le choix. Pour ce faire, il faut considérer d'autres critères liés, soit aux utilisateurs finaux, soit à la fonctionnalité fournie, soit enfin aux limites de chaque équipement.

En ce qui concerne l'utilisateur final, des critères tels que la transparence, la difficulté d'installation, le coût, la sécurité et enfin l'auto-adaptation aux modifications sont d'une grande importance.

Les critères liés à la fonctionnalité peuvent concerner l'isolement de défaillances, la segmentation/réassemblage des messages, la résolution d'incompatibilité d'adresses, la reprise sur erreurs et l'intelligence.

Enfin, les limites de chaque type d'équipement peuvent se traduire en terme de taille du réseau connecté par cet équipement, de différence de débits sur chaque interface, de la répartition de charges sur les liaisons de sorties ainsi que du nombre de protocoles qui peuvent être traités.

Ces critères sont illustrés dans le tableau suivant [Corr 91], [Cali 94], [Perlman 88]:

caractéristique	**répéteur**	**pont**	**routeur**	**passerelle**
niveau	physique	MAC	réseau	≥ transport
complexité	petite	moyenne	grande	très grande
temps de transit	négligeable	petit	moyen	grand
transparence à l'utilisateur	oui	oui	non	non
auto-configuration	oui	pont transparent	non	non
reconfiguration des stations	non	non	oui	oui
isolement de défaillances	non	oui	oui	oui
incompatibilité d'adresses	non	non	oui	oui
segmentation/réassemblage	non	non	oui	oui
débits hétérogènes sur les deux interfaces réseau	congestion des trames	congestion des trames	contrôle de congestion	contrôle de congestion
test de checksum	non	oui	oui	oui
reprise sur erreurs (duplication et désordre)	non	non	oui	oui
limites sur la taille du réseau	non	oui	non	non
utilisation efficace de la bande passante	non	non	oui	oui
contrôle de cycle de vie des données	non	non	oui	oui
intelligence	non	pont transparent	oui	oui
conversion de protocoles	non	non	niveau réseau	oui
sécurité	non	non	possible	excellente
difficulté d'installation	non	non	oui	plus difficile
acquittement de bout en bout	non	non	non	oui
filtrage des trames utiles	non	oui	oui	oui
coût	petit	moyen	grand	très grand

Tableau 1.1 — Les caractéristiques principales des différents types de relais

Nous introduisons dans la suite le nommage, l'adressage et le routage. Les utilisateurs finaux préfèrent communiquer avec leurs partenaires par l'intermédiaire des noms. La couche d'interconnexion se sert de l'adressage pour pouvoir garantir une identification

unique et non ambiguë du partenaire, elle utilise ainsi le routage pour choisir le meilleur chemin pour atteindre ce partenaire.

1.3. Nommage, Adressage et Routage

Pour bien acheminer les données à leur destination exacte, on doit distinguer entre nommage, adressage et routage.

Cette distinction est définie dans la littérature par:

- Un nom identifie un objet
- Une adresse identifie sa localisation
- Une route est un chemin pour l'atteindre

Compte tenu des recommandations ITU-T[6] (ex-CCITT) [X.500], on peut définir les termes suivants:

• *Nommage* : Un nom sert à identifier la machine hôte "logiquement" indépendamment de ses points d'attachement au réseau. Par ce fait, une machine hôte peut avoir plusieurs noms simultanément.

• *Adressage* : une méthode à trouver pour identifier d'une façon unique toutes les interfaces d'un système inter-réseaux.

Plusieurs méthodes peuvent être utilisées pour combiner deux espaces d'adressage ayant une intersection non vide. La première consiste à fusionner un espace dans l'autre en transformant toutes ses adresses dans des adresses non utilisées par l'autre. La deuxième transforme les adresses non utilisées de chaque réseau en des adresses souhaitées de l'autre.

On peut aussi se ramener à encapsuler les trames d'un réseau "A" dans sa totalité et l'insérer dans le champ données des trames à générer pour le réseau "B" [Varghese 90]. Cette stratégie pose deux problèmes majeurs : (i) l'utilisateur de chaque réseau sera capable de voir, manipuler et même comprendre les formats et la signification des adresses de l'autre réseau (ii) cela peut augmenter la taille des trames et la rendre impossible à encapsuler localement.

La stratégie la plus générale introduit des adresses hiérarchiques où un préfixe explicite de chaque réseau est ajouté aux suffixes locaux existants pour former une adresse complète. Cette approche provoque des modifications sur le protocole original ou sur la structure des trames qui doivent accepter la forme étendue des adresses.

• *Routage :* une fois la stratégie d'adressage choisie, le problème de routage se pose toujours : comment peut-on atteindre le destinataire à partir de son adresse? Dans la plupart des ponts où des adresses plates (par opposition aux adresses hiérarchiques) sont utilisées, des chemins concernant chaque destinataire sont maintenus dans des listes. Ces ponts doivent implémenter une procédure de routage qui évite la duplication ou l'inondation lorsqu'il existe des chemins multiples à travers les réseaux interconnectés, et trouver ainsi toutes les destinations efficacement.

Pour ce faire, diverses approches existent, la première est basée sur l'arbre de recouvrement (Spanning tree) minimal à travers un inter-réseaux. La deuxième approche qui utilise le routage par l'émetteur (source routing) est adaptée aux RLs (LAN) de type

6 International Telecommunication Union,

anneau à jeton. La troisième approche découvre et établit un chemin optimal dès que le destinataire est connecté. Elle envoie des séquences de paquets à travers ce chemin sans recourir à des calculs dans le pont.

Lorsqu'on utilise des adresses hiérarchiques, comme un protocole d'interconnexion de réseaux, le routage sera effectué en étapes du réseau initial jusqu'au final. Ceci réduit énormément la taille des tables de routage mais affecte l'optimalité [Sunshine 90].

On peut implémenter enfin des procédures de routage de type hybride (e.g., communicateurs de téléphone) pour avoir un bon compromis entre les deux types d'adressage (hiérarchique et plat).

1.4. Solutions OSI

Selon l'OSI, l'interconnexion des réseaux survient lorsque les données ont besoin de traverser deux ou plusieurs réseaux ayant quelques différences fondamentales [Burg 89]. Ces différences peuvent concerner les espaces d'adressage local à chaque réseau. Par exemple, deux réseaux identiques qui ont deux espaces d'adressage indépendants ont besoin d'être interconnectés.

Les termes : *relais, système intermédiaire et unité d'interconnexion* sont fréquemment utilisés par l'OSI pour représenter l'interconnexion de réseaux [ISO DP 10028] :

Un relais n'opère pas comme source ou comme destinataire de données, mais il maintient la cohérence de la sémantique de données qui arrivent ou qui partent. Ce terme peut être utilisé indépendamment de la couche.

Un système intermédiaire (SI) est un terme abstrait utilisé pour identifier les entités d'un environnement OSI qui effectuent seulement la fonctionnalité des trois couches basses du modèle de référence OSI.

Une unité d'interconnexion (UI) est une composante réelle d'équipement qui réagit comme un système intermédiaire pouvant interconnecter plusieurs sous-réseaux.

Parmi les fonctions reconnues par l'OSI, il y a la fonction de relais. Cette fonction est fournie par un système intermédiaire qui rassemble un groupe de procédures par lesquelles le système redirige les données d'un système vers un autre.

De plus, l'OSI définit les principaux rôles à considérer dans une telle interconnexion. Il s'agit notamment de :

- définir le service rendu par la couche réseau vers la couche transport (soit orienté connexion CONS[7] soit sans connexion CLNS[8]). Ce service doit maintenir l'indépendance de chaque type de sous-réseau ou topologie, etc.
- spécifier une adresse unique dans l'environnement OSI.
- la sélection du chemin pour atteindre la destination.
- l'hétérogénéïté des sous-réseaux traversés de la source vers le destinataire.

7 COnnection-mode Network Service,
8 ConnectionLess-mode Network Service,

1.4.1. L'adressage selon l'OSI

Dans l'environnement OSI, chaque entité (N) est par définition un élément actif qui implémente des fonctions de couche (N) ainsi que des protocoles de communication avec une entité homologue [Poo 90]. Elle doit être identifiée globalement et d'une façon unique. L'attribution d'une chaîne de chiffres à une telle entité pour l'identifier sans ambiguïté est connue comme l'adressage [Burg 89].

Pour bien localiser une entité OSI dans la pile des couches afin de lui permettre une communication avec soit une entité homologue, soit des entités adjacentes, l'OSI a introduit la notion du point d'accès aux services SAP[9](N). Un SAP(N) symbolise l'interface permettant à une entité (N+1) d'accéder aux services offerts par une entité N [ISO 7498-3]. Il est identifié globalement par une adresse(N) (Fig. 1.2).

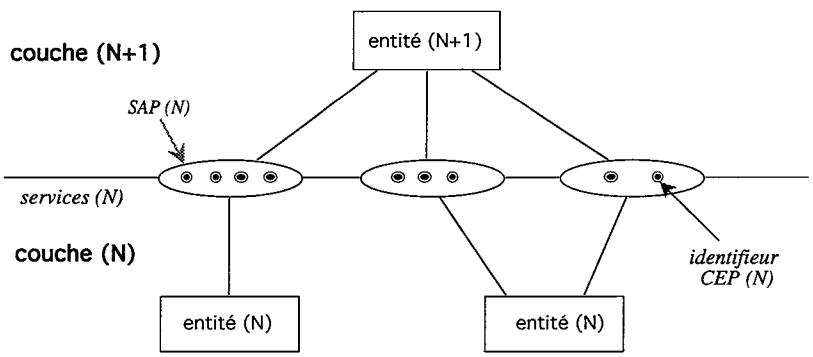


Fig. 1.2 — L'interface entre deux couches adjacentes

Le service à travers un SAP(N) doit être fourni par une seule entité(N) et utilisé par une seule entité(N+1). Par contre, une entité(N) peut servir plusieurs SAP(N) et de même, une entité (N+1) peut utiliser plusieurs SAP(N) [Svobodova 90].

Ce type d'adressage exige un mécanisme de correspondance d'adresse de SAP(N) à travers les couches. Pour ce faire, trois types de correspondances, comme le montre la figure 1.3, sont définis : Le premier est du type *une-à-une* impliquant une translation directe, le deuxième est du type *hiérarchique* où l'adresse de SAP(N) est le résultat de la concaténation de l'adresse de SAP(N-1) et le sélecteur(N). Le troisième type utilise une *table de correspondance.*

Dans le cas où plusieurs transactions doivent avoir lieu de façon concurrente sur le même SAP, chaque couche a la responsabilité de reconnaître localement l'identité de chaque transaction et de la relier à un message donné.

Cette fonction ne fait pas partie d'adressage de SAP et est effectuée par des identifieurs de CEP[10] seulement utilisés dans le transfert avec connexion. Dans ce cas là, chaque entité invoquée dans le transfert doit enregistrer les deux adresses appelante et appelée ainsi que l'état du protocole relativement à chaque connexion.

9 Service Access Point,
10 Connection End Point,

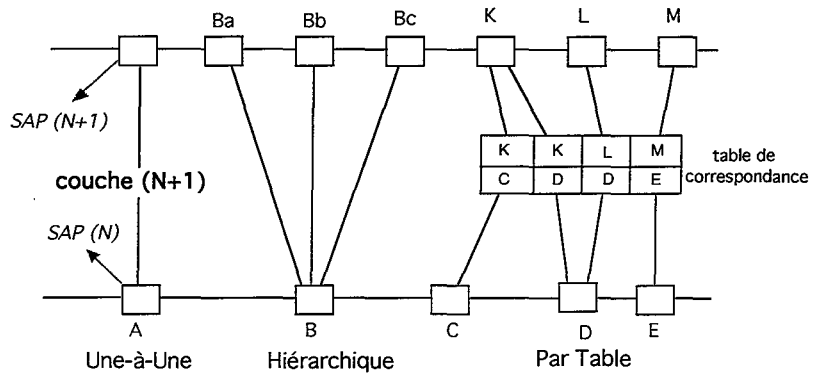


Fig. 1.3 — La correspondance entre les adresses de SAPs

En ce qui concerne les champs d'adresses source et destinataire des messages échangées entre les différentes entités, c'est l'adresse de NSAP ou de niveau réseau qui sera utilisée.

Pour les formats d'adresse, l'OSI a choisi la forme hiérarchique [ISO 8348/AD1]. Dans cette approche, l'adresse OSI comporte deux parties : la partie initiale de domaine IDP[11] et la partie spécifique du domaine DSP[12] (Fig. 1.4). L'IDP comprend l'autorité qui définit le format d'identifieur AFI[13] et le champs d'identifieur du domaine initial IDI[14]. Le code AFI spécifie les formats de l'IDI, l'autorité administrée de la valeur de l'IDI et la syntaxe abstraite du DSP. Le code IDI spécifie le domaine local d'adressage ainsi que la sémantique du DSP.

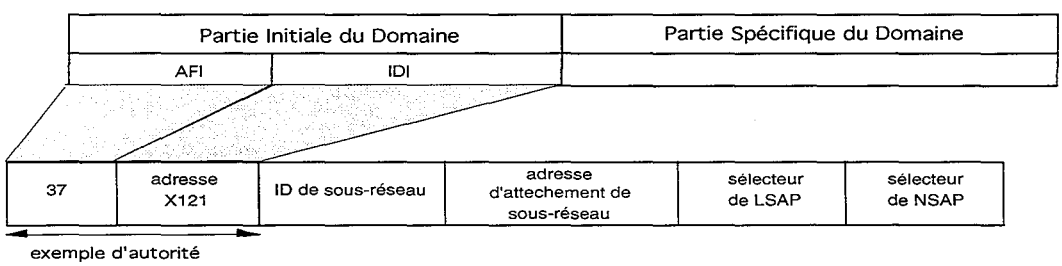


Fig. 1.4 — Les formats généraux et spécifiques de l'adressage OSI

Nous retrouvons dans [Zwecker 93] une étude approfondie concernée à l'adressage ISO. Des solutions permettant de remédier aux problèmes d'incompatibilité d'adresses et des couples service-protocole lors de l'inerconnexion des réseaux ont été également proposées. Le travail de [Faure 90] peut servir comme point de départ pour le problème d'interconnexion de réseaux hétérogènes en général et les concepts des frontières d'incompatibilité en particulier.

1.4.2. Le routage selon l'OSI

L'OSI rappelle qu'il faut bien distinguer entre le routage intra-réseaux et inter-réseaux. Le premier effectue les fonctions nécessaires pour transmettre les unités de données entre des

11 Initial Domain Part,
12 Domain Specific Part,
13 Authority Format Identifier,
14 Initial Domain Identifier,

entités attachées au même sous-réseau, i.e., entre des adresses SNPA[15] du même réseau. Alors que le routage inter-réseaux effectue les fonctions plus sophistiquées qui consistent à transmettre des messages entre des entités attachées à des sous-réseaux différents. Dans ce mode, les procédures de routage sont supposées capables de dériver l'adresse de SNPA (adresse physique) du relais suivant à partir de l'adresse réseau, unique et globale dans le contexte de l'OSI. Une fois le relais suivant connu, ce sont les processus de routage intra-réseaux qui seront appliqués [Burg 89].

Dans tous les cas, il est à noter que la tâche de routage est très simplifiée dans le contexte de l'OSI à cause de la structure hiérarchique d'adressage.

Par ce fait, le problème général du routage consiste donc à découvrir l'existence ainsi que l'accessibilité d'un système terminal (ES[16]), les systèmes intermédiaires (IS[17]) et déterminer ainsi lequel parmi les chemins candidats est le plus convenable pour faire communiquer deux ES spécifiques. Ce qui peut nous ramener (Fig. 1.5) à distinguer entre le routage ES-IS [ISO 9542], le routage intra-domaine IS-IS [ISO DP 10589] et le routage inter-domaine IS-to-IS [ISO 10747].

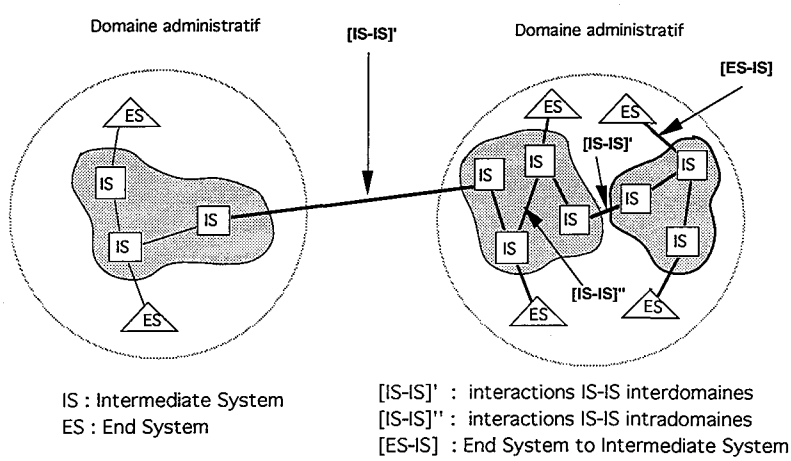


Fig. 1.5 — Hiérarchie de routage selon l'OSI

Dans l'OSI, les principaux objectifs du routage inter-réseaux peuvent être réduits à:

- accessibilité globale,
- indépendance de sous-réseaux,
- performance acceptable.

1.4.3. Le Nommage selon l'OSI

Il faut bien distinguer entre l'identité d'un processus d'application (son nom) et sa localisation dans le réseau (son adresse). Ceci a conduit ISO à définir toute une structure des noms dans une forme arborescente [X.500]. L'arbre des noms est appelé l'Arbre d'Information Répertoriée DIT[18], dans lequel, chaque noeud est étiqueté par un Nom Relatif Distingué RDN[19]. Un nom "Nom Distingué" est le résultat de concaténation d'une séquence ordonnée de RDNs que l'on rencontre en traversant l'arbre de la racine vers le RDN.

15 SubNetwork Point of Attachment,
16 End System,
17 Intermediate System,
18 Directory Information Tree,
19 Relative Distinguished Name,

Une opération de mise en correspondance entre l'espace des noms et celui des adresses est effectuée au niveau de la couche Application [Poo 90].

1.5. Les algorithmes d'extension des RLs

Il existe déjà deux algorithmes opérant au niveau de la couche liaison de données et qui ont été utilisés par l'IEEE pour permettre l'extension d'un réseau local sur plusieurs segments. L'algorithme de l'arbre recouvrant (spanning tree) et le routage par la source.

1.5.1. Algorithme de l'arbre recouvrant

Nous avons dit auparavant que les ponts transparents de [IEEE 802.1] utilisent l'algorithme de l'arbre recouvrant pour éliminer les éventuelles boucles et avoir un seul chemin entre n'importe quel couple de RLs.

Le déroulement de cet algorithme proposé par [Perlman 85] passe par les étapes suivantes :

1) *Sélection de la racine de l'arbre* : Il s'agit ici de choisir parmi les ponts, qui doivent être identifiés d'une façon unique, une racine vers laquelle les plus courts chemins seront calculés,
2) *sélection des ports vers la racine* : chaque pont désigne un de ses ports comme étant le port vers la racine ou le *port désigné*. Cette décision est fonction de la distance vers la racine,
3) *sélection des ponts désignés* : chaque RL doit désigner dans cette étape un pont qui lui est attaché et qui se trouve sur le plus court chemin vers la racine.

Les états des ports désignés ainsi que de tous les ports des ponts désignés sont supposés *actif* alors que tous les autres ports seront en état de *repos*. Pour éviter les boucles temporaires, le passage d'un état de *repos* vers un état *actif* doit passer par deux états intermédiaires : *écoute* et *apprentissage*. Dans l'état *écoute*, le port est autorisé à recevoir et à émettre seulement les messages concernant le protocole de l'arbre recouvrant BPDU[20]. Ni apprentissage ni expédition du trafic normal n'est autorisé dans cet état. Un temporisateur pour la réception du BPDU est associé à l'état *écoute* dont l'expiration change l'état vers *apprentissage*. Le comportement du port dans cet état ressemble à celui de l'état précédent sauf que le port aura la possibilité d'apprendre les adresses des stations. Cet état change vers *actif* sur une expiration du temporisateur pour une deuxième fois sans qu'un BPDU soit reçu. Enfin, un passage d'un état de *repos* à un état d'*écoute* se fait sur une nouvelle élection d'un pont désigné après que l'ancien pont désigné cesse d'envoyer des BPDUs pendant un certain délai.

Exemple (pris de [Backes 88]) :

20 Bridge Protocol Data Unit,

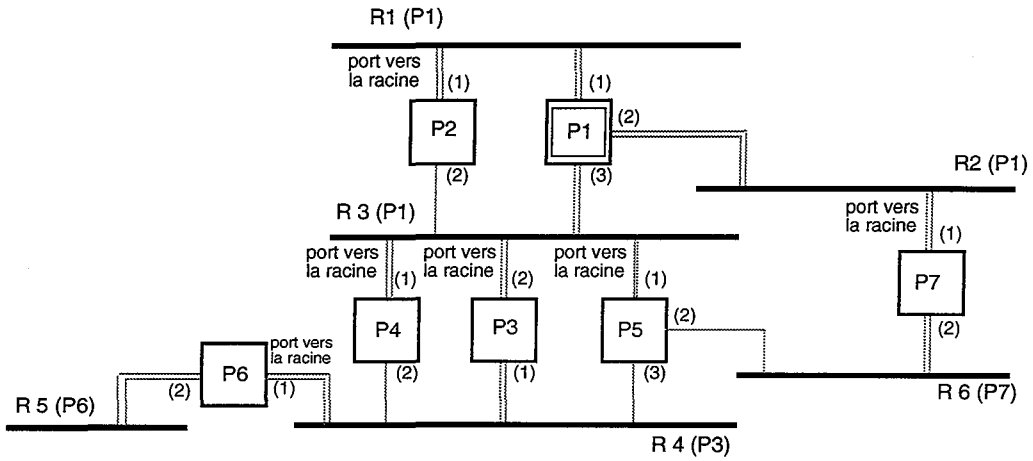


Fig 1.6 — Exemple d'une topologie d'interconnexion

La figure 1.6 présente une topologie de six réseaux locaux et de sept ponts. Le numéro de chaque pont est indiqué dans le rectangle. Les ports de chaque pont sont identifiés par les numéros entre parenthèses. Le pont P1 est choisi comme la racine. P1 est le pont désigné sur R1, R2 et R3 car il est la racine. Les ponts désignés pour les autres réseaux sont compris entre parenthèses à côté du nom de réseau (e.g. R5 a désigné (P6)). La figure montre aussi le port vers la racine pour chaque réseau.

P3 est le pont désigné pour R4 car il a le plus petit numéro parmi les deux autre ponts : P4 et P5 qui se situent à la même distance vers la racine.

Bien que les deux ports du P2 ont la même distance vers la racine, P2 a choisi le port numéro (1) comme le port vers la racine car il a un numéro inférieur à celui de l'autre port.

Les lignes doubles représentent des ports se trouvant dans l'état *actif* et les lignes simples représentent des ports se trouvant dans l'état *repos*.

Les informations nécessaires pour l'opération de l'algorithme de l'arbre recouvrant sont échangées entre les ponts en utilisant des messages spéciaux BPDUs. Ces messages contiennent des informations concernant la racine actuelle, la distance vers cette racine, l'age du message ainsi que l'identité du pont (y compris le port) qui a émis ce message. La racine émet périodiquement (1 à 4 secondes) un BPDU qui sera retransmis par les ponts désignés.

Nous supposons maintenant que les calculs précédents nous ont permis de construire l'arbre recouvrant la topologie d'interconnexion et que chaque pont connaît la branche qui conduit vers la racine et les branches qui conduisent vers les feuilles. Pour ce qui concerne l'apprentissage, chaque pont doit établir une table de couples (adresse, branche) qui lui permet d'aiguiller une trame reçue sur la branche k et ayant comme adresse de destination *dest* comme suit:

- s'il trouve une entrée j correspondant à l'adresse *dest* de façon que $j \neq k$, il envoie la trame sur cette entrée,
- si l'entrée est la même que k, le pont ignore cette trame parce que les autres branches ne conduisent pas à la destination,
- s'il ne trouve pas une entrée correspondant à *dest*, le pont diffuse la trame sur toutes les branches sauf la branche d'entrée.

Un pont construit la table de routage en testant l'adresse source *S* de chaque trame reçue sur une branche *k*. S'il ne trouve pas dans sa table de routage une entrée correspondant à *S* (ou il trouve une entrée mais avec une branche différente de k), le pont ajoute (modifie) le couple (S, k) dans la table.

1.5.2. Routage par la source

C'est un algorithme de routage plutôt que d'extension des réseaux [Soha 88] parce que chaque source doit découvrir tous les chemins possibles pour chaque destinataire pour pouvoir inclure ce chemin dans le message. Nous nous limitons ainsi au détail donné au paragraphe 1.2.2. pour ce type de routage ainsi que pour les critères et les propriétés de chacun des deux algorithmes précédents.

1.6. Les algorithmes de routage

Il existe deux types d'algorithmes de routage distribués qui sont utilisés généralement au niveau de la couche réseau [Perlman 94] :

1) le vecteur de distance,
2) l'état de liaisons.

1.6.1. Routage par vecteur distance

L'algorithme de base est connu sous le nom de ses inventeurs Bellman-Ford et pour lequel chaque routeur doit calculer les distances vers tous les destinataires possibles à partir de l'information contenue dans le vecteur distance de ses voisins :

- initialement, le vecteur distance pour chaque routeur comporte la valeur "0" pour lui-même et la valeur "∞" pour tous les autres routeurs,
- chaque routeur envoie le vecteur distance à tous ses voisins dès que l'information qui est en sa possession est modifiée,
- chaque routeur calcule son propre vecteur distance, en minimisant le coût de chaque destination, en additionnant le coût de cette destination rapporté par chaque voisin au coût de la liaison vers ce voisin.

Cet algorithme de routage est utilisé dans RIP (Routing Information Protocol) (RFC1058) pour le routage d'IP (Internet Protocol) de TCP/IP. Il a l'inconvénient d'être très lent à converger lors de la modification de la topologie.

1.6.2. Routage par l'état des liaisons

Le routage par l'état des liaisons, aussi connu comme le SPF[21], a été initialement proposé par l'ANSI[22] comme une norme internationale ISO pour le routage intra-domaine [ISO/IEC JTC1/SC 6]. Il a été implémenté également au dessus de IP dans OSPF[23] pour remplacer RIP dans Internet [Moy 89].

---

21 Shortest Path First,
22 American National Standardization Institute,
23 Open Shortest Path First,

Dans cet algorithme, qui est réparti aussi, l'information de routage, contrairement à l'algorithme précédent, doit se propager à tous les routeurs de la façon suivante:

- chaque routeur qui surveille les états de l'ensemble des liaisons et des sous-réseaux voisins, construit un paquet connu sous le nom de *paquet d'état des liaisons* ou LSP[24] qui contient une liste des noms et des coûts de chacun de ses voisins,
- ce LSP doit être transmis, lorsqu'il subit une modification, à tous les autres routeurs de l'interconnexion de réseaux, et chaque routeur enregistre le LSP généré le plus récemment par chaque autre routeur,
- chaque routeur, qui possède maintenant une connaissance complète de la topologie, calcule les routes vers chaque destination (souvent selon l'algorithme de Dijkstra [Dijkstra 59] de plus courts chemins). Cette route sera ainsi enregistrée sous forme de couple (sous-réseau suivant, routeur suivant).

Cet algorithme a l'avantage d'avoir une convergence rapide lors de la modification de la topologie. Cette modification va se propager vers tous les routeurs qui disposent chacun de la carte globale de la topologie.

Nous discutons maintenant les extensions apportées sur l'adressage et le routage pour autoriser la communication de groupes.

1.7. Adressage de groupe

Le but de l'adressage de groupe est de permettre à une entité d'envoyer un message ou d'entrer en communication avec un groupe d'entités. Cette extension d'adressage est bien connue au niveau de la couche physique où on peut configurer l'interface matérielle avec le support pour reconnaître toutes les adresses de groupes désirés. Elle est même reconnue dans certains réseaux physiques comme Ethernet. Ce dernier définit trois types d'adressage, à savoir individuel, diffusion et diffusion de groupe qui peuvent être utilisés pour envoyer une trame vers un, plusieurs ou toutes les stations terminales à l'intérieur d'un seul réseau physique.

La diffusion de groupe au niveau de la couche réseau ou IP dans TCP/IP doit permettre à une entité d'envoyer un message ou datagramme vers un ensemble d'entités. Les entités forment un même groupe de diffusion et peuvent être réparties sur plusieurs réseaux physiques distincts. Les règles suivantes sont souvent respectées dans ce type d'adressage :

- l'appartenance à un groupe de diffusion peut être statique ou dynamique,
- une entité peut envoyer un message à un groupe de diffusion sans y appartenir; un producteur de l'information ne doit pas faire partie de consommateurs par exemple,
- une adresse de groupe ne peut être utilisée que comme adresse de destination,
- il ne faut pas avoir de limite sur le nombre des membres de groupe (le nombre de consommateurs est illimité)

Les drafts : [ISO/IEC 8348], [ISO/IEC 9542] et [ISO/IEC 8473-1] portent des modifications sur le texte actuel de l'ISO pour permettre l'adressage et le routage multipoint. Par contre, les normes de fait IETF[25] ont bien avancé dans ce domaine et des normes concernant l'adressage de groupe ou le routage multipoint ont été déjà définies ou en cours d'expérimentation. Cela a permis à TCP/IP [Comer 92] et [Huitema 95] d'intégrer les

24 Link State Packet,
25 Internet Engineering Task Force,

adresses de diffusion de groupe (classe D) au niveau IP et d'utiliser ainsi le protocole IGMP[26] [RFC 1112] qui s'exécute dans les routeurs pour déterminer l'appartenance dynamique à un groupe de diffusion. Concernant la propagation des informations d'appartenance de groupes sur les différents réseaux, TCP/IP a recours à des routeurs spéciaux utilisant une version multipoint de l'algorithme de vecteur de distance, appelée DVMR[27] [RFC 1075] qui est en cours d'expérimentation. On retrouve également dans les travaux de [Pansiot 95] une proposition d'une sous-couche LAR (Logical Addressing and Routing) qui peut être integrée au dessus de la couche IP et qui s'occupe des adresses de groupes ainsi que de routage multipoint.

Il reste à signaler qu'une adresse de groupe est une adresse logique indépendante de la localisation physique. Elle ne donne aucune information concernant l'adresse individuelle de chaque membre de groupe .

1.8. Routage multipoint

Nous nous intéressons dans la suite aux algorithmes de routage multipoint dans une interconnexion en mode sans connexion (datagramme). En revanche, on retrouvera dans [Tanaka 90] un algorithme de routage multipoint pour la communication en mode avec connexion (circuit virtuel) dans les réseaux longue distance qui peut être utilisé pour assurer des conférences téléphoniques.

Le problème de routage multipoint dans une topologie arbitraire (maillée) d'interconnexion [Deering 90a] peut être présenté comme suit:

- Les routeurs/ponts doivent s'arranger entre eux pour délivrer une copie de chaque message multipoint à tout sous-réseau destinataire qui comporte un membre de groupe au moins, et
- le sous-réseau destinataire doit compléter le transfert du message à chaque membre de groupe à l'intérieur de ce sous-réseau.

Le fait que le routage multipoint est une généralisation du routage point-à-point traditionnel (dans ce contexte, un groupe réduit à un seul membre est équivalent au routage point-à-point) a conduit à étendre les algorithmes existants pour répondre à ce nouveau besoin. La suite est consacrée aux trois extensions qui portent soit sur l'algorithme de l'arbre recouvrant, soit sur l'algorithme du vecteur de distance, soit sur l'algorithme de l'état de liaisons.

1.8.1. Routage multipoint par un arbre recouvrant

Nous avons mentionné que les ponts transparents (arbre recouvrant) ne sont pas efficaces pour gérer les adresses de groupes. Cette inefficacité vient du fait qu'un pont n'est pas capable d'apprendre une adresse de groupe parce qu'elle ne peut pas être utilisée comme une adresse source, indépendante de la localisation physique en plus. Par conséquent, les trames comportant des adresses de groupe seront diffusées sur toutes les branches de l'arbre recouvrant pour atteindre tous les réseaux.

Les deux extensions suivantes qui portent sur cet algorithme ont été proposées par [Deering 90a]:

26 Internet Group Management Protocol,
27 Distance Vector Multicast Routing,

1) diffusion tronquée : il consiste comme son nom l'indique à tronquer la diffusion des trames sur les feuilles de l'arbre s'il n'existe aucun membre de groupe destinataire dans cette feuille,
2) routage multipoint sur l'arbre recouvrant : dans cette extension, chaque pont doit connaître les branches de l'arbre conduisant aux membres relatifs à chaque groupe de diffusion. Pour ce faire, le pont doit apprendre la localisation de ces membres d'une façon explicite. Autrement dit, chaque station doit indiquer au pont les différents groupes d'appartenance selon un protocole qui assure la propagation de cette appartenance vers les autres ponts.

1.8.2. Routage multipoint par un vecteur distance

Nous supposons ici que les calculs de l'algorithme de vecteur distance (voir 1.6.1) ont permis à chaque routeur d'associer à chaque destinataire le routeur suivant (destinataire, routeur suivant) pour l'atteindre. Ceci veut dire qu'un arbre de plus court chemin reliant chaque source avec tous les autres réseaux est établi.

Une solution immédiate pour envoyer une trame comportant une adresse de groupe est la diffusion [Dalal 78] ou la diffusion tronquée [Deering 91] vers tout destinataire.

Mais vu la surcharge et le nombre de copies engendrées par la diffusion, [Deering 90b] a proposé dans l'algorithme DVMR d'élaguer les branches finales de l'arbre qui ne comportent pas de membres de groupe et ainsi établir pour chaque couple (source, groupe) un arbre de routage multipoint.

1.8.3. Routage multipoint par l'état de liaisons

Nous avons dit auparavant (voir 1.6.2) que chaque routeur dispose d'une connaissance de la topologie complète de l'interconnexion. Ceci va permettre à chaque routeur de calculer l'arbre recouvrant la topologie totale pour chaque source et d'en déduire les liaisons parents sur lesquelles il doit recevoir les paquets en provenance de la source et les liaisons fils sur lesquelles il va envoyer ces paquets. Un routeur peut également décider lequel parmi ses fils (sous-réseaux) est une feuille de l'arbre de diffusion pour tronquer la diffusion dans le cas où le routage multipoint est effectué par une diffusion. Il faut noter ici que tout sous-réseau, même s'il ne contient pas de membres de groupe et qu'il n'est pas sur un chemin conduisant vers un membre de groupe, va recevoir le paquet.

Si par contre, nous nous intéressons à optimiser l'arbre d'acheminement en élaguant les sous-réseaux précédents, la version multipoint du routage par l'état des liaisons sera utilisée. Dans ce cas là, les informations relatives à l'appartenance aux différents groupes feront partie du paquet d'état des liaisons LSP. La modification d'un groupe va engendrer ainsi un changement d'état de liaison et donc un paquet de mise à jour va être diffusé vers tout autre routeur. Par conséquent, une connaissance complète et consistante concernant la distribution des membres de différents groupes sera partagée par tous les réseaux (et routeurs). Lors d'une modification d'un groupe, un routeur peut recalculer l'arbre d'acheminement entre les différentes sources possibles et les membres du groupe.

Il faut noter ici que les arbres précédents sont calculés à la base des plus courts chemins vers la racine (source) qui est connu comme l'algorithme de SPT[28].

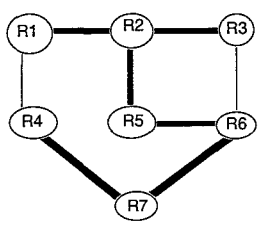
28 Shortest Path Tree,

Cette extension est utilisée dans MOSPF [RFC 1584] et [RFC 1585] pour le routage multipoint dans un système autonome (intra-domaine) utilisant le protocole IP.

1.9. Limites des algorithmes de routage multipoint

Nous avons traité auparavant les différents équipements d'interconnexion de réseaux ainsi que les différents algorithmes de routage ou d'extension. Nous nous intéressons, dans la suite, à deux facteurs qui sont équivalents dans le cas de routage en point à point mais qui deviennent deux problèmes distincts dans le cas de multipoint. Le premier facteur est les distances individuelles, en nombre de routeurs, entre la source et chaque destinataire. Minimiser cette distance signifie, dans la plupart des réseaux, minimiser les délais de propagation. Nous citons parmi ces réseaux : les réseaux véhiculant des données multimédia et les bus de terrain, ou autrement dit les réseaux temps réel et les réseaux temps critique. Le deuxième facteur concerne le nombre total de routeurs utilisé pour acheminer un paquet vers ses destinataires. Il s'agit ici d'optimiser, en même temps, les ressources et le nombre de copies d'un paquet (en fait, le nombre de copies dépend du nombre de branches dans l'arbre d'acheminement). Et par conséquent, il faut éviter, tant que possible, les problèmes de congestion.

1. Cas de l'extension multipoint de l'arbre recouvrant : le fait qu'un seul arbre est construit pour toute la topologie rend ce type de pont inefficace pour minimiser la distance vers les destinataires. La figure suivante montre un exemple de cette inefficacité :



Les noeuds sont les sous-réseaux, les arcs sont les ponts et la racine est le réseau R5

Fig. 1.7 — Exemple du routage multipoint sur un arbre recouvrant

Dans l'exemple précédent, tous les trafics de l'interconnexion vont emprunter l'arbre dénoté en gras. Si maintenant le réseau R1 doit envoyer un paquet vers le réseau R4, bien qu'il existe un pont direct reliant les deux réseaux, le chemin sera {R1, R2, R5, R6, R7, R4} ce qui signifie l'accroissement de la longueur du chemin dans ce type d'algorithmes. Supposant maintenant que les deux réseaux R1 et R3 font partie d'un groupe de diffusion, si le réseau R4 va envoyer un paquet pour ce groupe, l'arbre emprunté sera l'arbre de diffusion complet {R4, R7, R6, R5, R2, (R1, R3)} même si on utilise un algorithme de routage multipoint.

2. Les algorithmes fondés sur le vecteur distance et sur l'état des liaisons : Les deux autres algorithmes minimisent bien la distance vers les destinataires parce que l'arbre de plus court chemin est construit pour chaque source ce qui évite le problème précédent mais par contre ils n'optimisent pas le coût total de l'arbre en nombre de routeurs.

Une étude comparative concernant les différents algorithmes de routage multipoint ainsi que les propriétés et la complexité de chaque algorithme est donnée dans le chapitre 6.

Chapitre 2

Présentation du réseau de terrain FIP

FIP[1] est conçu principalement pour faire communiquer ou assurer l'échange de données entre capteurs, actionneurs, et les différents équipements de commande et de contrôle tel que les automates programmables et les régulateurs [Thomesse 89a], [Thomesse 93]. Les réseaux de terrain sont généralement caractérisés par un coût faible de câblage et par la connaissance préalable, lors de l'installation de l'application, du trafic échangé (une mesure de pression est toujours émise par le même capteur et reçue par le(s) même(s) automate(s)). Cette connaissance concerne non seulement la source et le(s) destinataire(s) du trafic mais aussi des paramètres temporels qui peuvent concerner la fréquence d'échange et la cohérence des données issues de la répartition de l'application sur plusieurs sites.

Nous nous concentrons pour le reste de ce chapitre sur l'architecture générale du réseau FIP et le mode de coopération entre les processus d'application d'une part et sur les idées innovantes retenues par ce réseau comme la mise à jour en temps réel d'une base de données répartie qui, en assurant la cohérence temporelle et spatiale de ses données, fiabilise la communication en mode multipoint. Il s'y ajoute le mode d'adressage logique (par objet) du trafic identifié qui permet de cacher les détails concernant l'origine ou le(s) destinataire(s) de l'information. Toutes ces caractéristiques ont fait que FIP est devenu la norme française pour les bus de terrain, la plupart de ces mécanismes ont été retenus pour la normalisation Européenne [EMUG 89] et Internationale [Grant 92].

Puisque notre étude est principalement concernée par la couche liaison de données, nous présentons en détail les services rendus par cette couche ainsi que le contrôle d'accès au médium.

1 Factory Instrumentation Protocol ou Flux Information Processus,

2.1. Architecture générale

L'OSI [Zimmerman 80] a défini un modèle de référence à 7 couches. FIP [UTE 90a] en a retenu la couche physique, la couche liaison de données et la couche application. Ce choix se justifie par le fait que :

- la réduction de nombre de couches diminue le temps cumulé de traversée de ces couches mais elle ne devient dans aucun cas une condition suffisante pour garantir a priori les contraintes temporelles [Elloy 90], [LeLann 91].
- un réseau de terrain est, à l'origine, monosegment ou étendu par des ponts. Cette hypothèse a entraîné la suppression de la couche réseau.
- l'adaptation des messages à la taille des trames liaison et la connaissance préalable du flux de données et de la qualité de services rend la suppression des couches transport et session possible aussi.

Contrairement au trafic d'objets identifiés, la configuration, le téléchargement et le transfert de comptes rendus de diagnostics pour les équipements de terrain font appel à des messages classiques qui utilisent des adresses physiques et qui peuvent être bipoint ou multipoint. Ce type de messagerie, non assujetti à des contraintes temporelles, a entraîné l'adoption de deux profils de communication : le premier assure le trafic temps réel des objets identifiés et le deuxième assure le trafic de messagerie comme illustré dans la figure suivante :

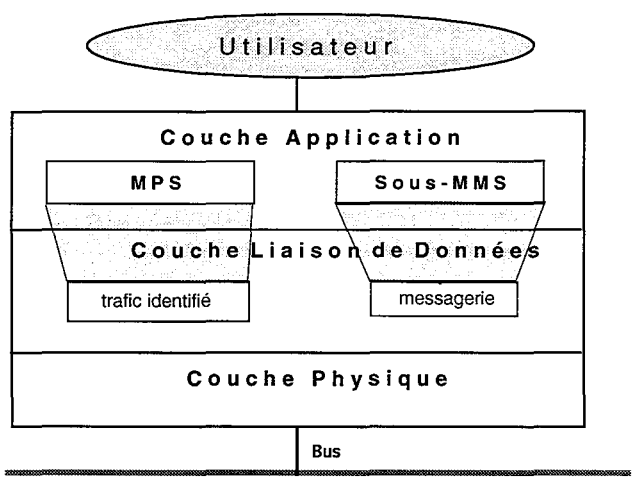


Fig. 2.1 — FIP : architecture et services

La signification et le contexte des termes utilisés dans la figure 2.1 sont abordés dans les sections suivantes.

2.2. Aperçu de la couche application

En ce qui concerne la couche application, FIP offre deux ensembles de services : MPS[2] [UTE 90b] et SUB-MMS[3][UTE 92].

2 Message Periodic/aperiodic Specifications,
3 Manufacturing Message Specification,

Par l'intermédiaire du premier ensemble de services, FIP offre à l'utilisateur la possibilité de définir les variables produites et/ou consommées ainsi que les types de valeurs portées par ces variables et les différents statuts de validité temporelle tels que la promptitude associée à la consommation et le rafraîchissement associé à la production. Il offre aussi l'accès aux listes des variables avec éventuellement des statuts de cohérence temporelle et spatiale.

Les principaux services mis à la disposition de l'utilisateur sont les suivants :

- lecture et écriture locale de variables qui fournissent à l'utilisateur la possibilité de lecture de la valeur courante d'une variable consommée ou d'écriture de la valeur d'une variable produite. Ces services sont effectués sur des variables disponibles localement et ne provoquent pas de transmissions sur le bus,
- lecture et écriture distante de variables qui fournissent les mêmes services qu'auparavant mais avec une transmission de la valeur produite sur le bus vers les consommateurs,
- demande de mise à jour de variables : il s'agit ici de demander explicitement la mise à jour et la transmission sur le bus du producteur vers les consommateurs d'une valeur de variable. L'initiateur de cette demande peut être soit un producteur, soit un consommateur, soit une tierce entité,
- lecture de listes de variables : ce service permet à l'utilisateur d'effectuer une lecture locale de l'ensemble des valeurs des variables composant une liste déclarée localement en consommation. Il fournit optionnellement les divers statuts de cohérence temporelle et spatiale associés à cette liste.

Les services de messagerie industrielle SUB-MMS sont principalement utilisés pour le besoin de la gestion de la configuration, des modes de marche et des paramètres de l'application répartie et de la supervision.

2.3. Modèle PDC[4]

Les besoins d'un modèle aussi bien de services que de protocoles qui fournit des moyens pour répondre aux contraintes liées à la validité temporelle et à la cohérence des données échangées en multipoint, ont donné naissance au modèle PDC [Thomesse 89b] qui est illustré dans la figure suivante :

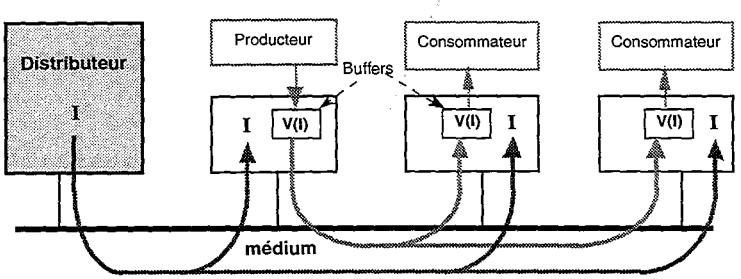


Fig. 2.2 — Modèle PDC

Trois types de processus, à savoir, P, D et C, cohabitent pour assurer l'échange d'objets selon PDC : un producteur P produit localement la valeur d'un objet (V(I) pour la valeur et I pour l'objet); le distributeur D déclenche le transfert et la recopie chez les consommateurs

4 Producteur-Distributeur-Consommateurs,

C1, C2,...Cn de la valeur originale prise chez le producteur; les consommateurs utilisent la copie locale de l'objet.

Les processus précédents peuvent être périodiques ou apériodiques et les relations d'interdépendance temporelle, par exemple les fréquences de production, de consommation ou de transmission pour l'échange périodique, dépendent de l'application.

Ce modèle, outre les points mentionnés au-dessus, est bien adapté aux bus de terrain parce que nous disposons de connaissances sur la nature, la source ainsi que les destinataires des grandeurs échangées (mesures numériques ou booléennes en provenance des capteurs, variables d'états entre les organes de commande). Nous ajoutons la possibilité de synchronisation de plusieurs processus (sur réception d'une valeur dont un processus est consommateur) à un délai de propagation près ce qui a permis ainsi de qualifier la cohérence de données par rapport au temps et par rapport à l'espace.

2.4. Couche liaison de données

La couche liaison de données [UTE 90c] offre deux types de services :

- échange de variables
- transfert de messages.

En plus, chaque type de service existe sous deux formes : cyclique (c'est à dire, avec des périodes fixées à la configuration de l'application) et apériodique par demande explicite d'échange.

2.4.1. Gestion d'accès au médium

L'adaptation du modèle PDC au niveau liaison de données fait appel à une entité, appelée Arbitre de Bus (AB). Cette entité, qui remplace le distributeur, contrôle le droit d'accès au médium de chaque producteur en émettant une trame contenant un identifieur. A un moment donné, il ne doit y avoir qu'un seul Arbitre de bus actif par segment réseau FIP.

Trois classes d'allocation de médium sont définies :

- échange cyclique de variables, de requêtes ou de messages,
- demande explicite d'échange de variables,
- demande explicite de transfert de messages.

Pour les échanges cycliques de variables, une transaction élémentaire se compose d'une phase de diffusion d'une trame identifieur de variables par l'Arbitre de bus puis d'une phase de diffusion d'une trame réponse variable par l'unique producteur de l'information requise. Au cours de cette même phase, l'information est copiée par les consommateurs. La figure 2.3 décrit les différentes phases d'une transaction d'échange de variable. A l'issue d'une transaction, l'Arbitre de bus déclenche la transaction suivante selon les directives définies à la configuration du système.

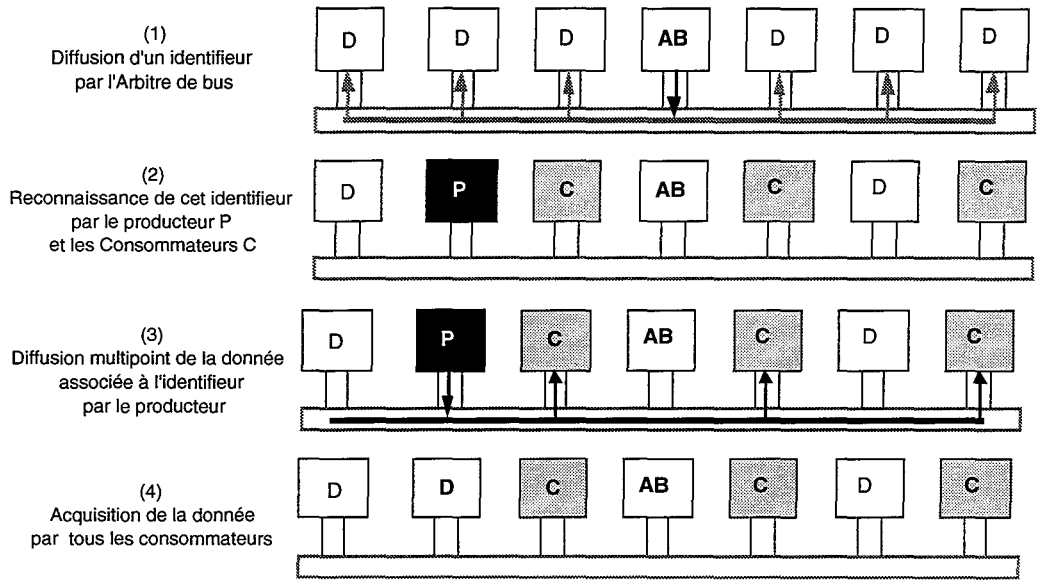


Fig. 2.3 — Le contrôle d'accès au médium

L'Arbitre de bus doit octroyer la parole à chaque producteur d'information en passant par les quatre phases suivantes :

- scrutation périodique de variables, de messages et de listes de variables,
- scrutation apériodique de variables,
- scrutation apériodique de messages,
- synchronisation

Les quatre fenêtres constituent un cycle élémentaire (micro-cycle) de scrutation ayant, grâce à la phase de synchronisation, une durée constante qui peut être d'une grande importance pour des applications temps réel. Les différentes phases de scrutation (voir Fig. 2.4) sont :

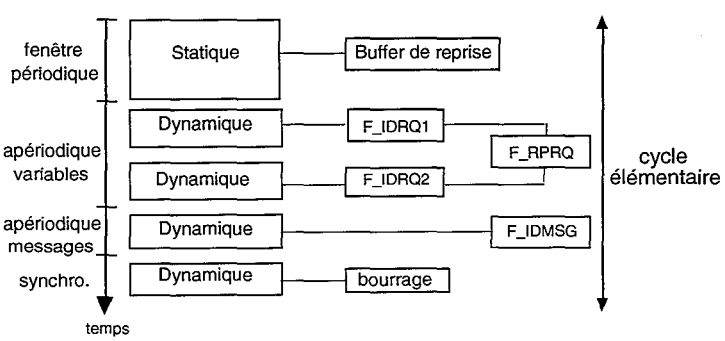


Fig. 2.4 — Les différentes phases de scrutation de l'AB

1- une fenêtre périodique de scrutation : cette fenêtre comporte une suite d'identifieurs fixée à la configuration du système qui garantit la périodicité de toute variable. Un identifieur peut désigner un buffer contenant la valeur d'une variable périodique (lorsque la trame diffusée par l'AB est ID_DAT(id)), une file d'attente de message périodique (lorsque la trame est ID_MSG(id)), dans ce cas là, une file d'attente doit être allouée par identifieur, ou enfin une liste de variables (le buffer de reprise est ainsi utilisé pour stocker les identifieurs contenus dans cette liste). Il faut bien noter

que l'adresse destinataire et le corps d'un message d'une part et les variables contenues dans la liste d'autre part ne sont pas connus à l'avance.

2- la fenêtre apériodique de variables permet le transfert des listes de variables suivies par les valeurs des variables dont des demandes explicites ont été effectuées dans une fenêtre périodique (pas forcément la dernière fenêtre périodique). Ce sont les files d'attente F_IDRQ1 et F_IDRQ2 qui permettent à l'AB de stocker les identifieurs portant les demandes explicites urgentes ou normales. La file d'attente F_RPRQ, quant à elle, permet à l'AB de stocker la liste des identifieurs demandés.

3- la fenêtre apériodique de messages permet le transfert de messages dont des demandes explicites ont été déjà effectuées. L'AB dispose d'une file d'attente F_IDMSG lui permettant de stocker les différents identifieurs qui ont porté les demandes.

4- la fenêtre de synchronisation permet à l'AB d'ajuster une durée constante du cycle élémentaire. L'AB diffuse, pendant cette fenêtre, une trame particulière de bourrage qui est reconnu par toutes les autres stations.

A titre d'exemple, la figure suivante montre une configuration valide d'une table de scrutation d'un AB qui respecte les périodicités des variables (a,b,c,d) de (2,4,6,12) respectivement.

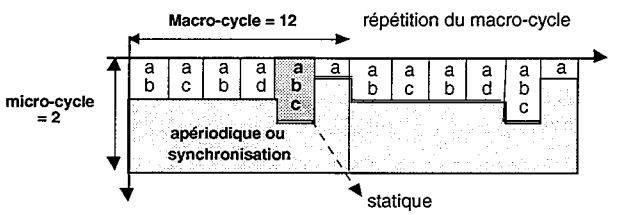


Fig. 2.5 — Exemple d'une table de scrutation

2.4.2. Services offerts par la couche Liaison de données

Les services rendus par la couche liaison de données concernent 3 types de trafic :

1. TRAFIC DE VARIABLES IDENTIFIÉES

Trois types de primitives relatives au trafic de variables identifiées globalement sont offerts par la couche Liaison de données :

- écriture de la valeur dans un buffer au niveau Liaison (primitive L_PUT),
- transmission de la valeur (primitives L_SENT et L_RECEIVED),
- lecture de valeur par la couche application (primitive L_GET).

La figure 2.6 illustre l'enchaînement de ces primitives.

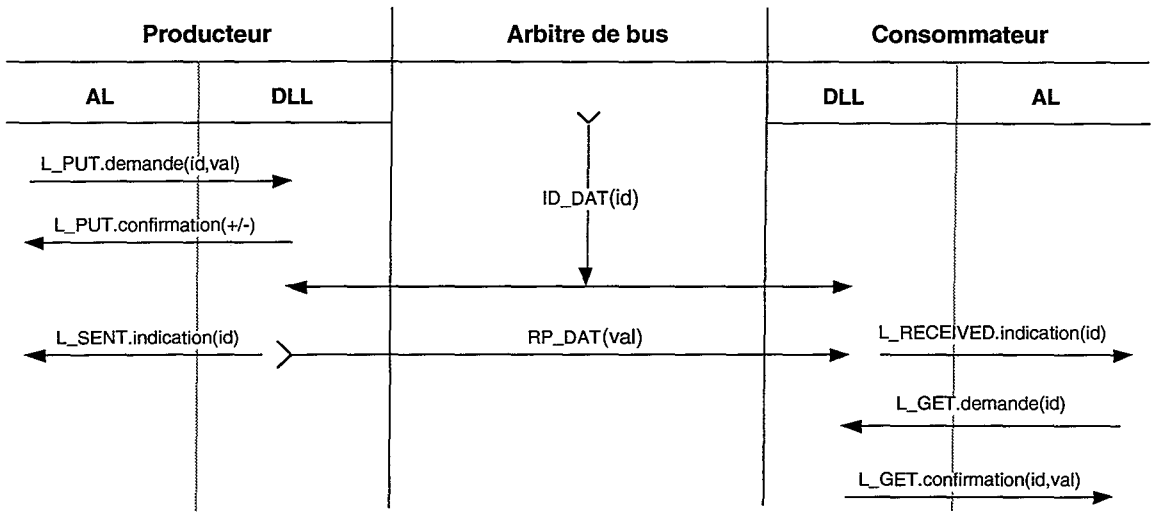


Fig. 2.6 — Trafic identifié

Puisque les trois opérations : écriture, transmission et lecture sont indépendantes les unes des autres, des situations telles que l'écriture dans un buffer qui n'a pas encore été transmis, une transmission successive de la même valeur, lecture multiple de la même valeur ou un écrasement d'une valeur qui n'a pas été lue par une autre valeur peuvent survenir.

Quant au service de transfert, il est déclenché par l'AB qui, selon sa table de scrutation, diffuse une trame ID_DAT(id) vers toutes les stations. A la réception de ID_DAT, les consommateurs se préparent pour recevoir la prochaine trame qui contiendra la valeur associée à l'identifieur *id* alors que le producteur de *id* diffuse cette valeur sur le bus dans une trame RP_DAT(Val) et envoie, au même instant, l'indication de l'émission vers la couche application; chaque consommateur de l'identifieur *id,* en recevant RP_DAT, envoie une indication de réception vers sa couche application. Nous remarquons que la production (L-PUT, la transmission et la consommation de l'information sont indépendantes entre elles).

Le déroulement de la transaction précédente peut s'effectuer d'une façon périodique ou apériodique selon la phase de scrutation.

2. TRAFIC DE MESSAGERIE

Pour assurer le trafic de messagerie, la couche Liaison de FIP offre les trois types de services suivants :

- dépôt du message dans la couche Liaison (L_MESSAGE.demande),
- transmission du message selon le protocole d'accès de FIP,
- indication, pour le destinataire, de la réception du message et confirmation, pour la source, de l'envoi du message.

Le transfert d'un message sans acquittement sur le bus s'effectue par l'une des deux façons suivantes :

1) lorsque l'identifieur `id` contenu dans *L_MESSAGE.demande* est configuré périodique de messagerie, le déroulement de cette transaction se fait comme suit (voir Fig. 2.7) : l'AB, à un instant donné de la table de scrutation, diffuse une trame ID_MSG(id); les stations qui ne sont pas productrices de cet identifieur se préparent en étant des

éventuels destinataires; le producteur (la source de message) de cet identifieur diffuse la trame RP_MSG_NoAck qui contient son adresse source (adresse physique), l'adresse de destinataire (ou un groupe de destinataires) et le corps du message. Les destinataires du message se reconnaissent à partir de l'adresse destinataire contenue dans la trame RP_MSG_NoAck et non pas de l'identifieur contenu dans la trame ID_MSG. La trame RP_FIN va libérer toutes les stations y compris l'AB, de l'état d'attente.

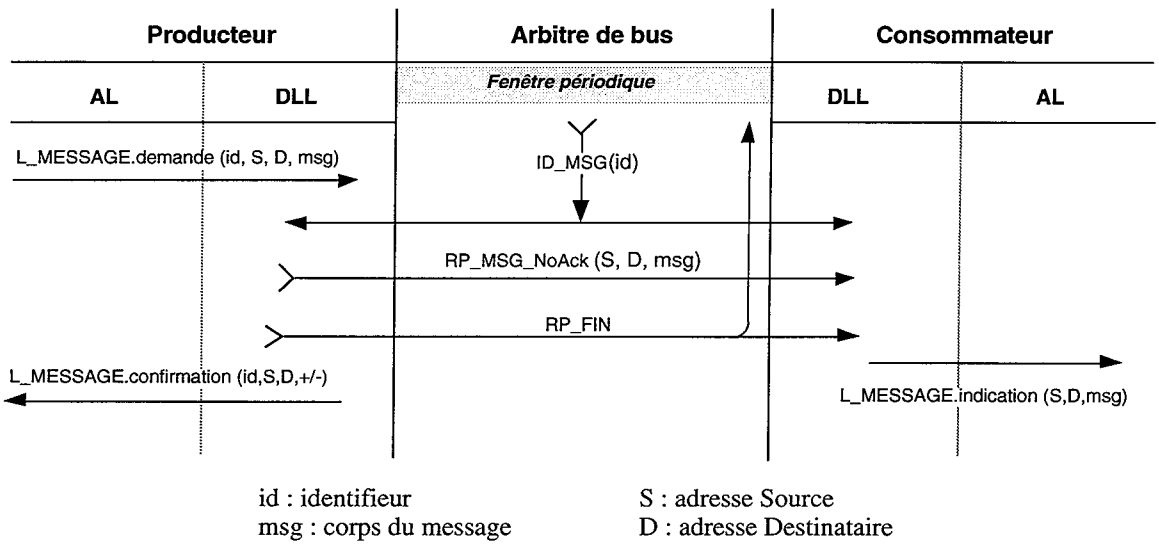


Fig. 2.7 — Trafic périodique de messages sans acquittement

2) Pour envoyer un message sans acquittement dans la fenêtre apériodique (voir Fig. 2.8), la couche application doit désigner un identifieur *id* relatif à une variable périodique produite par cette station. A la réception de la trame ID_DAT(id) du même identifieur, cette station, au lieu d'envoyer la trame RP_DAT(val), va envoyer une trame RP_DAT_MSG(val) pour exprimer son désir d'envoyer un message apériodique sur le même identifieur *id*. Le déroulement sera ainsi identique au précédent.

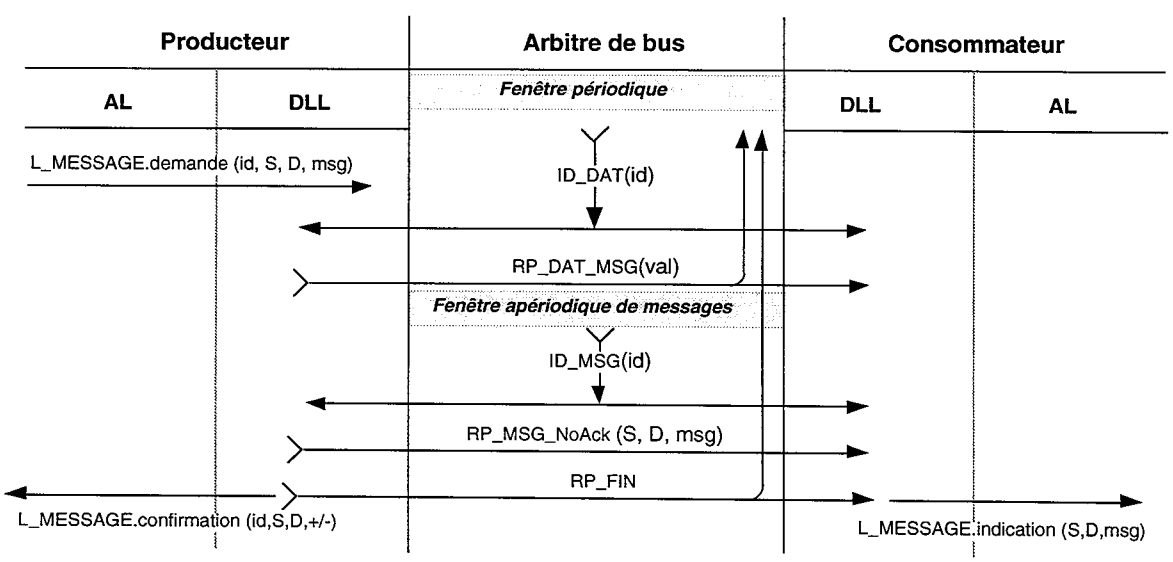


Fig. 2.8 — Trafic apériodique de messages sans acquittement

Il faut noter qu'un trafic de messagerie point-à-point avec acquittement est également assuré par la couche liaison de données. La seule différence concernant le déroulement de la transaction avec la messagerie non acquittée est que la source du message envoie RP_MSG_Ack ; un protocole d'acquittement entre la source, le destinataire et l'AB prend place. Ce protocole doit aboutir soit à l'envoi de RP_FIN par la source [UTE 90c], soit à l'échec de la reprise détecté chez l'AB. Nous ne nous intéressons pas à ce type de trafic parce qu'il n'est pas abordé dans l'étude du pont.

3. TRAFIC DE LISTES DE VARIABLES

Les valeurs associées à une liste de variables peuvent être échangées sur le bus par l'intermédiaire de ce service. La couche Liaison fournit deux types de services : spécifié (les primitives utilisées : L_SPEC_UPDATE.demande et L_SPEC_UPDATE.confirmation) et libre (les primitives utilisées : L_FREE_UPDATE.demande et L_FREE_UPDATE.confirmation). Dans le premier type, la couche application désigne l'identifieur de variable périodique qui doit porter la demande alors que dans le deuxième type, la liste sera associée dynamiquement par la couche Liaison au premier identifieur reçu dont la station est productrice.

La figure 2.9 montre l'enchaînement des primitives qui sont utilisées pour permettre le transfert de la liste d'identifieurs [a,b] dans la scrutation apériodique de requêtes :

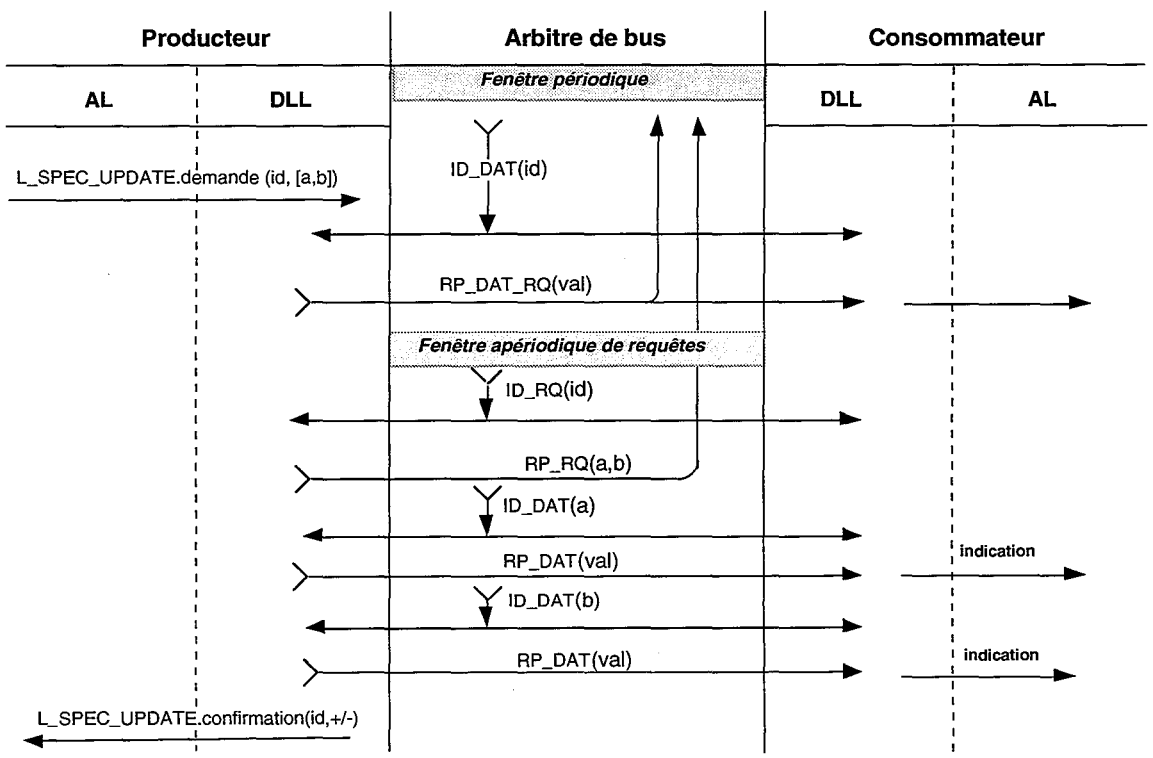


Fig. 2.9 — Trafic apériodique de requêtes

Comme dans la messagerie, le transfert s'effectue d'une des deux façons suivantes :

1) pour transférer une liste de variables dans la fenêtre périodique, un identifieur *id* périodique de requêtes doit être configuré dans la table de scrutation de l'AB. Ce dernier va diffuser une trame de requête de listes de variables ID_RQ(id) à l'issue de chaque période; le producteur de cet identifieur envoie vers l'AB la trame réponse

RP_RQ(liste_id) contenant la liste d'identifieurs souhaités suivie par un déroulement normal de variables identifiées.

2) pour transférer une liste de variables dans la fenêtre apériodique (voir figure 2.9), la station initiatrice de la demande, à la réception de la trame ID_DAT(id) , *id* doit être soit désigné explicitement par la couche application, soit le premier identifieur produit. Cette station envoie à la place de RP_DAT(val), la trame RP_DAT_RQ(val) pour exprimer son désir de voir circuler les valeurs associées à une liste de variables sur le bus. Le déroulement sera ainsi identique au précédent.

Il faut noter ici que les variables contenues dans la liste peuvent ne pas être produites ou consommées par l'initiateur de la demande. De plus, cette transaction met en communication les N producteurs successifs de la liste d'identifieurs avec les M consommateurs. Ce type de trafic est donc multipoint-multipoint entre N producteurs et M consommateurs.

FIP utilise aussi deux qualités de services pour présenter le dégré d'importance des demandes, à savoir urgent et normal.

2.4.3. Adressage

Le modèle d'adressage d'un réseau FIP [UTE 91b] fait apparaître deux espaces d'adressage complètement différents : le premier espace utilise des identifieurs pour le trafic identifié, pour attribuer le droit de parole et pour désigner enfin les producteurs successifs ainsi que les consommateurs, le deuxième espace est l'adressage classique permettant le transfert de messages en point-à-point et en multipoint.

1. ADRESSAGE PAR OBJETS

Ce mode d'adressage est effectué par les identifieurs. Un identifieur est une adresse logique, codé sur 16 bits, qui a une portée locale à un seul segment et qui permet à l'AB d'attribuer le droit de parole à son unique producteur et le droit d'écoute à :

- ses consommateurs à l'issue de la trame ID_DAT. L'identifieur est utilisé ici comme une adresse spéciale d'un groupe composé d'un producteur et d'ensemble de consommateurs,
- toute autre station à l'issue de la trame ID_MSG. Il faut noter ici que l'identifieur est utilisé comme une adresse de diffusion générale et que le producteur qui devient une source du message redésignera son (ses) destinataire(s) par une adresse classique de messagerie,
- l'AB lui-même à l'issue de la trame ID_RQ. L'identifieur est utilisé ici comme une adresse individuelle.

Compte tenu des points précédents ainsi que de la possibilité d'avoir des trames qui ne contiennent pas d'identifieur ni d'autre forme d'adresse physique de destination comme la trame réponse RP_DAT par exemple, l'identifieur tout seul demeure insuffisant pour désigner sans ambiguïté le destinataire. Tout cela nous a conduit à identifier la notion de SAP et de CEPi qu'on va illustrer en détail dans le paragraphe suivant pour éclaircir ce mode d'adressage d'une part et d'identifier les connexions qui peuvent remplacer l'adressage explicite d'autre part.

2. ADRESSAGE POUR LA MESSAGERIE

Ce mode d'adressage est utilisé pour le transfert de messages en point-à-point et en multipoint. Pour les messages en point-à-point avec ou sans acquittement, une adresse source et une adresse destinataire sont désignées par une chaîne de 24 bits dont 8 bits désignent le numéro de segment et 16 bits désignent une station à l'intérieur d'un segment. Pour les messages multipoint qui sont non acquittés, un mode d'adressage de groupe est fourni pour permettre à une source d'envoyer un message à un groupe de destinataires (messages multi-destinataires). Une chaîne de 24 bits est utilisée pour désigner un groupe logique de destinataires qui est supposé statique.

Il convient de signaler enfin le manque de précisions dans les normes de la couche Liaison [UTE 91b] et [UTE 90c] concernant le droit d'utilisation d'une adresse de groupe : le droit d'émettre à une adresse de groupe est-t-il restreint aux membres de ce groupe ou bien n'importe quelle station peut-elle émettre? la source est-elle fixée à l'avance?

PARTIE II

CHAPITRE 3 : Structuration de la couche Liaison de FIP

CHAPITRE 4 : Interconnexion de réseaux FIP

CHAPITRE 5 : Spécification formelle d'un réseau FIP connecté à un pont

CHAPITRE 6 : Algorithme de routage multipoint de plus courts délais et de coût minimal

Chapitre 3

Structuration de la couche Liaison de FIP

Le fait qu'un pont agit habituellement au niveau MAC et que le trafic objet de FIP doit être globalement identifié d'une façon unique et non ambiguë, nous a conduit à effectuer ce travail.

Nous présentons dans ce chapitre la solution retenue pour identifier les fonctionnalités et les services rendus par chacune des sous-couches MAC et LLC. L'interface entre ces deux sous-couches est symbolisée par des points d'accès aux services SAPs. Nous présentons aussi les modifications apportées sur les machines à états finis aussi bien pour l'Arbitre de Bus que pour une station terminale pour tenir compte de l'identification précédente et de la notion de SAP.

3.1. Identification de SAP/CEPi

3.1.1. Définitions :

Entité (N) : est un élément actif qui implémente des fonctions de couche (N) ainsi que des protocoles de communication avec une entité homologue. Plus précisément, une entité peut être un processus dans un système multiprocesseur ou tout simplement une procédure [Poo 90].

Le point d'accès à des services (N) (SAP(N)) : symbolise l'interface permettant à une entité (N+1) d'accéder aux services offerts par une entité (N). Le lien entre deux points d'accès peut être soit statique, soit dynamique [ISO 7498-3].

connexion(N) : association établie par la couche (N) entre deux ou plusieurs entités pour le transfert de données.

extrémité de connexion (N) CEP : terminaison d'une connexion (N) en un point d'accès à des services (N).

connexion multipoint : connexion comportant plus de deux extrémités de connexion.

Identificateur d'extrémité de connexion (N) CEPi : Identificateur de l'extrémité d'une connexion (N) destiné à identifier, en un point d'accès à des services (N), la connexion (N) correspondante.

Source de données (N) : entité (N) qui envoie des unités de données du service (N-1) sur une connexion (N-1) [CCITT X.200].

La correspondance des adresses N-SAP peut être soit une à une, soit hiérarchique, soit par des tables (cf. §1.4.1).

L'intérêt principal de SAP est de connaître globalement l'identité de l'émetteur ainsi que du récepteur pour pouvoir répondre. Et la structuration de l'adresse de SAP en traversant les différentes couches du modèle OSI se fait comme suit :

```
adresse_Entité(N)  =  Sélecteur(N) + adresse_Entité(N-1)
adresse_NSAP       =  Sélecteur_NSAP + adresse_LSAP
                   =  Sélecteur_NSAP + Sélecteur_LSAP + adresse_MAC
```

Nous rappelons que le terme SAP est souvent utilisé dans les services d'échange sans connexion où l'entité source doit ajouter son adresse SAP (le sélecteur) et celle de l'entité destinataire dans le même PDU alors que pour les échanges d'information en mode connecté, il suffit d'utiliser, à la place, le numéro de la connexion, qui identifie les deux extrémités de connexion.

L'OSI recommande la définition du mode d'échange de données, soit sans connexion (CLNS), soit avec connexion (CONS) dans les couches responsables de la transmission de données. Dans ce contexte, une connexion (de bout en bout) au sens OSI peut avoir lieu plus précisément soit au niveau de la couche réseau pour le service CONS, soit au niveau de la couche transport pour le service de type TP4[1].

3.1.2. Le mode d'échange de données dans FIP

Pour le cas de FIP, un Processus d'Application peut écrire ou lire une valeur quand il le veut. Ceci ressemble aux boîtes aux lettres où la communication est sans connexion alors que l'échange de données du niveau Liaison est avec connexion parce qu'un identifieur synchronisant le producteur et les consommateurs doit déclencher la phase d'ouverture de la connexion. La figure suivante montre ces deux types d'échange de données :

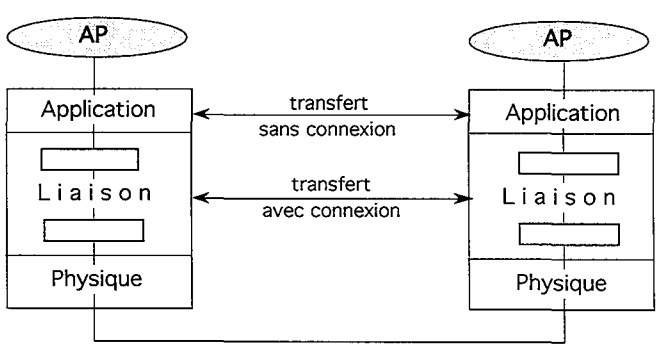


Fig. 3.1 — Les deux modes d'échange de données en FIP

1 couche transport de classe 4,

Dans ce contexte, une entité d'application désirant communiquer avec une entité homologue éloignée, doit désigner les SAPs source et destinataire ou bien le numéro de CEP selon le mode d'échange et selon la structuration de SAP ainsi que le CEPi en traversant les couches.

Selon l'OSI, l'échange de données en mode connecté doit respecter les trois phases suivantes :

(1) Établissement de connexion : dans cette phase s'effectue la négociation pour établir une connexion entre deux entités terminales. Et les primitives ainsi utilisées sont (*CONNECT_Rq, CONNECT_Ind, CONNECT_Rsp* et *CONNECT_Conf(+/-)*).

(2) Transfert de données : dans cette phase sont véhiculés les PDUs comportant les données. Et les primitives permettant d'effectuer cette phase sont (*DATA_Rq, DATA_Ind et DATA_Ack(+/-)*).

(3) Libération de la connexion : l'une des entités qui fait partie de la connexion peut demander de se déconnecter dans cette phase. Les primitives qui permettent de libérer la connexion sont (*DISCONNECT_Rq, DISCONNECT_Ind* et *DISCONNECT_Conf(+/)*).

Il reste à signaler qu'il existe bien d'autres primitives qui dépendent essentiellement de la Qualité de Service négociée comme le contrôle de flux, la récupération sur erreurs, la réinitialisation de connexion, la gestion de perte, de duplication et de déséquencement des PDUs que l'on peut introduire dans les différentes phases précédentes.

Pour le cas de FIP, la seule différence est que les primitives qui gèrent les connexions dépendent de la phase d'échange de données (cyclique, apériodique de messages ou apériodique de requêtes) [Soulas 89] et que la plupart des connexions sont de type multipoint. La sémantique de la connexion multipoint et la notion de SAP multipoint sont en cours de normalisation et font l'objet des travaux OSI actuels [ISO/IEC 8348], [ISO/IEC 9542]. Nous allons détailler dans la suite la correspondance multipoint des primitives précédentes dans le réseau FIP, selon la phase et le type du service.

3.1.2.1. Service concernant le trafic identifié

Les connexions sont préconfigurées et l'Arbitre de Bus en diffusant ID_DAT active la connexion multipoint entre le producteur et les consommateurs. Nous rappelons que puisque une seule connexion est active à un instant donné, le producteur utilise cette connexion pour envoyer la valeur associée à l'identifieur précédent sans désigner ni l'identifieur ni les destinataires. Le transfert de données se fait ainsi par l'émission de RP_DAT du producteur vers les consommateurs. Quant à la libération temporaire ou désactivation de la connexion, elle se termine normalement à la réception de RP_DAT par l'Arbitre de Bus. Sinon, ce sont les temporisateurs associés à l'AB et les entités consommatrices qui désactivent la connexion. La réception de ID_DAT suivante synchronisera les entités consommatrices et l'entité productrice. L'activation d'une connexion peut avoir lieu d'une façon périodique ou apériodique selon la phase de scrutation de l'AB et les demandes explicites.

3.1.2.2. Service concernant les listes d'identifieurs

Pour une mise à jour apériodique d'une liste d'identifieurs, la demande pour établir une connexion se fait à l'initiative d'un producteur qui en émettant une trame RP_DAT_RQi superpose sa demande sur la réponse, la prise en compte par l'AB et l'envoie de ID_RQi activera la connexion. Le transfert de la liste d'identifieurs est immédiat et est suivi par la

désactivation de la connexion par l'AB qui attend soit RP_RQi soit l'expiration d'un temporisateur qui lui est associé. La trame RP_RQi ne contient pas d'adresse de destinataire parce qu'elle emprunte la seule connexion active entre le producteur et l'AB.

Pour une mise à jour périodique, une configuration de la connexion chez l'AB lui permet d'activer périodiquement cette connexion sans avoir besoin d'une demande explicite du transfert.

3.1.2.3. Service concernant la messagerie

Comme pour les listes, une entité désirant envoyer un message doit soit l'associer à un identifieur cyclique correspondant à une connexion préétablie, soit demander la mise à jour d'une façon apériodique.

Dans les deux cas, l'activation de la connexion se fait à l'initiative de l'AB qui envoie ID_MSG à destination de toutes les entités. La connexion est établie entre la source de message (producteur de ID_MSG) et toutes les autres entités en tant que consommatrices du prochain message. Le transfert de données est caractérisé par l'envoi d'un message avec ou sans acquittement et éventuellement la réception d'acquittement ainsi que la désactivation de la connexion qui aura lieu soit à l'initiative du producteur qui envoie RP_FIN ou à l'initiative de l'AB qui envoie l'identifieur suivant au bout de certain temps.

Puisque le destinataire d'un message ne peut pas être connu à l'avance, le producteur doit utiliser l'adresse de messagerie pour sélectionner le(s) destinataire(s).

Nous proposons dans la suite une étude de tous les cas de figures possibles permettant d'identifier le type et le sens de connexion en fonction de SAP et de CEPi utilisés.

3.1.3. Architecture de SAP et de CEPi dans le réseau FIP

L'OSI a confié la normalisation de réseaux locaux (en autre terme, les deux couches inférieures des LAN) au comité 802 de l'IEEE. Ce comité a choisi de subdiviser la couche Liaison de données en deux sous-couches :

(1) La sous-couche MAC pour contrôler l'accès au support physique. Comme, par exemple, la circulation de jetons dans 802.4 et 802.5 ou la résolution de collision dans 802.3 ou enfin la détection d'erreurs et le codage des données en trames.

(2) La sous-couche LLC pour assurer le partage logique d'utilisation du support par les entités qui se trouvent au-dessus de la couche LLC. Trois types de contrôle logique ont été ainsi définis : LLC1 pour une transmission sans connexion et sans acquittement, LLC2 pour une transmission avec connexion et avec acquittement et enfin LLC3 pour une transmission sans connexion et avec acquittement.

Puisque l'interface entre MAC et LLC est symbolisée par les M_SAPs, nous abordons tout d'abrod le problème d'identification de M_SAP/CEPi et nous complétons ensuite le travail par la séparation des deux sous-couches MAC et LLC.

Dans tous les cas, une éventuelle structuration doit tenir en compte l'architecture réduite à trois couches de FIP, la diversité de types et de tailles des trames qui passent par le MAC, la coexistence de trois types de services (variables identifiées, messagerie et demandes explicites) qui doivent être reconnus dans le MAC et l'absence de l'adressage explicite dans le cas des variables identifiées. Ces spécificités ont rendu la tâche de structuration compliquée.

Nous nous appuyons dans la suite sur le champ contrôle et les identifieurs portés par les trames pour définir les SAPs et les CEPi.

3.1.3.1. Distinction de SAP par identifieur

Il s'agit de définir une entité LLC par identifieur (voir Fig. 3.2). Cette entité doit gérer tous les services associés à cet identifieur. Ces services qui dépendent du type d'identifieur peuvent concerner le buffer de production ou de consommation associé à l'identifieur, les files d'attente pour la liste des variables ou les messages associées à cet identifieur. Dans ce contexte, une adresse de SAP correspond à un identifieur alors que les CEPi qui sont gérés localement correspondent au champ contrôle et dépendent de la configuration de l'identifieur.

A partir du champ contrôle, une entité peut distinguer plusieurs types de connexions :

- ID_DAT : deux types de connexion multipoint et unilatérale peuvent avoir lieu en cas d'existence d'une entité associée à cet identifieur :
  - du producteur (CEP source) et vers cette entité en cas de configuration de l'identifieur en consommation,
  - de l'entité vers les consommateurs en cas de configuration de l'identifieur en production
- ID_RQi : en cas d'existence d'une entité produisant cet identifieur, une connexion bipoint et bilatérale entre cette entité et l'Arbitre de Bus aura lieu et pendant laquelle l'entité en réception de ID_RQi met à jour la liste d'identifieurs associés.
- ID_MSG : cette trame déclenche une connexion entre l'entité produisant cet identifieur et toutes autres entités. La source doit préciser l'adresse de SAP de destinataire à l'intérieur de la connexion ainsi que le type de la trame (avec ou sans acquittement) comme suit :
  - si la trame est RP_MSG_Ack alors une connexion bipoint et bilatérale avec le destinataire prend place. Cette connexion est utilisée par le destinataire pour envoyer la trame d'acquittement RP_Ack(-/+) sans désigner le destinataire.
  - si la trame est RP_MSG_NoAck alors une connexion bipoint (multipoint) et unilatérale entre le SAP de l'entité productrice et le(s) SAP(s) de(s) entité(s) destinataire(s) (faisant parties de l'adresse de groupe) prend place. L'hypothèse est faite pour qu'un seul M_SAP reçoit aussi bien les messages bipoint désignant une adresse individuelle que les messages multipoint désignant une adresse de groupe. C'est au niveau LLC qu'on doit définir un L_SAP par adresse de groupe.

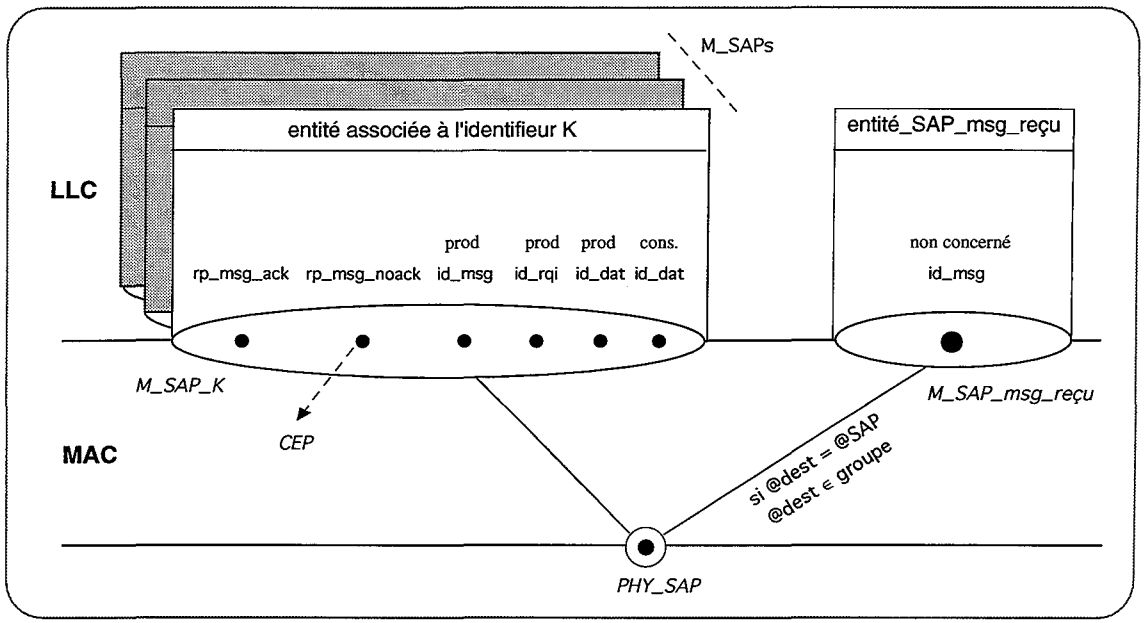


Fig. 3.2 — Représentation de M_SAP et CEPi

Par conséquent, le nombre de M_SAP sera égal au nombre des différents identifieurs reconnus en consommation et en production aussi bien pour le trafic identifié que la messagerie et les demandes explicites. On ajoute un SAP gérant les messages en entrée qui lui sont destinés.

Dans ce type de représentation, le MAC assure, en plus de la détection des erreurs, la reconnaissance des identifieurs (@SAP), le décodage des adresses individuelles ou de groupe ainsi que l'ajout de l'adresse physique qui correspond à l'adresse de messagerie en cas d'envoi de RP_MSG_xx (xx signifie avec ou sans acquittement).

Cette stratégie d'adressage peut causer une confusion pour la messagerie parce qu'une entité en étant une source d'un message doit désigner l'adresse source et l'adresse destinataire qui sont indépendants de l'identifieur (@SAP) associé à ID_MSG.

Il reste à signaler que cette solution s'avère non pratique car elle alourdit la couche MAC par un nombre important de M_SAPs ainsi que par des entités gérant les SAPs qui n'ont pas de différence fondamentale dans leurs opérations. Nous ajoutons aussi le critère séquentiel de FIP, deux transactions concernant deux identifieurs différents ne peuvent pas avoir lieu en même temps, ce qui fait que la coexistence de plusieurs entités se révèle non utile.

3.1.3.2. Distinction de SAP par type de service (3 M_SAP)

Dans ce modèle, trois types de SAP peuvent être identifiés selon le service assuré : M_SAP_Data, M_SAP_RQ et M_SAP_MSG. Dans le M_SAP_Data s'effectuent toutes les opérations liées à l'échange des variables identifiées qui sont pré-configurées, dans M_SAP_RQ s'effectuent toutes les opérations de demandes explicites (listes de variables, messages) alors que dans M_SAP_MSG se font les opérations qui relèvent d'envoi/réception de messages.

Cette structuration nous permet de définir un M_SAP par classe de service, de définir les CEPi comme étant les numéros des identifieurs dans les SAPs ce qui rend la connexion définie par le couple (M_SAP, No_identifieur) reconnue globalement, de reconnaître le champ contrôle dans le MAC et d' activer ainsi la connexion dans le SAP correspondant.

Il convient de signaler enfin que la coexistence dans la sous-couche LLC de deux types de service : messagerie et trafic identifié, qui sont différents du point de vue adressage et primitives de service, entraîne naturellement la définition de plusieurs M_SAPs.

La stratégie d'adressage retenue par FIP est celle d'un adressage habituel pour la messagerie et un adressage au moyen des identifieurs pour le trafic identifié. Dans le dernier cas, on ne peut pas localiser le producteur d'une variable sur le réseau ou d'autre manière si une entité de supervision se rend compte qu'il y a un producteur défectueux elle ne pourrait pas l'avertir par un message car elle ne peut pas déduire son adresse à partir de l'identifieur. Par contre, si l'adresse de M_SAP_Data était le résultat de concaténation de l'adresse physique avec le numéro de l'identifieur, on aurait pu déduire l'adresse de messagerie à partir de l'identifieur circulant.

3.1.3.3. Un seul SAP

Cette solution est inspirée de la plupart des réseaux de communication comme dans les réseaux locaux (bus à jeton, anneau à jeton et Ethernet) où un seul M_SAP est défini par une adresse physique. Cela est justifié par le fait qu'un seul type de service (allouer le médium pour transmettre une trame ayant une structure unique) est assuré dans le MAC qui ne tient compte à l'interface LLC que des adresses source et destination.

En adoptant cette structure pour le cas de FIP, nous aurons à identifier les CEPi selon le champ contrôle et les numéros d'identifieur (voir Fig. 3.3). Mais étant donné le fait qu'un SAP identifie normalement une entité, la sous-couche LLC sera vue comme une seule entité qui gère aussi bien le trafic identifié que la messagerie, deux fonctionnalités tout à fait indépendantes.

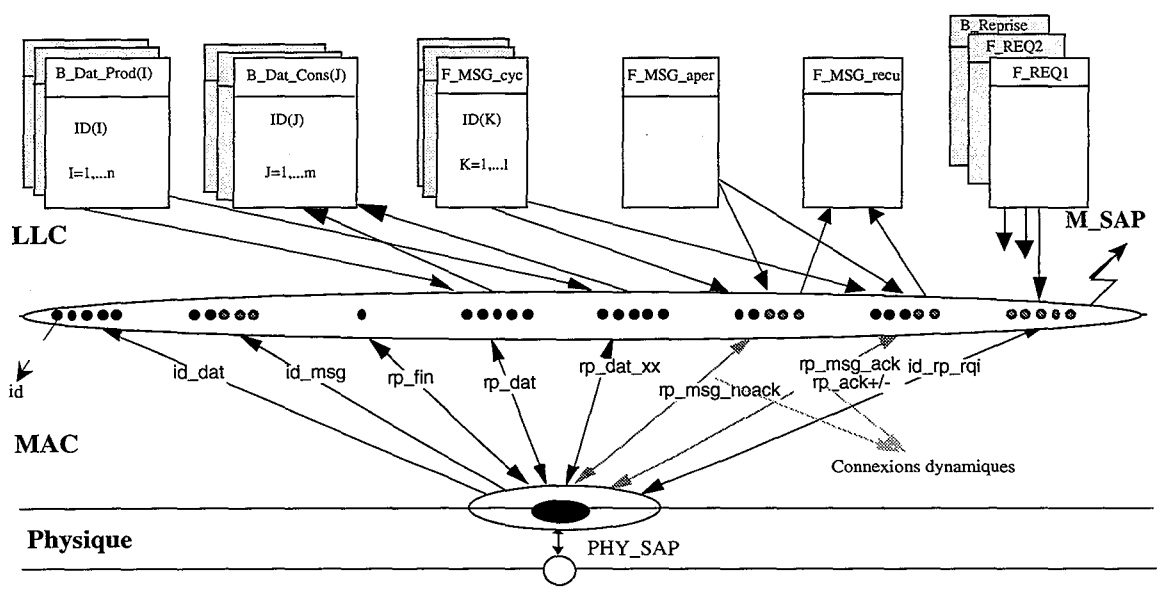


Fig. 3.3 — Modèle d'un seul M_SAP

Dans cette structuration, toutes les connexions sont prédéfinies et le rôle principal du MAC consiste à activer la connexion correspondant à une trame reçue à partir du champs contrôle et du numéro d'identifieur. Par exemple, en cas de réception de ID_DAT, le MAC active la connexion ID_DAT.id si elle existe, cela sera de la responsabilité du LLC de reconnaître l'identifieur en production ou en consommation. Pour le cas de la réception d'une trame message acquitté par le destinataire, le MAC active la connexion RP_MSG_Ack.

On remarque ici que l'acquittement est de niveau LLC, ce qui facilite la tâche du MAC puisqu'il ne doit mémoriser ni la dernière copie du message transmis, ni le compteur de reprise, etc.

3.1.4. Études comparatives

Nous avons introduit trois solutions possibles pour structurer les SAPs et les CEPs dans un environnement FIP. La solution qui consiste à définir un M_SAP par identifieur complique la tâche de la sous-couche LLC par un nombre important des entités qui n'ont pas de différence fondamentale et peut causer ainsi un conflit pour l'adressage de messagerie. La solution qui consiste à définir un M_SAP unique ou autrement dit, une correspondance point-à-point entre le SAP physique et le MAC_SAP s'avère intéressante lorsque la structure des trames est unifiée comme c'est le cas pour les normes IEEE concernant les RLs. Par contre, concernant le réseau FIP, pour lequel sont définis plusieurs types de trames et d'adressage, il faut définir des entités disposant de fonctionnalités différentes ainsi que de ressources différentes, et donc fournissant des services différents. Par conséquent, il vaut mieux définir plusieurs adresses de M_SAP.

Dans une troisième solution, nous avons proposé une structure de 3 M_SAPs; ce qui correspond bien aux critères mentionnés ci-dessus. Cette solution nous semble la plus adéquate soit pour localiser une entité selon le service qu'elle fournit, soit pour séparer les fonctionnalités différentes sur différents points d'accès aux services, soit pour distinguer les parties des trames locales au MAC de celles locales au LLC, soit enfin pour séparer l'aspect synchrone du MAC qui gère l'accès au médium de l'aspect asynchrone du MAC qui gère les demandes explicites de transfert de messages ou de buffers.

3.2. Identification de LLC/MAC

Le but de cette section consiste à identifier les deux sous couches MAC et LLC aussi bien dans une entité terminale que dans l'Arbitre de Bus. Il se trouve que cette distinction des sous-couches MAC et LLC de FIP n'ont pas encore été réalisées et la norme elle même ne fait pas la différence entre les deux sous couches.

Quant aux services MAC tels qu'ils sont définis dans [ISO 8802-5] et [ISO/IEC 802.1] "fournissent les moyens permettant aux entités (LLC) d'échanger des L_PDUs en faisant appel aux services MAC via des M_SDU (MAC Service Data Unit)".

Les trois services ainsi définis sont :

(a) MA_DATA_Request appelé par l'entité LLC afin de demander l'émission d'une L_PDU,

(b) MA_DATA_Confirmation reçue par l'entité LLC afin de transmettre l'acceptation ou le rejet de la requête précédente par l'entité MAC,

(c) MA_DATA_Indication afin de signaler à l'entité LLC l'arrivée d'une L_PDU au niveau de l'entité MAC (ce qui correspond à la réception d'une trame MAC).

Une identification informelle se trouve dans [Kachkachi 92] qui a utilisé ces primitives pour simuler un pont entre deux bus FIP indépendamment de la notion de SAP. Ce travail a servi comme point de départ pour notre étude qui relie chaque primitive à son point d'accès aux services SAP. Sachant que les paramètres de services ont été modifiés dans les étapes de vérification et surtout chez l'Arbitre de Bus pour lui permettre une mise à jour des identifieurs.

3.2.1. Pour une entité terminale

FIP fournit trois types différents de services à travers la couche Liaison de données : à savoir, le trafic identifié, le trafic de messagerie et le trafic engendré par les demandes explicites ou les listes des identifieurs. Trois types de points d'accès aux services SAP ont été précédemment définis. A l'intérieur de chaque SAP, les connexions (identifieurs) peuvent être gérées.

3.2.1.1. La structure de la sous-couche LLC

Nous avons choisi de structurer la sous-couche LLC en trois. Ces entités assurent la gestion d'accès sur les différents SAPs comme illustré dans la figure suivante :

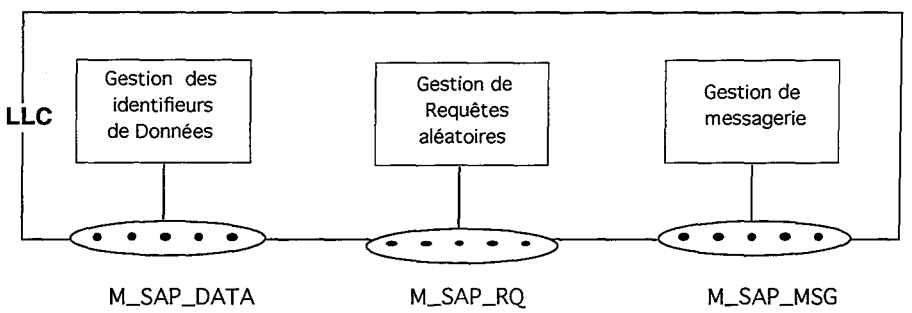


Fig. 3.4 — Architecture de LLC de FIP

Cette structuration tient compte des recommandations [IEEE 802.2] qui définissent trois types de composantes dans la sous-couche LLC (voir figure suivante) :

(1) Composante de station : Cette composante est responsable de la gestion des événements qui affectent la sous-couche LLC en sa totalité. Une composante de station doit être définie pour chaque MAC_SAP.

(2) Composante de lien de SAP : Cette composante est responsable de la gestion des événements qui affectent un SAP spécifique. Une composante doit être définie pour chaque L_SAP.

(3) Composante de connexion : Cette composante est responsable de la gestion des événements qui affectent une connexion spécifique du niveau liaison. Une seule composante de connexion doit être définie par connexion.

La structure hiérarchique des composantes est la suivante :

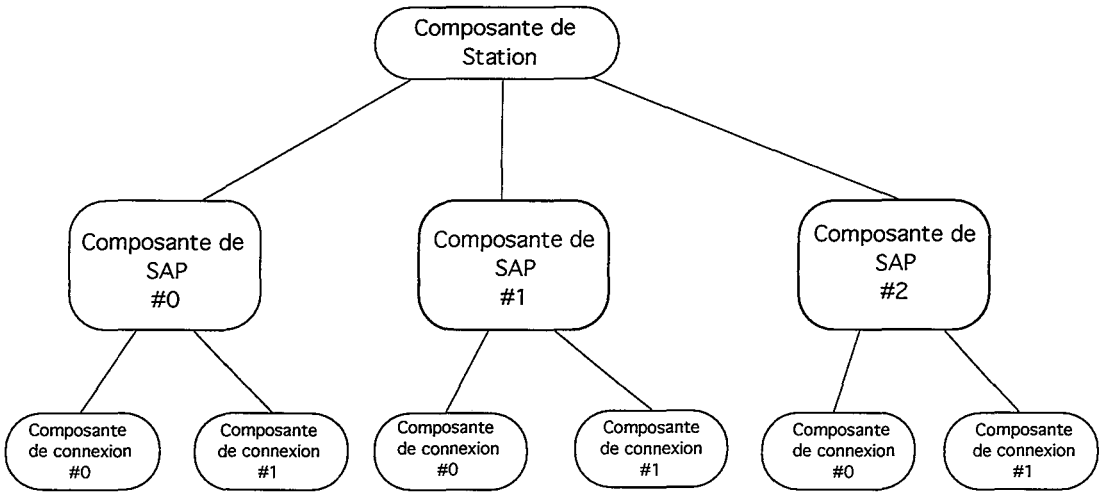


Fig. 3.5 — Les relations entre les composantes LLC de IEEE 802.2

Par conséquent, on aura besoin, pour le cas de FIP, de définir trois composantes de stations qui correspondent respectivement à M_SAP_Data, M_SAP_RQ et M_SAP_MSG. Et pour chaque composante de SAP, on aura à définir autant de composantes de connexions qu'il y aura d'identifieurs.

1. GESTION DES IDENTIFIEURS

Elle correspond à une composante de station relative à M_SAP_Data. Cette composante est responsable de la gestion des primitives suivantes :

- L_PUT.demande(identifieur, L_DATA)
- L_PUT.confirmation(identifieur, statut, L_DATA)
- L_SENT.indication(identifieur)
- L_RECEIVED.indication(identifieur)

Selon la valeur de l'identifieur, on peut désigner la composante de connexion correspondante. Par conséquent, le nombre de connexions sera égal au nombre des identifieurs configurés "produits" plus ceux configurés "consommés" par rapport à la station.

2. GESTION DES REQUÊTES

Elle correspond à une composante de station relative à M_SAP_RQ. Cette composante est responsable de la gestion des primitives suivantes :

- L_SPEC_UPDATE.demande(identifieur_spécifié, suite_identifieurs),
- L_SPEC_UPDATE.confirmation(identifieur_spécifié, statut),
- L_FREE_UPDATE.demande(suite_identifieurs, statut),
- L_FREE_UPDATE.confirmation(statut)

Les deux premières primitives (voir Fig. 2.9) permettent au producteur de l'identifieur_spécifié de superposer (avec confirmation) une demande de transfert d'une suite d'identifieurs (_RQ) sur une trame réponse RP_DAT. Les autres primitives permettent à l'entité initiatrice de superposer (avec confirmation) une demande de transfert d'une suite

d'identifieurs sur une trame réponse RP_DAT correspondant au premier identifieur produit et configuré "libre".

Selon la valeur de l'identifieur spécifié, on peut désigner une composante de connexion pour chaque identifieur spécifié. Quant à la demande libre (FREE), on peut toujours désigner une connexion statique avec un identifieur de valeur nulle. Cette connexion sera opérationnelle sur occurrence du premier identifieur configuré "libre". Et parce que les demandes libres s'écrasent (une nouvelle demande sur un identifieur remplacera une ancienne demande même si l'ancienne demande n'a pas encore été transmise) et que l'ensemble des identifieurs configurés "spécifiques" est indépendant de celui configuré "libre", nous n'aurons ni d'ambiguïté ni de duplication de demandes sur la même connexion.

Si nous avons N identifieurs configurés "spécifiés", nous aurons besoin de N+1 composantes de connexions pour cette composante de SAP (par rapport à la station).

3. GESTION DE MESSAGERIE

Elle correspond à une composante de station relative à M_SAP_MSG. Cette composante est responsable de gérer les primitives suivantes :

- L_MESSAGE.demande(identifieur_spécifié, @dest, @source, message),
- L_MESSAGE.indication(@dest, @source, message),
- L_MESSAGE.confirmation(identifieur_spécifié, @dest, @source, statut),
- L_MESSAGE_ACK.demande(identifieur_spécifié, @dest, @source, message),
- L_MESSAGE_ACK.indication(@dest, @source, message),
- L_MESSAGE_ACK.confirmation(identifieur_spécifié, @dest, @source, statut),

Dans ce cas, l'identifieur spécifié est configuré soit "cyclique", soit "apériodique". En tout cas, nous désignons la composante de connexion par la valeur de l'identifieur.

Par conséquent, le nombre de composantes possibles sera celui des identifieurs de messagerie cyclique plus celui des identifieurs configurés "apériodiques" dans le contexte d'une station.

N.B. : Nous n'avons pas réduit le nombre de connexions pour les identifieurs apériodiques à un seul parce que nous n'avons pas la possibilité d'écrasement ici comme dans le cas des requêtes.

3.2.1.2. La structure de MAC

Après avoir identifié les M_SAPs, nous pouvons formaliser les fonctions effectuées au niveau du MAC pour le réseau FIP comme le montre la figure suivante :

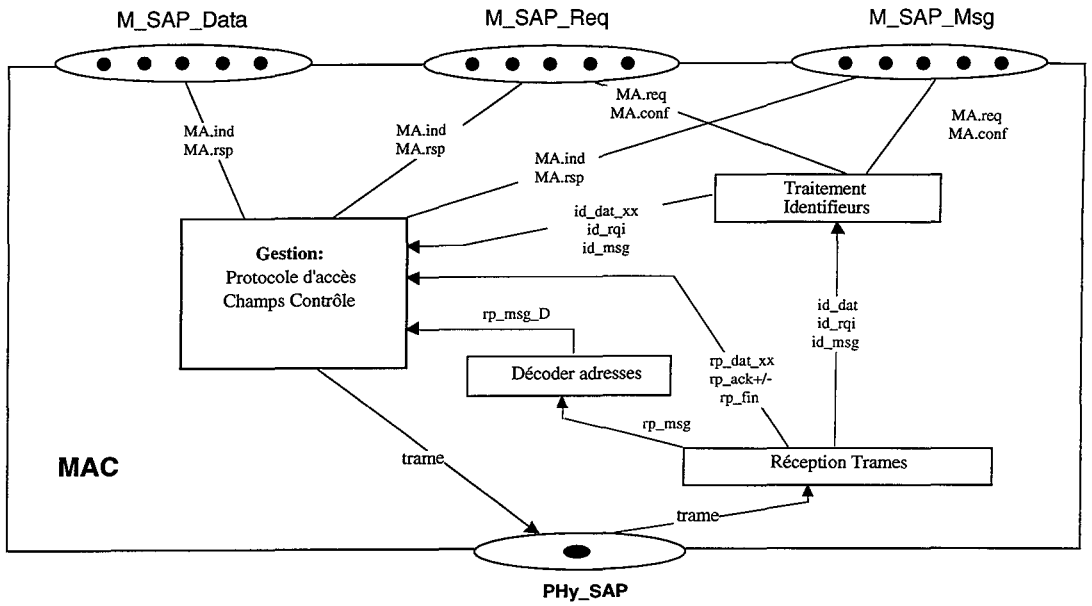


Fig. 3.6 — MAC de FIP

Selon la fonctionnalité rendue, la figure précédente définit les quatre machines suivantes :

1. MACHINE DE RÉCEPTION DE TRAMES

Cette machine est en charge de :

- la détection des erreurs physiques en ignorant les trames non valides,
- l'identification de trois types de trames et redirection des trames contenant les identifieurs ID_DAT, ID_RQ et ID_MSG vers la machine de traitement des identifieurs, les trames RP_MSG(Ack/NoAck) vers la machine de décodage d'adresses alors que les *autres* trames sont redirigées vers la machine du protocole MAC.

2. MACHINE DE TRAITEMENT DES IDENTIFIEURS

Cette machine possède trois interfaces : une interface avec la sous-couche LLC, une autre avec la machine de réception des trames et une interface avec la machine du protocole MAC. La première interface lui permet d'associer aux identifieurs produits, d'une façon asynchrone, les demandes explicites de mise à jour de messages (à l'issue du service MA.Req sur M_SAP_MSG) ou de liste de variables (à l'issue du service MA.Req sur M_SAP_RQ).

A réception et reconnaissance d'une trame identifieur ID_DAT, cette machine va tester s'il existe déjà une demande explicite associée à cet identifieur et avertir par suite la machine du protocole MAC de cette demande explicite pour l'aider à compléter la prochaine trame de réponse. Par exemple, s'il y avait une demande d'envoi de message sur l'identifieur No_id (c'est à dire MA.Req(No_id)), cette machine notifierait la machine de gestion du protocole MAC par le triplet (ID_DAT, No_Id, indicateur de message).

A réception et reconnaissance d'une trame de requêtes ID_RQ ou de messages ID_MSG, cette trame sera redirigée vers la machine de gestion de protocole MAC.

Des services de confirmation (MA.Conf) concernant les demandes indiqueront le bon ou le mauvais déroulement de ces services.

3. MACHINE DE DÉCODAGE D'ADRESSES

Cette machine est responsable de décoder l'adresse MAC contenue dans RP_MSG(@Dest) pour se reconnaître comme destinataire du message aussi bien pour un message bipoint que pour un message multipoint. Dans ce dernier cas, un test d'appartenance à l'adresse de groupe doit avoir lieu. Le résultat est dirigé vers la machine de gestion du protocole MAC même si on n'est pas destinataire.

4. MACHINE DU PROTOCOLE MAC

C'est la machine principale qui gère le contrôle d'accès au médium ainsi que les champs contrôle des trames. L'évolution interne de cette machine est illustrée par la figure 3.7.

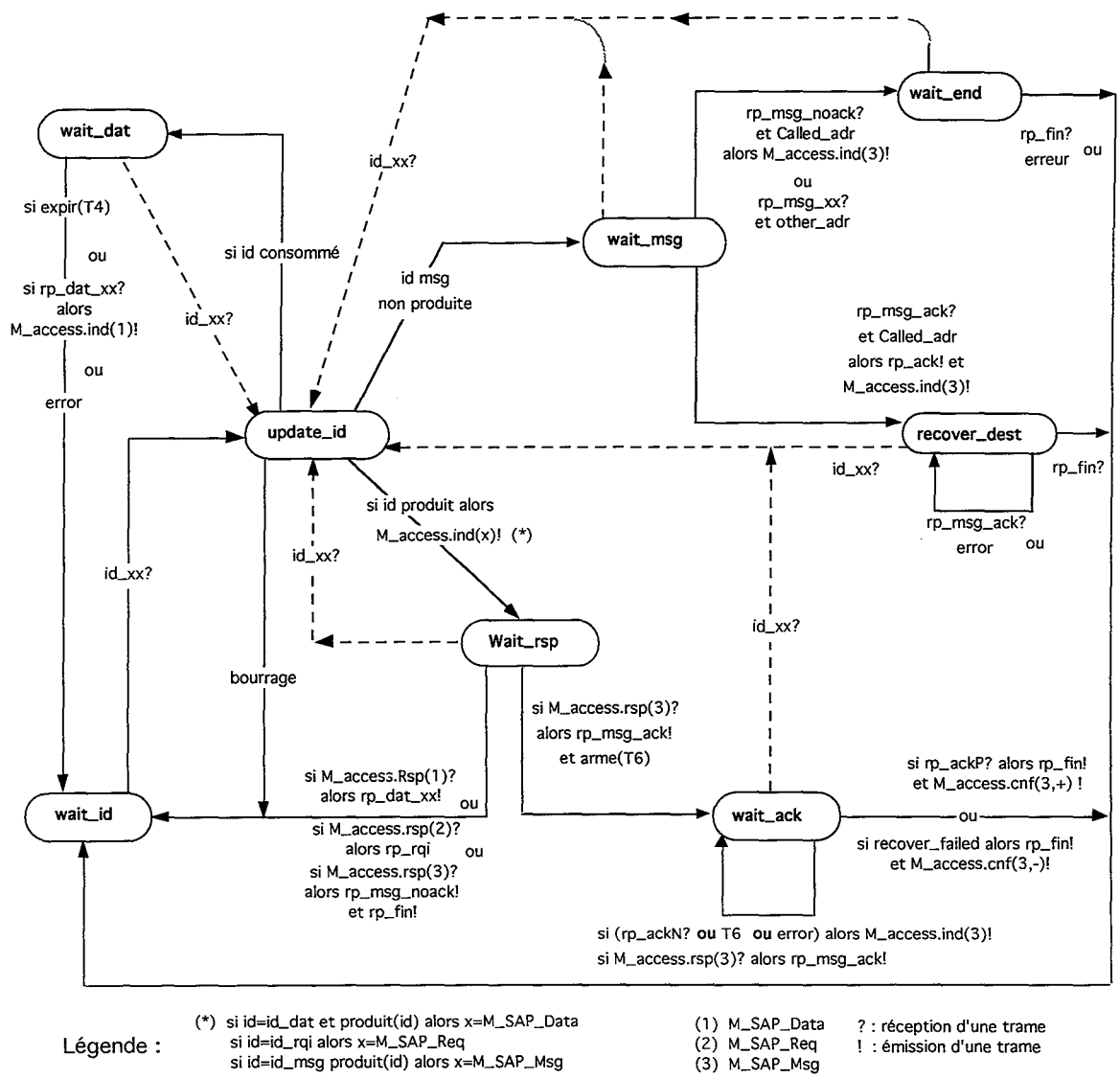


Fig. 3.7 — Évolution interne de la machine du protocole MAC

Explication :

MTI = Machine Traitement Identifieurs MA = Medium Access,
MRT = Machine Réception Trames, MDA = Machine de Décodage d'Adresses,
RP_DAT_xx appartient à l'ensemble {RP_DAT, RP_DAT_MSG, RP_DAT_RQ, RP_DAT_RQ_RQ}
Etat initial := wait_id {attente trame identifieur }
wait_id → update_id en réception d'une des trames identifieur suivantes :

- ID_DAT(id) de MTI
- ID_MSG(id) de MTI,
- ID_RQi(id) de MTI

update_id → Wait_rsp si l'identifieur (id) est produit alors tester le champ contrôle :

- si ID_DAT alors émettre MA.Ind sur M_SAP_Data et attendre la valeur associée
- si ID_MSG alors émettre MA.Ind sur M_SAP_MSG et attendre le message associé
- si ID_RQ alors émettre MA.Ind sur M_SAP_RQ et attendre la liste d'identifieurs

update_id → wait_msg s'il s'agit d'une trame de messagerie ID_MSG et si l'identifieur (id) n'est pas produit alors attendre message en étant un éventuel destinataire,

update_id → wait_dat s'il s'agit d'une trame de données ID_DAT et si l'identifieur (id) est consommé alors attend la valeur de la variable associée à id,

Wait_rsp → wait_id selon la réponse attendue du LLC :

- si réception de MA.Rsp(valeur) du M_SAP_Data alors émettre RP_DAT_xx(valeur)
- si réception de MA.Rsp(list_id) du M_SAP_RQ alors émettre RP_RQi(list_id)
- si réception de MA.Rsp(msg_noack, adr_S, adr_D, msg_bdy) du M_SAP_MSG alors émettre RP_MSG_NoAck(adr_S, adr_D, msg_bdy) suivi d'une trame RP_FIN,

Wait_rsp → wait_ack en réception de MA.Rsp(msg_ack, adr_S, adr_D, msg_bdy) du M_SAP_MSG alors émettre sur le médium RP_MSG_Ack(adr_S, adr_D, msg_bdy) et armer le temporisateur(T6),

wait_dat → wait_id retour à l'état initial :

- soit en réception de la trame RP_DAT_xx contenant la valeur associée à l'identifieur reçu précédemment et en émettant MA.Ind vers M_SAP_Data,
- soit en expiration du temporisateur (T4) associé à cette transaction

wait_msg → wait_end selon l'adresse destinataire du message alors :

- si réception d'une trame RP_MSG_noack de MDA dont on est destinataire alors émettre MA.Ind vers M_SAP_MSG, ou
- si réception d'une trame RP_MSG_noack de MDA dont on n'est pas destinataire

wait_msg → recover_dest {un algorithme réparti de reprise sur erreurs qui doit être exécuté entre la source, le destinataire et l'Arbitre de Bus prend place}, le déroulement normal est le suivant :

- si réception trame RP_MSG_ack dont le MAC est destinataire alors émettre RP_ack(+/-) et émettre MA.Ind vers M_SAP_MSG

wait_end → wait_id en réception de la trame RP_FIN de MRT

recover_dest → wait_id en réception de trame RP_FIN de MRT

wait_ack → wait_id {chez la source d'un message acquitté} selon que :

- si reception trame RP_ack(+) alors émettre RP_FIN sur le médium et MA.Conf(+) vers M_SAP_MSG,
- si la reprise sur erreurs est échouée alors émettre RP_FIN sur le médium et MA.Conf(-) vers M_SAP_MSG

Les transitions en pointillées représentent une synchronisation en réception d'une trame identifieur en provenance de l'Arbitre de Bus.

Nous détaillons dans la suite les services rendus par le MAC vers le LLC :

A) SERVICE DE REQUÊTE MA.RQ

La sémantique de ce service, qui est traité dans la machine de traitement des identifieurs, dépend essentiellement du MAC_SAP où ce service a été évoqué :

- MA.Rq(type, ID_number) : ce service, lorsqu'il est associé au MAC_SAP_Rq, provoque la demande de mise en circulation d'une liste d'identifieurs. Le paramètre *type* indique s'il

s'agit d'une demande de mise à jour libre ou spécifiée. Dans le cas d'une demande spécifiée, ID_number indique le numéro d'identifieur qui doit porter cette demande.

- MA.Rq((ID_number) : ce service, lorsqu'il est associé au MAC_SAP_MSG, permet à la sous-couche LLC de demander la mise en circulation d'un identifieur de messagerie.

B) SERVICE DE CONFIRMATION MA.CONF

Deux types de confirmation peuvent avoir lieu dans une entité terminale. Le premier indique la prise en compte des demandes explicites de mise en circulation des identifieurs alors que le deuxième indique le succès de la transmission des trame réponses RP_DAT, RP_DAT_XX, RP_RQi et RP_MSG (Ack ou NoAck).

- MA.Conf(statut) : ce service offert au MAC_SAP_RQ ou MAC_SAP_MSG, permet à l'entité de traitement des identifieurs d'indiquer si l'identifieur demandé est valide ou non,
- MA.Conf(statut) : ce service, lorsqu'il est associé au MAC_SAP_Data, permet à l'entité de gestion d'accès au médium de :
  - confirmer le succès ou l'échec d'émission de RP_DAT_XX sur MAC_SAP_Data et RP_RQi sur MAC_SAP_RQ,
  - confirmer que le message sans acquittement RP_MSG_NOAck suivi par RP_FIN ont été émis sur MAC_SAP_MSG,
  - confirmer le succès ou non de la transaction associée à la trame réponse RP_MSG_Ack. Soit la réception de RP_ACK dans l'échéance et soit l'échec de la reprise associé au message sur MAC_SAP_MSG.

C) SERVICE D'INDICATION MA.IND

Ce service est généré lors de la réception d'une trame identifieur après que l'entité de traitement des identifieurs ait reconnu l'identifieur en production ou en consommation :

- MA.Ind(ID_number , contrôle) : indique chez le MAC_SAP_Data la réception de la trame ID_DAT(ID). Le paramètre contrôle indique si l'entité est productrice ou consommatrice de cet identifieur.
- MA.Ind(ID_number, value) : indique chez MAC_SAP_Data la réception d'une valeur associée à l'identifieur ID_number déjà reconnu en consommation,
- MA.Ind(ID_number) : indique chez le MAC_SAP_RQ la réception d'une trame ID_RQi dont l'entité est productrice,
- MA.Ind(ID_number) : Indique chez le MAC_SAP_MSG la réception d'une trame ID_MSG produite dans cette entité,
- MA.Ind(Code, message, @source, @dest) : indique la réception d'un message avec ou sans acquittement (Code) dont l'entité est destinataire. Nous avons choisi de traiter l'acquittement dans le niveau MAC; celui-ci envoie une indication vers LLC et un acquittement vers la source s'il s'agit d'un message acquitté.

D) SERVICE DE RÉPONSE MA.RSP

Ce service est généré par le LLC sur les différents SAPs et peut avoir une des formes suivantes :

- MA.Rsp(value) : permet à l'entité LLC d'envoyer à travers le MAC_SAP_Data la valeur associée à un identifieur reçu par le service MA_Ind(ID_number, prod, cons),

- MA.Rsp(liste_ids) : permet à l'entité LLC d'envoyer à travers MAC_SAP_RQ la liste d'identifieurs souhaités,
- MA.Rsp(code_msg, msg_bdy, adr_source, adr_dest) : permet au LLC de demander la transmission du message *msg_bdy* ayant *code_msg* comme qualité de service (avec ou sans acquittement) vers le destinataire ayant *adr_dest* comme adresse de destination.

3.2.2. Pour l'Arbitre de Bus

En ce qui concerne l'entité Arbitre de Bus, nous l'avons subdivisé en deux parties, à savoir, la sous-couche MAC et les couches supérieures (sous-couche LLC et Application). Ceci nous permet de séparer les deux notions de la table de scrutation : statique imposée par le Processus d'Application de l'AB et dynamique imposée par les entités productrices. Elle permet aussi une mise à jour des identifieurs de la partie statique qui est indispensable, comme nous allons le voir, pour ajouter le pont.

Dans la mesure où plusieurs bus FIP doivent être interconnectés entre eux, des besoins concernants la mise à jour de la partie statique de la table de scrutation de l'AB se manifestent. Ces besoins impliquent l'ajout ou la suppression des identifieurs produits par les ponts liés à ce bus.

Pour ce faire, nous allons proposer dans la suite une structure de l'AB qui permet à la fois :

- ajout/suppression des identifieurs,
- auto-adaptation de la partie dynamique du MAC à ses modifications,
- identification des fonctions et des ressources qui correspondent au niveau MAC et les niveaux supérieurs.
- séparation des primitives de services qui sont à l'initiative du Processus d'Application AP de l'AB (autrement dit, la partie statique) et celles des entités productrices (partie dynamique).

3.2.2.1. Spécification de la sous-couche LLC et la couche Application

La sous-couche LLC et la couche Application de l'AB se composeront de :

- une table désignant les identifieurs produits dans le bus,
- la partie statique de la table de scrutation,

Les primitives rendues par la sous-couche LLC vers la couche Application constituent essentiellement l'insertion ou la suppression des identifieurs. Cette hypothèse pré-juge l'existence des trous dans la table de scrutation qui peuvent être comblés par des identifieurs ou même la réinitialisation d'une table.

3.2.2.2. Spécification de la partie MAC de l'AB

Quant aux primitives MAC, nous avons choisi deux types de primitives : soit celles de demande de service MA.REQ, soit celles de confirmation MA.CONF. Ces primitives s'exécutent pendant la phase statique de scrutation et ont la sémantique suivante :

1- MA.REQ(CODE, NO_ID, STATUT, LAST_ID)

Ce service permet au LLC de demander la mise en circulation de l'identifieur No_id. Elle lui permet également de préciser le statut de l'identifieur comme étant un nouvel identifieur

ou pas. Dans le cas d'un nouvel identifieur, le MAC doit ajouter l'identifieur dans sa table. Ce service est engendré à chaque fois que LLC reçoit un MA.Conf(Statut) associé à l'identifieur précédent. Trois types de demande peuvent être utilisés :

- Code = ID_DAT : correspond à la mise en circulation de la demande d'un identifieur de données.
- Code = ID_RQ : correspond à la demande de mise en circulation d'un identifieur de demande explicite.
- Code = ID_MSG : correspond à la demande de mise en circulation d'un identifieur de messagerie.

LLC doit être capable de notifier au MAC le dernier identifieur de la partie statique (ou Last_id = vrai).

2- MA.CONF(STATUT)

C'est la confirmation de l'exécution de la demande. La confirmation peut être soit locale dans le cas d'un identifieur invalide (e.g., ne figure pas dans la liste des identifieurs produits du MAC et le paramètre *Statut* ne précise pas un nouvel identifieur), soit distant après avoir reçu la réponse de l'entité distante. Cette dernière réponse dépend de la nature de la demande :

- Code = ID_DAT : C'est la réception de RP_DAT_XX qui déclenche la confirmation positive ou l'expiration de temporisateur associé à la réponse pour la confirmation négative.
- Code = ID_RQ : Nous considérons que c'est la demande de mise à jour d'une liste d'identifieurs. Donc, la confirmation doit indiquer si la liste d'identifieurs a été rafraîchie dans sa totalité.Dans ce contexte, la confirmation peut être soit positive lorsque toutes les réponses associées aux identifieurs de la liste ont été reçues dans les échéances (ou si la liste contenue dans RP_RQ est vide), soit négative si une réponse associée à un identifieur au moins n'a pas été reçue ou si la trame contenant la liste d'identifieurs RP_RQi n'a pas été envoyée dans l'échéance.
- Code = ID_MSG : La confirmation positive correspond à la réception de RP_FIN en provenance de la source de message alors que la confirmation négative sera émis lorsque le temporisateur associé à la messagerie expire sans que RP_FIN ne soit reçu.

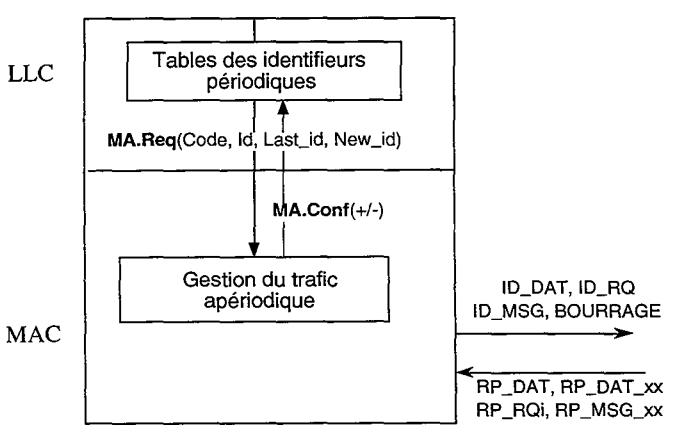


Fig. 3.8 — Modèle d'Arbitre de Bus

Les fonctions et les ressources nécessaires pour assurer le fonctionnement normal du MAC de l'Arbitre de Bus sont les suivantes :

1 - Une table représentant les identifieurs les plus récemment utilisés (initialement vide). Quant aux identifieurs qui ne circulent plus sur le réseau, des mécanismes de temporisation associés à chaque identifieur seront mis en place pour les supprimer au fur et à mesure de la table.

2 - trois files d'attente, F_IDREQ1, F_IDREQ2 et F_IDMSG sont utilisées pour assurer l'allocation dynamique des listes des identifieurs,

3 - deux files d'attente, B_REPRISE et F_RPRQ, pour assurer la reprise des identifieurs aussi bien dans la partie statique (F_REPRISE) que dans la partie dynamique (F_RPRQ).

4 - une machine pour gérer l'évolution interne de la table de scrutation dynamique en fonction de demandes explicites qui proviennent du réseau.

Il faut noter le fait que la structure du MAC, tout à fait indépendante du celle du LLC, nous permet de l'appeler un MAC auto-adaptatif dans le sens que nous n'avons plus besoin de configurer le MAC même dans le cas d'insertion des identifieurs.

Nous préciserons dans la figure suivante la machine à états modifiée de la Norme Liaison qui gère l'évolution interne du MAC afin de partager les accès au médium pendant la scrutation statique parmi les producteurs potentiels des identifieurs.

Il faut bien noter que les différents états restent inchangés pendant la scrutation apériodique et seuls les services de confirmation et de demandes de mise à jour des identifieurs vont disparaître.

Nous rappelons qu'une spécification formelle de MAC chez l'AB (pendant la phase statique et dynamique) et chez une entité terminale (les quatre machines) est abordée dans le chapitre 5.

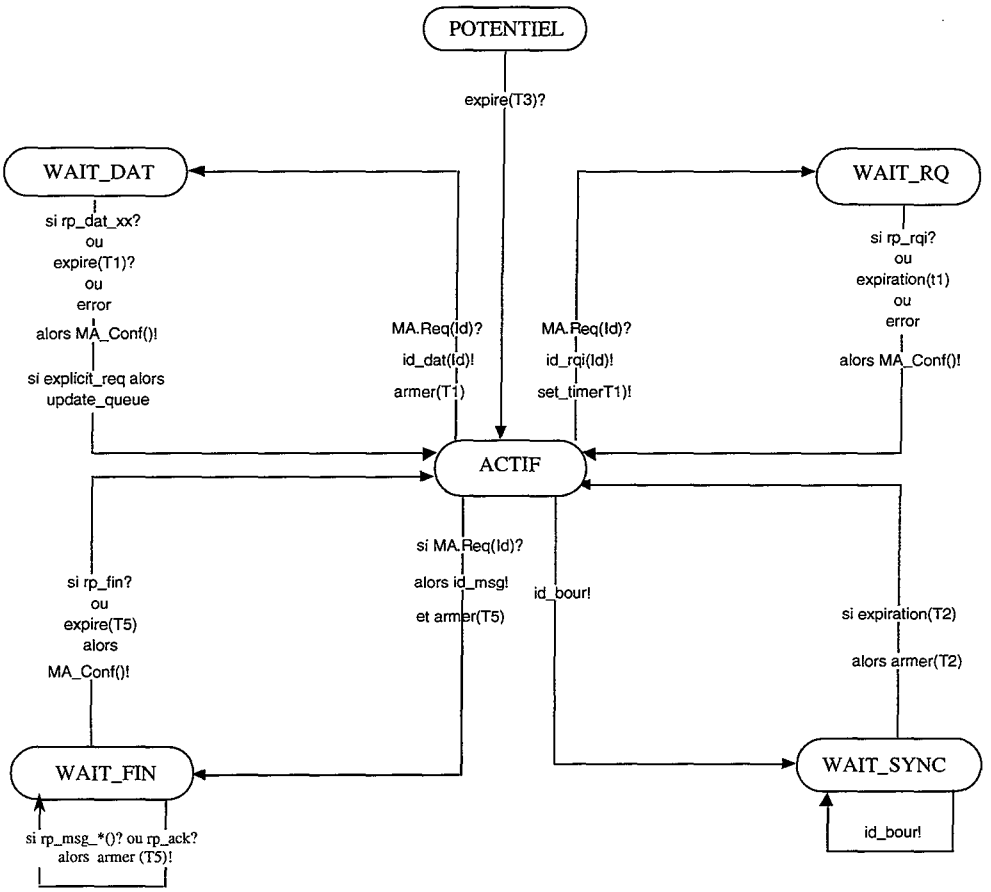


Fig. 3.9 — Évolution interne du MAC de l'AB pendant la scrutation statique

Explication :

Plusieurs ABs peuvent être configurés dans l'état potentiel et un seul dans l'état initial ACTIF.

ACTIF → WAIT_DAT : sur réception du LLC d'une demande d'accès MA.Req("data", Id) contenant l'identifieur cyclique à faire véhiculer alors émettre sur le médium la trame ID_DAT(Id) et armer le temporisateur T1,

ACTIF → WAIT_RQ : sur réception de MA.Req("Request", Id) alors émettre la trame ID_RQi (i = 1 pour demande urgente et 2 pour demande normale) et armer le temporisateur T1,

ACTIF → WAIT_FIN : sur réception de MA.Req("Msg",Id) alors émettre la trame ID_MSG sur le médium et armer le temporisateur T5,

ACTIF → WAIT_SYNC : l'Arbitre de Bus envoie une trame de bourrage sur le médium,

WAIT_DAT → ACTIF : selon la trame réponse reçue :

- si RP_DAT alors émettre MA.Conf(+) vers LLC,
- si RP_DAT_xx alors émettre MA.Conf(+) vers LLC et stocker identifieur dans une file d'attente correspondant à la demande explicite,
- si expiration temporisateur(T1) alors émettre MA.Conf(-) vers LLC

WAIT_RQ → ACTIF

- si réception RP_RQi alors émettre MA.Conf(+/-) vers LLC,
- si expiration temporisateur T1 alors émettre MA.Conf(-) vers LLC

WAIT_FIN → ACTIF

- si réception trame RP_FIN alors émettre MA.Conf(+) vers LLC,
- si expiration temporisateur T5 alors émettre MA.Conf(-) vers LLC

WAIT_SYNC → ACTIF : sur expiration du temporisateur T2 (micro-cycle) alors réarmer le temporisateur pour un nouveau micro cycle

Chapitre 4

Interconnexion de réseaux FIP

4.1. Besoins et problématique d'interconnexion

L'installation d'un seul bus FIP est devenue insuffisante pour répondre à tous les besoins de communication, qui peuvent concerner :

1. le nombre des identifieurs périodiques qui peuvent être échangés sur un bus,
2. le nombre de stations composant l'application peut dépasser le nombre maximal de stations raccordables à un bus FIP,
3. l'étendue géographique couverte par l'application dépasse la longueur maximale d'un bus,
4. améliorer la performance d'un bus en terme de trafic variables, messages et liste de variables apériodiques.

Pour faire face à ces exigences, nous allons proposer dans la suite des solutions permettant l'interconnexion des bus FIP par l'intermédiaire des ponts. Habituellement l'interconnexion de réseaux identiques ne pose pas beaucoup de problèmes. Cela revient de la réduction des fonctions de conversion grâce à la similitude des formats de trames, des protocoles ainsi que des modes de communication et d'adressage. Malheureusement, ce n'est pas le cas pour le bus FIP, cela étant dû à plusieurs raisons :

- la cohabitation de deux modes d'adressage complètement différents (l'adressage objet et l'adressage station),
- la coexistence de plusieurs types de trames (objets identifiés, messages, listes d'objets identifiés, ...) sur le médium; ces trames pouvant être adressées à un ou plusieurs destinataires explicitement ou implicitement désignés,
- l'existence de deux types de trafics, l'un périodique, l'autre apériodique, tous les deux étant applicables aux objets identifiés et aux messages,

- l'accès au médium est centralisé et géré par une station dite arbitre de bus qui doit connaître, avant le démarrage de l'application, les périodes de tous les trafics périodiques.

Les solution présentées dans cette section sont fondées sur l'identification LLC/MAC ainsi que la structure de 3 M_SAPs que nous venons d'étudier. Elles tiennent compte également des points précédents en cas d'interconnexion de bus FIP.

Une telle solution doit permettre aux entités appartenant à une topologie maillée de bus FIP interconnectés par des ponts de communiquer. Cette communication doit fournir les moyens :

- d'étendre le modèle PDC à l'interconnexion de bus : les consommateurs peuvent appartenir à d'autres bus que le producteur,
- d'étendre le service de messagerie point-à-point non acquitté à l'interconnexion,
- d'étendre le service de messagerie multipoint à l'interconnexion,
- de garder l'autonomie de chaque bus,
- de ne pas dégrader la performance de chaque bus,
- de garder tant que possible le principe de transparence. Par conséquent, les entités qui appartiennent aux différents bus ne doivent pas prendre en considération l'existence des ponts,
- les chemins et les ressources utilisés pour amener un message ou une variable à ses destinataires doivent être optimaux.

4.2. Hypothèses

Nous supposons que les hypothèses suivantes sont satisfaites dans notre étude :

(1) la topologie d'interconnexion est donnée à l'avance comme un besoin applicatif et nous ne cherchons pas à l'optimiser. Et il est préférable de reconstruire les différentes tables de routage en cas de la modification de cette topologie.

(2) la distribution du producteur et des consommateurs, pour chaque variable, est connue à la configuration et elle est également statique,

(3) les adresses de groupes pour les messages multipoints sont statiques et sont définies avant le démarrage de l'application. Nous ajoutons que la norme ne prévoit aucune modification sur les groupes pendant l'exécution de l'application. Le type des groupes qui ne sont pas abordés dans la norme sont supposés ouverts (i.e., toute station peut envoyer un message vers un groupe et seuls les membres de ce groupe reçoivent une copie du message précédent),

(4) les débits des bus sont égaux,

(5) les identifieurs sont globaux. Cette hypothèse peut être réduite aux identifieurs des variables produites seulement (on peut définir toujours le même identifieur en messagerie ou requête cyclique dans deux bus différents).

L'hypothèse (1) est justifiée par le fait qu'un site industriel reste stable pendant une longue période de temps et les modifications sont rares. L'hypothèse (2) est inhérente à un réseau FIP mono-segment. Traiter l'hypothèse (3) entraîne l'ajout d'une nouvelle couche pour gérer l'appartenance aux groupes qui n'entre pas dans notre préoccupation actuelle.

Si les bus peuvent avoir des débits hétérogènes, des situations de congestion peuvent avoir lieu pour le trafic de messagerie, et de retard pour les variables identifiées.

La cinquième hypothèse est introduite seulement pour simplifier le raisonnement. Un mécanisme de conversion des identifieurs peut se substituer à cette hypothèse.

4.3. Résolution du problème d'adressage

Nous allons résoudre tout d'abord le problème de l'adressage, pour pouvoir décrire ensuite les ressources et le comportement interne du pont [Saba 95b].

4.3.1. Adressage pour les objets identifiés

Il s'agit ici de variables désignées par des identifieurs globaux. Dans un seul bus, la production et la consommation d'une variable se fait dans une seule transaction : l'AB diffuse une trame ID_DAT contenant l'identifieur de la variable; le producteur diffuse la valeur associée à cette variable; les consommateurs reçoivent cette valeur. L'identifieur, dans ce cas, peut être considéré comme une adresse de groupe de destinataires (qui sont les consommateurs) pour lequel une seule entité, le producteur, peut utiliser cette adresse.

Lorsqu'un objet est produit sur un bus et est consommé sur d'autres bus, les ponts qui interconnectent les différents bus et les arbitres de bus des différents bus doivent être configurés en tenant compte de ce type d'objet. Ainsi, pour permettre le passage d'un objet X d'un bus B1 vers un bus B2, le pont reliant ces deux bus doit être configuré en tant que consommateur de X, pour le bus B1, et en tant que producteur de X, pour le bus B2.

Nous avons choisi de représenter l'ensemble des bus interconnectés et des objets qui y circulent à l'aide d'un graphe d'interconnexion défini comme suit :

- chaque bus FIP est représenté par un noeud,
- chaque pont entre deux bus est représenté par un arc entre les noeuds associés aux bus concernés.

L'exemple suivant (Fig. 4.1) permet d'illustrer le passage d'un ensemble de six bus interconnectés à un graphe d'interconnexion. Cet exemple montre que l'acheminement de l'objet α, produit sur le bus B1, vers les bus B3, B4 et B5 où se trouvent les consommateurs est défini par l'arbre représenté en gras. Le pont C doit être configuré en tant que consommateur de α pour le bus B1 et en tant que producteur de α pour le bus B2. Un raisonnement similaire s'applique aux ponts B, D, E et H.

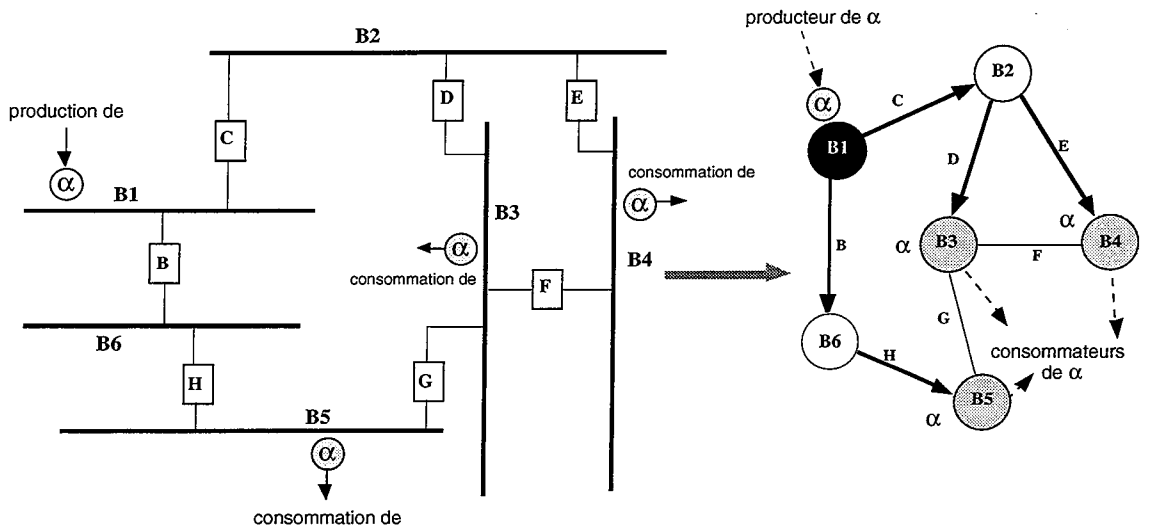


Fig. 4.1 — Exemple de passage d'un ensemble de bus à un graphe d'interconnexion

Le graphe d'interconnexion est généralement maillé. Par conséquent, plusieurs chemins peuvent être choisis pour véhiculer un objet du producteur à un consommateur. A chaque objet peut donc être associé un arbre représentant les différents chemins possibles. La racine de cet arbre est le noeud représentant le bus auquel est connecté le producteur, et les feuilles correspondent aux différents noeuds associés aux bus auxquels sont connectés les consommateurs (on peut avoir un consommateur qui appartient à un nœud intermédiaire). Comme les producteurs et les consommateurs sont définis de manière statique, les différents arbres de routage des objets sont connus de manière statique aussi.

Dans un pont transparent, c'est-à-dire auto-adaptatif, le pont apprend la localisation des stations au fur et à mesure que les trames le traversent [IEEE 802.1], [Perlman 85]. Ce type de pont n'est pas adapté à l'interconnexion de bus FIP, parce que le pont ne peut pas savoir quel consommateur est associé aux objets diffusés à partir du moment où les objets diffusés ne contiennent pas l'adresse de destination. Par ailleurs, nous noterons que les ponts utilisant le routage par l'émetteur [IEEE 802.5] sont difficiles à généraliser pour les adapter aux objets ayant plusieurs consommateurs parce que dans ce type de routage, la source utilise la diffusion pour localiser ses destinataires ce qui est côuteux, par ailleurs il risque d'avoir des boucles.

4.3.2. Adressage pour les messages

Dans le cas d'un seul bus FIP, une transaction de message s'effectue de la manière suivante :

- l'arbitre de bus émet une trame ID_MSG contenant l'identifieur permettant de transporter le message,
- ensuite le producteur du message émet une trame RP_MSG contenant le message.

Contrairement aux variables pour lesquelles les consommateurs se reconnaissent à partir de l'identifieur diffusé, les consommateurs d'un message se reconnaissent en testant l'adresse de destination contenue dans le message. L'adresse de destination d'un message est composée soit d'un numéro de bus et d'une adresse de station à l'intérieur de ce bus pour les messages en point à point, soit d'une adresse de groupe pour les messages en

multipoint. Pour permettre l'échange de message en point à point en cas d'interconnexion de bus FIP, nous retenons la solution suivante :

chaque pont reliant plusieurs bus sur lesquels sont connectés, de part et d'autre, des producteurs et des consommateurs de messages, dispose d'une table de routage. Cette table permet de déterminer le port de sortie (c'est-à-dire, le bus vers lequel il faut faire passer le message) à partir de l'adresse de destination du message. Ainsi, le pont devient consommateur de messages d'un côté et producteur de l'autre. Il faut enfin configurer les arbitres de bus pour tenir compte des messages qui transitent par les ponts.

Dans le cas d'un message multipoint, la transaction s'effectue de manière analogue au cas précédent, mais le pont peut avoir un ou plusieurs ports de sortie. Ainsi, pour un même message, le pont devient producteur autant de fois qu'il y a de chemins reliant le producteur initial du message aux consommateurs appartenant à l'adresse de groupe contenue dans le message émis.

Par analogie avec les adresses de groupes utilisées pour les objets identifiés, nous sommes amenés ici à construire un arbre pour chaque message destiné à une adresse de groupe. Malheureusement, la source de l'adresse de groupe ainsi que le droit d'utiliser une adresse de groupe n'ont pas été abordés dans les normes. Par conséquent et en se situant dans le cas le plus général où n'importe quelle station peut utiliser l'adresse de groupe, il faut construire un arbre par source recouvrant cette adresse. Dans ce cas, la décision de laisser passer un message dépend non seulement de l'adresse destinataire mais aussi de l'adresse source.

La figure suivante illustre la construction progressive d'une forêt de 6 arbres qui sont associés à chaque bus en tant que source de message de groupe. Nous reportons les critères d'optimisation des différents arbres au chapitre 6.

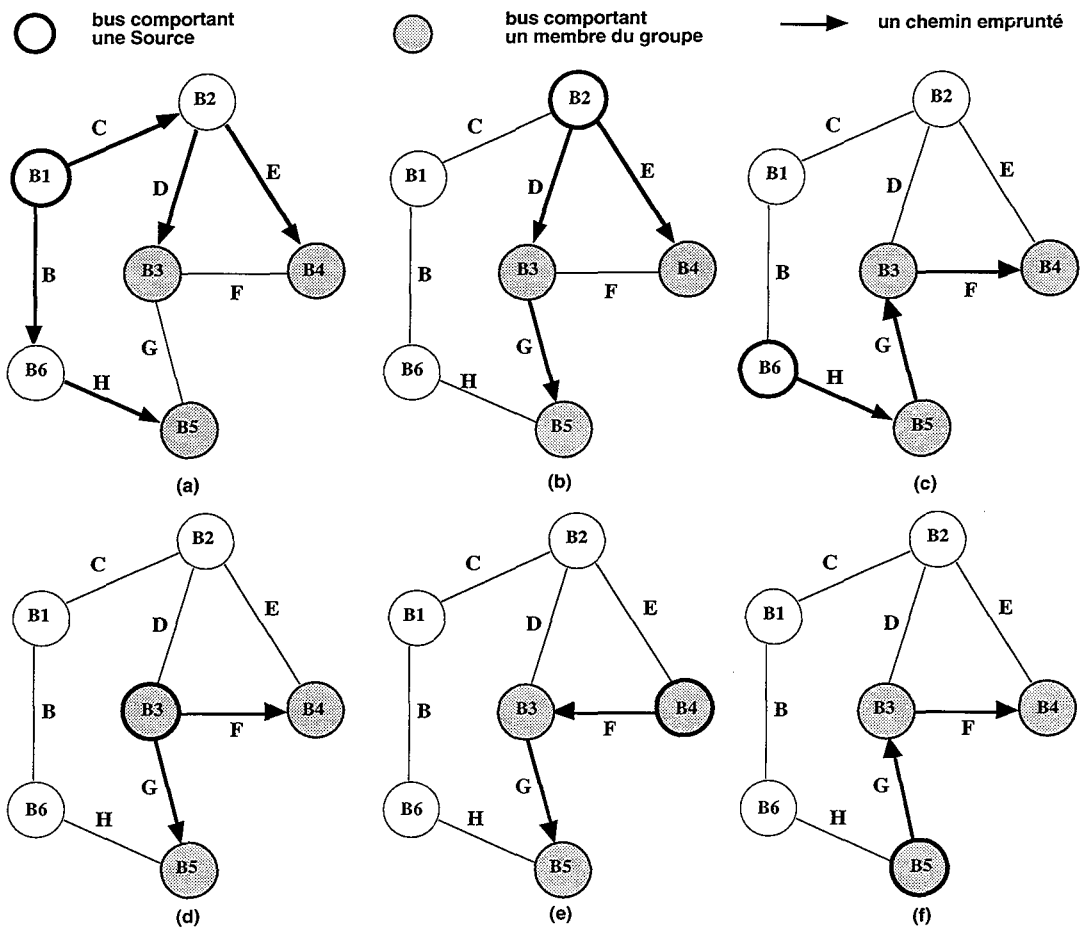


Fig. 4.2 — Utilisation d'une adresse de groupe par tous les bus

4.2.a correspond à n'importe quelle source qui appartient au bus B1 et au groupe G réparti sur les bus B3, B4 et B5. Dans cette étape, nous configurons les ponts, selon le sens de la flèche, C, D, E, B, H comme destinataires du couple (B1, G),

4.2.b correspond à n'importe quelle source qui appartient au bus B2 et au groupe G réparti sur les bus B3, B4 et B5. Dans cette étape, nous configurons les ponts, selon le sens de la flèche, D, E, G comme destinataires du couple (B2, G).

A l'issue de la sixième exécution, le port sur B3 du pont F va acheminer les messages vers les membres du groupe seulement si la source appartient à l'ensemble {B3, B5, B6} et sur le port de B4 si la source appartient à B4.

4.4. Fonctionnement interne des ponts FIP

4.4.1. Trafic d'objets identifiés

Pour assurer le trafic d'objets identifiés, nous proposons deux types de solutions : une solution basée sur l'introduction d'objets identifiés dans les ponts et une autre basée sur les listes d'objets.

1) La première solution est fondée sur le principe suivant :

tout objet identifié qui doit traverser des ponts doit être intégré à toutes les tables des arbitres de bus traversés, du bus source au bus destination. Ainsi, il faut ajouter un identifieur, par pont, pour tout objet devant passer d'un bus à un autre.

Pour atteindre ses consommateurs, la valeur d'un objet identifié doit traverser tous les ponts (en utilisant une suite de ID_DAT et RP_DAT). Par exemple, si un objet X doit aller d'un bus B1 à un bus B3 en passant par deux ponts, un pont P1 reliant les bus B1 et B2 et un pont P2 reliant les bus B2 et B3 : les transactions suivantes sont initiées :

1) un ID_DAT suivi d'un RP_DAT sur B1 permet au producteur de diffuser la valeur de X
2) un ID_DAT suivi d'un RP_DAT sur B2 permet au pont P1 de diffuser la valeur de X
3) un ID_DAT suivi d'un RP_DAT sur B3 permet au pont P2 de diffuser la valeur de X qui va atteindre le consommateur concerné.

En général, chaque pont se trouvant dans l'arbre d'acheminement d'un objet participe à deux transactions ID_DAT suivie de RP_DAT (une pour récupérer la valeur de l'objet à partir du bus en amont et une transaction pour diffuser la valeur de l'objet en aval).

C'est une solution statique parce qu'il s'agit de trouver les arbres d'acheminement pour chaque objet et configurer ainsi les Arbitres de Bus appartenant aux bus qui sont des nœuds de ces arbres pour tenir compte de la périodicité de l'objet. Si un AB au moins est saturé ou la partie cyclique de scrutation est pleine, cette solution ne peut plus garantir le transfert des objets.

Ce principe est simple à implanter et il est surtout adapté aux variables périodiques. Cependant, il conduit à une augmentation de la taille des tables de scrutation des arbitres de bus.

2) La deuxième solution est fondée sur le principe suivant :

l'ensemble des identifieurs d'objets en provenance du bus en amont et à destination du bus en aval sont rassemblés dans une liste d'objets. Ceci conduit à déclarer un identifieur cyclique de liste par port de sortie du pont. Ainsi, le pont rassemble les identifieurs correspondant aux valeurs récemment reçues du port d'entrée dans une liste pour une prochaine demande de mise à disposition de cette liste sur le port de sortie.

Avec cette deuxième solution, le délai de passage vers un bus est majoré par la période de l'identifieur de la liste de variables configuré pour le pont.

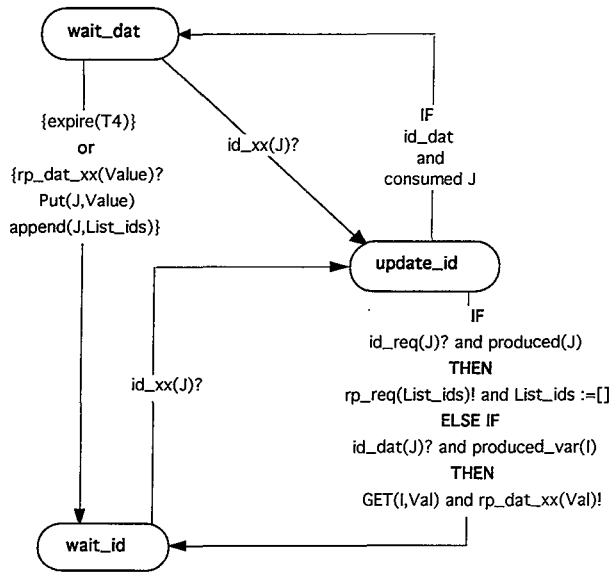


Fig. 4.3 — Machine à états associée à un port du pont de la deuxième solution

Explication :

Légendes : ! : émission ? : réception T4 : temporisateur pour rp_dat

Etat initial := wait_id,

wait_id → update_id en réception d'une trame identifieur (J)

update_id → wait_dat si la trame est ID_DAT et l'identifieur J est consommé

update_id → wait_id selon la trame reçue :

- s'il s'agit d'une trame de requêtes (ID_RQ) et J est un identifieur cyclique pour le port alors émettre la trame RP_RQ(List_ids) contenant la liste d'identifieurs correspondant aux variables produites et qui sont déjà stockées dans la liste et vider ainsi cette liste,
- s'il s'agit d'une trame ID_DAT et l'identifieur J correspond à une variable produite alors lire la valeur contenue dans le buffer partagé (associé à J) et émettre la trame RP_DAT_xx(Valeur). RP_DAT_xx peut être soit RP_DAT s'il n'existe pas de message en attente ou RP_DAT_MSG sinon,
- (voir paragraph suivant pour une trame de mesagerie),
- si le port n'est pas concernée par l'identifieur,

wait_dat → wait_id le port est en état d'attente d'une réponse de données :

- en réception de la trame réponse RP_DAT_xx contenant la valeur associée à l'identifieur J, alors écrire cette valeur dans le buffer partagé par tous les ports (associé J), et ajouter l'identifieur J dans la liste d'identifieurs de chaque port producteur de l'identifieur J pour une prochaine demande de liste,
- expiration du temporisateur T4

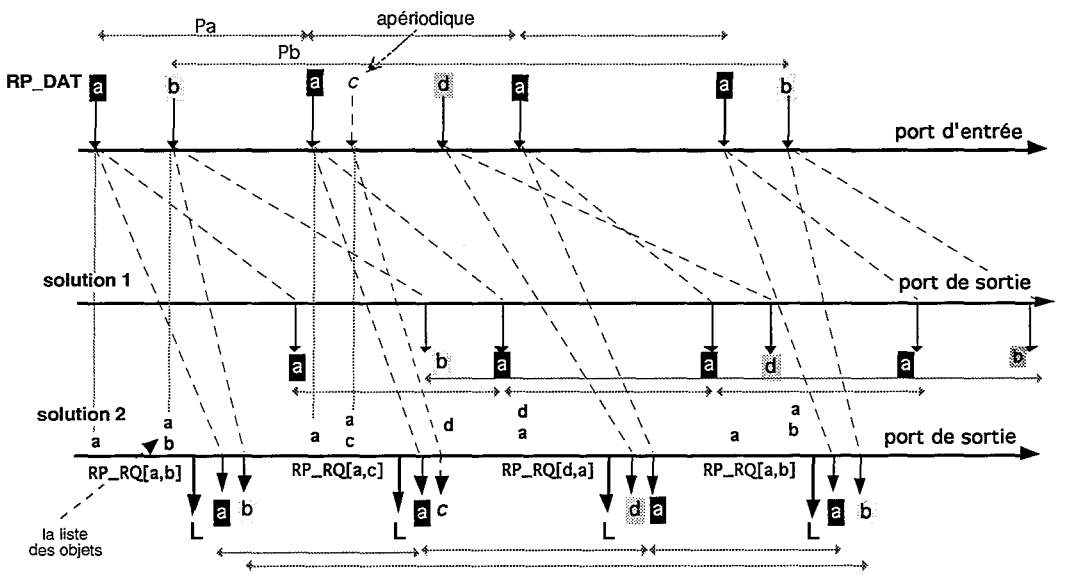
Avec la première solution, pour chaque objet transitant par un pont, il faut introduire un nouvel identifieur dans la table de l'AB auquel est relié ce pont. Avec la deuxième solution, il faut introduire un identifieur de liste d'objets par port de sortie du pont. Ainsi, la deuxième solution permet de minimiser les identifieurs à rajouter aux AB des bus à interconnecter. On peut montrer aussi que les délais de transfert des objets sont aussi optimisés dans le cas de la deuxième solution.

La figure 4.4 montre un exemple de comparaison entre les deux solutions proposées. Cette figure ne traite que le cas d'un seul pont :

- la première ligne représente l'arrivée des valeurs d'objets sur le port d'entrée du pont,

- la deuxième ligne représente les instants d'émission des valeurs par le port de sortie du pont dans le cas de la première solution,
- la troisième ligne représente les instants d'émission des valeurs par le port de sortie du pont dans le cas de la deuxième solution,
- les périodes des variables "a", "b" et "d" sont identiques aussi bien pour le bus producteur (ligne 1) que pour le bus consommateur,
- la période de l'identifieur "L" de demande explicite de transfert de la liste de variables pour le pont a été choisie égale à la plus petite période des variables (c'est-à-dire, égale à la période de la variable "a"),
- un rafraîchissement apériodique d'une variable "c" est considéré.

Pour ne pas surcharger la figure suivante, nous avons omis de mettre les trames véhiculant les identifieurs (c'est-à-dire, les trames ID_DAT et ID_RQ).



Pa (resp. Pb) : période de la variable a (resp. b)

Fig. 4.4 — Exemple de comparaison entre les deux solutions

L'exemple illustré par la figure précédente montre bien que :

- Avec la première solution, la période de chaque variable est respectée, mais avec un délai égal, au plus, à période elle-même.
- Avec la deuxième solution, les périodes des variables sont respectées mais avec un délai égal à la période de l'identifieur de demande explicite de la liste de variables configuré pour le pont.
- La variable apériodique "c" n'a pas été transmise, dans la première solution, elle l'a été dans la deuxième solution.

Nous rappelons que pour une variable donnée V avec une période P_V, un arbre indiquant les ponts intermédiaires empruntés doit être construit. Pour assurer maintenant que V est transmise avec la même période P_V, sur les bus où se trouvent ses consommateurs, il existe deux méthodes. Avec la première méthode, chaque pont situé sur l'arbre d'acheminement de V transmet périodiquement la valeur de V en respectant sa périodicité P_V. Ceci implique que la période de la liste d'identifieurs P_L définie pour ce pont doit être égale au PGCD des périodes des variables produites par ce pont. Avec la deuxième méthode,

chaque pont assure qu'une valeur d'une variable résidante dans l'un de ces buffers ne doit pas être écrasée par une nouvelle valeur de la même variable sans que l'ancienne valeur soit transmise. Ceci implique que la période de la liste définie pour chaque pont P_L doit être inférieure ou égale à la plus petite période des variables produites par le même pont (il s'agit d'une condition plus faible que le PGCD de la première méthode). Nous ajoutons maintenant l'identifieur de V aux tables de scrutation des arbitres de bus où se trouvent les consommateurs. Ceci garantit que chaque consommateur final de V reçoit périodiquement la valeur de V sans avoir de perte de valeurs.

Quant au délai total du transfert d'une valeur sur un chemin constitué de N ponts, il est majoré par la somme des périodes de listes définies pour les ponts intermédiaires (P_{L1} + P_{L2}+...P_{LN}), en supposant que le temps de transmission sur le médium est négligeable par rapport à la période définie pour chaque liste.

4.4.2. Le trafic de messagerie

Pour la gestion du trafic de messages, le principe suivant est retenu pour élaborer le fonctionnement d'un pont FIP :

- une table de routage est utilisée pour la mise en correspondance des adresses des messages en point-à-point,
- une table de routage est utilisée pour la mise en correspondance des adresses des messages multipoint,
- à la réception, par le pont, d'un message (il s'agit d'une trame RP_MSG_NoAck), les deux tables de routage précédentes sont consultées pour connaître le (ou les) port(s) de sortie sur lequel (lesquels,) il faut envoyer le message, puis le message est mis dans la (les) file(s) d'attente associée(s) à (aux) sortie(s) qui vient (viennent) d'être déterminée(s).
- Les messages stockés dans les files d'attente des ports de sortie du pont sont acheminés vers leurs destinataires, en utilisant des identifieurs de messages cycliques réservés pour le pont ou bien par des demandes explicites de transfert de messages. Nous montrons dans la figure suivante la modification portant sur la figure 4.4 pour permettre le transfert de messages sans acquittement :

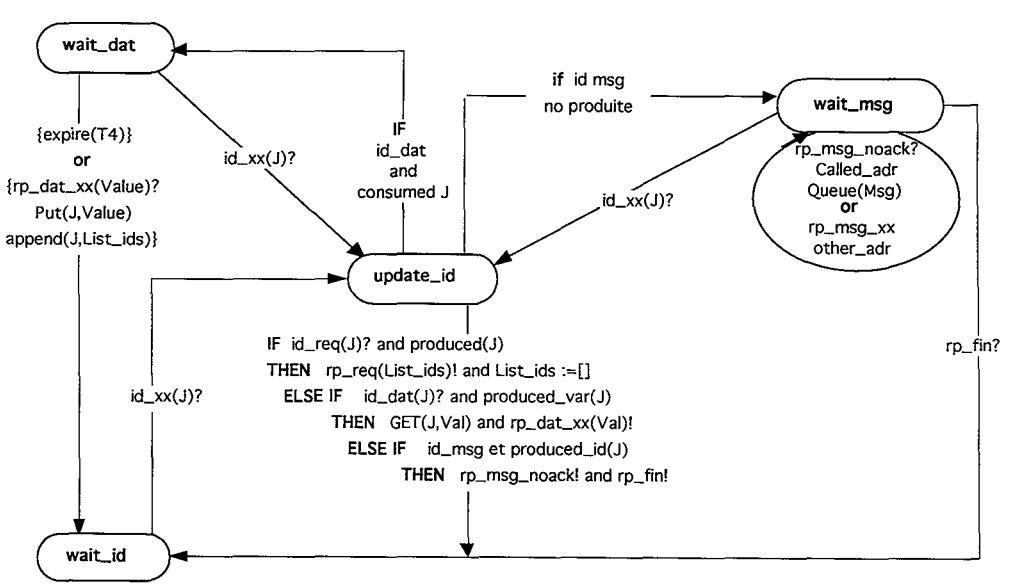


Fig. 4.5 — Machine à états associée à un port d'un pont

Modification :

update_id → wait_id si la trame est ID_MSG dont l'identifieur est produit alors envoyer un mesage (RP_MSG_NoAck!) de la file d'attente associée à cet port, s'il en existe un, et envoyer la trame RP_FIN!,

update_id → wait_msg en réception d'une trame ID_MSG dont l'identifieur n'est pas produit le port se prépare comme un éventuel destinataire,

wait_msg → wait_msg

- soit en réception d'une trame RP_MSG_NoAck(msg) dont le port est destinataire alors stocker le message dans le port de sortie correspondante,
- soit en réception d'une trame RP_MSG_NoAck dont le port n'est pas destinataire ou une trame RP_MSG_Ack alors ignorer la trame,

wait_msg → wait_id en réception d'une trame RP_FIN.

Il faut noter que pour avoir un fonctionnement correct du pont, les points suivant doivent être considérés :

1) Eviter la congestion de messages dans la file d'attente du pont

Il se trouve que le flux de messages n'est pas connu à l'avance dans un bus FIP et que la norme ne fournit aucun moyen pour contrôler ce flux. Il s'y ajoute le fait que nous avons associé une seule file d'attente par port à laquelle sont associés les messages cycliques et apériodiques. C'est pourquoi, nous suggérons un mécanisme de demandes explicites de transfert apériodique de messages. Ce mécanisme est le suivant : en supposant qu'il y ait un compteur de messages résiduels dans la file d'attente, un port du pont, en recevant une trame d'un objet identifié ID_DAT dont il est producteur et le compteur de messages n'est pas nul, il émet RP_DAT_MSG pour que l'AB lui donne la main dans la scrutation apériodique. Le transfert d'un message selon ce mécanisme peut subir un léger retard qui reste négligeable tant que la demande apériodique est satisafaite dans le même cycle élémentaire.

2) Traitement de messages avec acquittement

Nous avons dit au-dessus que le pont ne traite pas les messages acquittés donc des mécanismes pour informer la source pour ne pas envoyer un message avec acquittement à travers les ponts doivent être fournis : une source de message ajoute habituellement le numéro du bus à l'adresse destinataire losqu'il désire envoyer un message. Donc, elle peut naturellement vérifier que le type de message doit être non aquitté lorsque le destinataire ne se trouve pas sur le même bus.

4.5. Les ressources nécessaires pour implanter un pont

La figure suivante permet de résumer les différentes ressources utilisées par un pont FIP fondé sur l'utilisation des listes de variables (c'est-à-dire, fondé sur la solution 2 précédemment présentée).

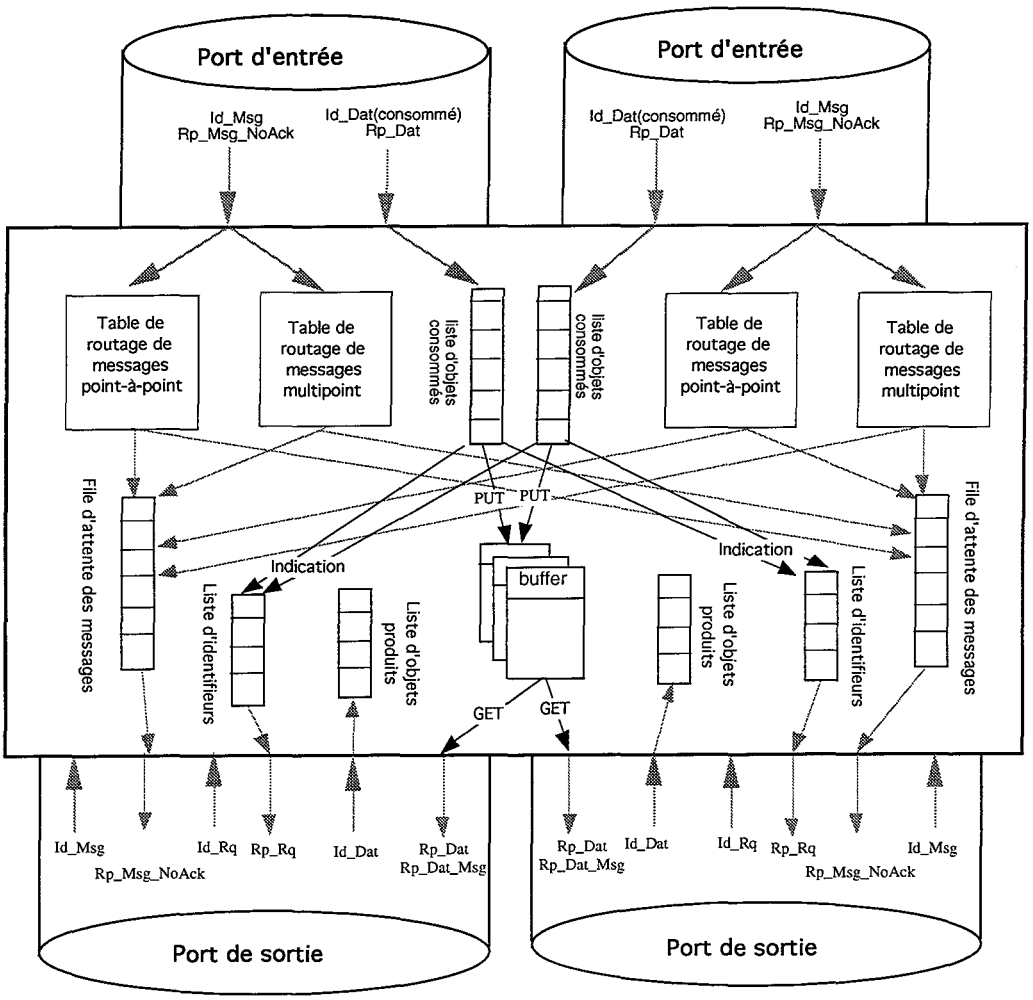


Fig. 4.6 — Structure interne d'un pont FIP

Il faut bien noter que les services de messagerie acquittée ainsi que celui de demande explicite de mise à jour de liste d'identifieurs ont été restreints au trafic local. Nous justifions notre choix par le fait que la garantie de la communication de bout en bout qui relève normalement de la couche transport et non pas du MAC ne peut dans aucun cas être implantée dans un pont qui effectue le transfert par étapes (*stepwise bridge ou store and forward bridge*) comme le nôtre où nous n'établissons pas de connexion de bout en bout. Aucune synchronisation entre les Arbitres de Bus ne sera envisageable et de même nous ne traitons pas l'acquittement du pont.

Pour le cas d'une demande explicite d'une liste d'objets, cette demande ne sera pas propagée jusqu'aux producteurs initiaux des objets et ce sont les copies des variables demandées qui sont stockées dans les ponts et qui vont remplacer la propagation de la variable. Nous avons choisi cette solution parce qu'elle permet de préserver l'autonomie, l'indépendance et la transparence de chaque bus ainsi que de limiter l'interaction entre les différents bus pour ne pas dégrader la performance globale du réseau.

Par conséquent, les fonctions et les ressources minimales de la couche Liaison, nécessaires au fonctionnement du pont sont celles du trafic identifié, celles de la messagerie, celles des listes d'identifieurs cycliques et enfin celles des demandes explicites de transfert de messages sans acquittement. Ces dernières sont les suivantes :

- un mécanisme permettant au pont d'ignorer les trames invalides. La décision peut dépendre de la séquence de contrôle sur trame (FCS), du champ contrôle ou tout simplement si la trame n'est pas conforme à la structure liaison de données des trames,
- un mécanisme d'analyse du type de trame (champ contrôle),
- une table des identifieurs de variables reconnus en production/consommation par rapport à chaque bus,
- une table des identifieurs requêtes (messagerie et listes cycliques) utilisés dans chaque port,
- un buffer, partagé par tous les ports concernés, est défini pour chaque identifieur de variable qui doit être soit en émission, soit en réception. Ce buffer doit permettre un double accès (écriture/lecture) à savoir qu'un même identifieur ne peut pas être produit et consommé sur le même port,
- une table de routage pour les messages point à point. En consultant cette table, un port d'entrée d'un pont peut décider de réexpédier la trame sur le port de sortie ou de l'ignorer. Puisque le routage se fait à la base du numéro de bus, la taille de cette table est proportionnelle avec le nombre de bus,
- une table de routage pour les messages multipoints. En consultant cette table un port d'entrée peut décider de réexpédier (copier) le message sur le(s) port(s) de sortie(s) ou de l'ignorer. La décision est faite selon l'adresse source et l'adresse de destination (adresse de groupe),
- un identifieur de liste de variables, au moins, est défini par port du pont qui lui permet de faire circuler les demandes de listes de variables cycliques,
- une file d'attente d'émission par port lui permettant d'envoyer les messages stockés. L'envoi de messages peuvent s'effectuer soit dans la scrutation cyclique : le pont attend la diffusion par l'AB d'une trame de messagerie contenant l'identifieur associé au pont, soit dans la scrutation apériodique effectuée par une demande explicite de transfert de message sur un identifieur de variable produite par le pont,
- un indicateur de nombre de messages résiduels dans la file d'attente (si indicateur est égal à 0, il signifie une file vide, s'il est égal à *max*, il signifie une file pleine).

4.6. Conclusion

Les résultats obtenus jusqu'à présent nous ont permis de résoudre les difficultés inhérent à la spécificité du réseau FIP et de proposer des solutions permettant l'interconnexion de réseaux FIP par des ponts.

Une spécification formelle d'un bus FIP connecté à un pont a fait l'objet d'un travail complémentaire qui nous a permis de valider nos propositions et sera présenté dans le chapitre suivant.

Chapitre 5

Spécification formelle d'un réseau FIP connecté à un pont

Ce chapitre a pour objet la spécification formelle et la validation qualitative des propositions qui ont été faites dans les deux chapitres précédents.

Pour ce faire, nous allons passer en revue les principales techniques de spécification formelle et de vérification. Nous choisissons celles qui s'adaptent bien au domaine des protocoles de communication.

5.1. Introduction

Selon le modèle OSI, un système de communication est constitué des services fournis localement par une couche N à une couche supérieure (N+1) et des protocoles d'échange de données entre deux entités distantes du même niveau. La description de ces services et protocoles est faite traditionnellement par des langages naturels qui ont l'inconvénient d'inconsistance, d'incomplétude, d'avoir des ambiguïtés et enfin de ne pas fournir un moyen pour la vérification

Une des solutions prometteuses pour faire face à ce besoin est d'adopter les méthodes formelles qui, en donnant une représentation abstraite du système, ont l'aptitude de rendre la spécification plus concise et moins ambiguë. Elles peuvent fournir un moyen pour le raisonnement. Or, ces techniques peuvent produire un système d'une très grande intégrité comportant un nombre réduit d'erreurs et ayant un fonctionnement correct vis-à-vis de sa spécification.

Mais pour pouvoir le plus profiter des méthodes formelles, il faudra bien choisir un compromis entre la puissance d'expression et les niveaux d'abstraction qui peuvent être supportés. Une puissance descriptive d'un langage réduit la taille de la spécification mais en revanche peut rendre le raisonnement plus délicat. Il convient aussi de bien choisir la méthode formelle la plus adaptée au type de système supporté et aux types de propriétés qui peuvent être vérifiées. Il est enfin de grande importance d'identifier les phases dans

lesquelles les méthodes formelles seront exploitées; une utilisation intégrale par exemple implique selon [Bowen 94b] la spécification du système considéré, démontrer que les propriétés attendues sont satisfaites et les propriétés non désirées sont évitées, utiliser ainsi le raffinement de spécification pour concrétiser les abstractions au fur et à mesure jusqu'à la représentation finale par un code exécutable.

Selon [Bowen 94a] les méthodes formelles sont recommandées dans tout système où on évoque un fonctionnement correct. Cela a incité les organisations internationales de normalisation comme l'ITU[1] (ex-CCITT) et l'ISO à reconnaître les techniques de description formelles (FDT[2]) et à définir Estelle [ISO3, ISO4], Lotos [ISO5] et SDL[3] [CCITT 84, ITU 93] comme des normes internationales pour permettre une spécification non ambiguë, claire, concise, ayant un support pour l'analyse et comportant une base pour générer des implantations [Blyth 87]. [Bowen 93] recense les applications industrielles faisant appel aux méthodes formelles comme (aviation, chemin de fer, réacteurs nucléaires, systèmes médicaux). L'utilisation des méthodes formelles dans les applications industrielles est une exception plutôt qu'une norme.

En ce qui concerne les réseaux de Petri, ils sont de plus en plus utilisés dans les systèmes répartis en général et dans le domaine des protocoles de communication en particulier. Nous nous référons à [Diaz 86] qui présente un état de l'art contenant les principaux protocoles modélisés ainsi que les types de RdP les plus souvent utilisés, à savoir : RdP Temporisé et RdP Prédicat-Transition.

Nous proposons dans ce chapitre une application des RdP à un nouveau domaine, à savoir l'interconnexion de réseaux. Il s'agit d'une spécification formelle d'un pont connecté à un réseau FIP par les RdP Étiquetés Prédicat-Transition [Saba 95c]. Notre objectif consiste donc à :

- spécifier formellement le comportement d'un pont en parallèle avec le comportement d'un segment comportant : un producteur, un consommateur, un Arbitre de Bus et le médium. Cette spécification considère la structure de 3 M_SAP et l'identification LLC/MAC qui ont fait l'objet du chapitre 3.
- vérifier le comportement interne de chaque entité ainsi que la cohérence de la composition parallèle du pont avec le segment pour valider la bonne opération de ce dernier.

Nous organisons le reste de ce chapitre comme suit : un rappel des RdP Étiquetés Prédicat/Transition (EPrT) est donné dans la deuxième section. La troisième section est consacrée à la modélisation du pont ainsi que d'un segment FIP en utilisant l'outil EVAL [Verilog 91, Lloret 92] qui nous a permis d'obtenir les résultats présentés dans la quatrième section.

5.2. Réseaux de Petri Étiquetés Prédicat-Transition

La capacité à décrire la concurrence, le parallélisme et le partage de ressources de manière naturelle ainsi que les bases mathématiques très rigoureuses sur lesquelles sont fondés les réseaux de Petri font que ces derniers ont gagné une confiance acceptable pour modéliser les systèmes répartis en général.

1 International Telecommunication Union,
2 Formal Description Techniques,
3 Specification and Description Language,

Malheureusement, les RdP de bas niveau comme les réseaux élémentaires [Thiagarajan 86], Place-Transition [Reisig 86] et condition/événement s'avèrent insuffisants pour traiter des systèmes réels complexes comme les protocoles de communication. Dans ce type de systèmes, des besoins de structuration, de généricité et d'amélioration de la capacité descriptive en préservant, en même temps, la puissance analytique se sont manifestés.

Ces nouveaux besoins ont donné naissance aux RdP de haut-niveau [Genrich 86, Jensen 86, Bruno 92] dans lesquels, une paramétrisation des transitions ou des places qui sont dorénavant munies de structures de données est devenue possible.

Pour notre cas, nous avons choisi les RdP EPrT (Étiquetés Prédicat-Transition) qui, grâce aux techniques d'Étiquetage de [Lloret 90] fournissent le moyen nécessaire pour décrire la communication avec l'environnement en rendant les contraintes de synchronisation plus explicites. Et grâce aux prédicats qui sont associées aux transitions, ils permettent, d'après [Genrich 86], de construire une spécification hiérarchique, générique, plus concise et fournissent, par les moyens de la logique du premier ordre, la sémantique nécessaire pour définir et manipuler les données et les fonctions. Cela permet, par conséquent, d'établir une spécification indépendante de l'implémentation et peut servir comme outil de modélisation formelle d'un intérêt général [Heuser 92].

L'étiquetage permet donc la synchronisation atomique, sur un point d'interaction, entre le système et son environnement. L'occurrence d'un événement sur un point d'interaction est représentée par une interaction qui porte éventuellement des valeurs.

Dans un RdP PrT, une place contient des classes d'objets de données alors qu'une instance dénote un élément individuel de la classe. Le marquage d'une place est défini comme la présence ou l'absence de ces objets. On parle ici d'instance d'une transition correspondante à chaque substitution possible de ses variables.

Dans ce sens, les RdP EPrT sont des paramétrisations, à deux niveaux, des réseaux étiquetés. Dans le premier niveau, nous gagnons la généricité obtenue en colorant les jetons et en paramétrant les transitions par des variables qui permettent d'activer plusieurs instances d'une transition. Le deuxième niveau de paramétrisation s'appuie sur la possibilité d'avoir plusieurs instances d'un réseau de Petri dépendant de la configuration.

L'élément clé de cette représentation s'appuie sur la capacité de déplier (ou décomposer) les représentations PrT en RdP élémentaires sans perdre des propriétés. Ce qui préserve toutes les techniques de vérification caractérisant les RdP comme les invariants de places ou de transitions, l'absence d'interblocage, la vivacité et la réinitialisabilité.

Il convient de noter que les extensions comme la possibilité d'ajouter des poids sur les arcs et la possibilité d'avoir des places munies des capacités variées ont un grand intérêt pour l'étape de la spécification.

5.3. Spécification formelle et validation

Nous retrouvons dans [Fip 88] une spécification formelle de la couche Liaison de Données FIP par des réseaux de Petri Étiquetés. Mais, le besoin de séparation des services de MAC de ceux de LLC, de structuration de ces deux sous couches en termes de SAPs et de CEPi ainsi que de spécification d'un pont qui opère au niveau MAC, nous ont amenés à effectuer ce travail. Celui-ci consiste à :

- identifier les services rendus par la sous-couche MAC

- définir les M_SAPs
- identifier une sous-couche MAC dans l'Arbitre de Bus de façon à autoriser la demande de mise à jour des identifieurs issus des couches supérieures ainsi que le comportement de l'AB pendant la scrutation dynamique,
- spécifier le comportement interne du pont ainsi que les ressources demandées par ce dernier,
- valider le comportement du pont en interaction avec un segment

Pour atteindre notre objectif, nous étions amenés à utiliser l'outil EVAL [Verilog 90], outil orienté RdP, qui permet la conception d'une architecture et son analyse en même temps.

Pour ce faire, nous présentons en premier temps la conception hiérarchique du système (segment et pont) en modules communicants, nous définissons le comportement attendu de chaque module dans un deuxième temps. Et enfin, la vérification de propriétés sera la dernière étape pour valider notre proposition.

5.3.1. La conception de l'architecture

La notion d'architecture est connue dans le modèle OSI [Zimmerman 80] qui recommande la décomposition d'un système en plusieurs sous-systèmes séparés et coopérants. Dans ce contexte, une entité de protocole est construite en plusieurs modules et canaux interconnectés. Il recommande également la répartition des fonctions en couches de façon à rassembler les fonctions homologues dans la même couche.

La conception d'une architecture, selon EVAL, consiste à décomposer le système global en sous-systèmes et ainsi de suite jusqu'à obtenir des unités élémentaires dont les comportements sont bien maîtrisés. Elle exprime ainsi les éventuelles communications entre les différents modules (sous-systèmes) qui peuvent être soit dans un sens vertical (père-fils) soit dans un sens horizontal (fils-fils) d'une manière inspirée de la norme internationale des techniques de description formelles ESTELLE [ISO3], [Amer 89], [Courtiat 87].

Concernant le type de communication, il est devenu évident que les mécanismes de communication asynchrone (par files d'attente FIFO) sont insuffisants pour représenter fidèlement une architecture de communication. C'est pourquoi, nous utilisons en plus le mécanisme de communication synchrone, ou par rendez-vous, jugé indispensable dans [Courtiat 89] pour modéliser les protocoles de communication

La figure suivante illustre l'architecture emboîtée des modules ainsi que les éventuelles communications telle qu'elle était construite au fur et à mesure de l'avancement de spécification et de vérification.

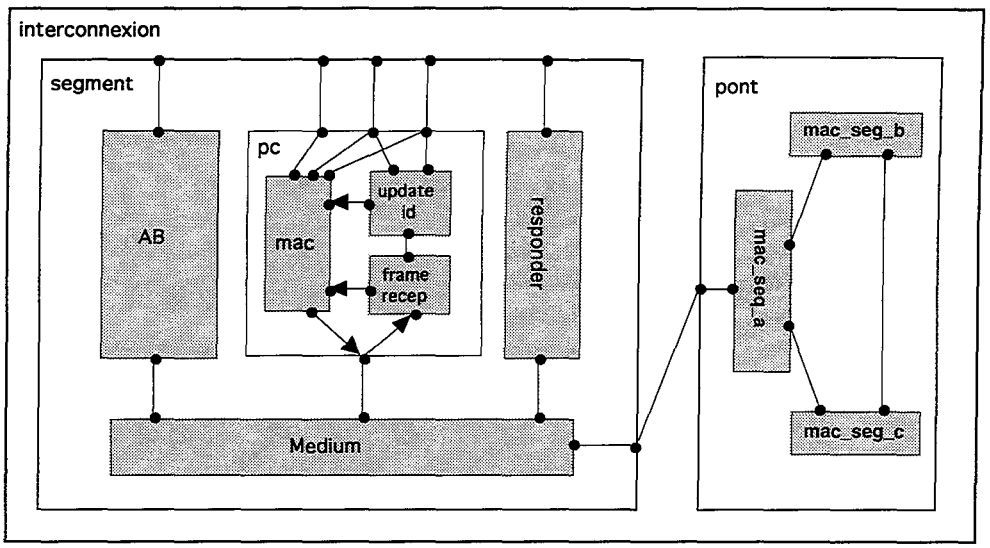


Fig. 5.1 — Hiérarchie générale d'un segment FIP connecté à un pont

N.B., les canaux bilatéraux de la figure sont représentés par des doubles canaux unilatéraux dans la spécification réelle car EVAL ne permet pas de définir des canaux bilatéraux.

Il convient de noter qu'un comportement a été associé avec chaque module en gris qui représente le dernier niveau d'abstraction. Les comportements relatifs aux autres modules, comme le module PC par exemple, peuvent être obtenus soit par composition parallèle de ses sous-modules, à savoir, `mac`, `update_id` et `frame_reception`, soit par une spécification explicite indépendamment de ses sous-modules. Cela permet la portabilité du module ainsi que la possibilité de vérification de conformité du module à deux niveaux d'abstraction.

5.3.1.1. Le module segment

Ce module modélise le comportement interne d'un segment FIP, vu en terme de points d'interaction avec le pont et avec la sous-couche LLC. Il est composé de quatre modules :

1) l'Arbitre de Bus (AB) tel que défini dans §3.3.2 modélise l'évolution interne de la partie MAC en fonction des demandes de service LLC concernant la mise en circulation des identifieurs et la collection des demandes explicites dans la partie périodique pour pouvoir les émettre dans la partie apériodique.
2) Le module Producteur/Consommateur (PC) représente une entité terminale générique qui se reconnaît en production ou en consommation selon la valeur de l'identifieur diffusé sur le médium. Elle comporte trois modules : un module `frame_recep` reçoit les trames logiques (sous formes de suites des paramètres) et envoie les identifieurs au module `update_id` et les autres au module `mac`. Le module `update_id` gère les demandes explicites ainsi que la reconnaissance des identifieurs. Lorsqu'il reconnaît un identifieur en production, il vérifie l'existence d'une demande explicite associé à cet identifieur et envoie le résultat (identifieur et réponse) au module `mac`. Il modélise aussi l'interface asynchrone avec le LLC sur laquelle les demandes explicites surviennent aux points d'accès aux services. Le module `mac` est responsable de la gestion du champ contrôle et d'allouer le médium à une entité LLC lorsqu'il est opérationnel.

3) Le répondeur `responder` assure l'opération d'un consommateur lorsque le module `PC` est producteur ou bien le destinataire d'un message pour envoyer l'acquittement lorsqu'il s'agit d'un message avec acquittement.
4) Le module `medium` fait abstraction de la couche physique ainsi que de l'encodage des données et la détection d'erreurs. Il sert également à diffuser les trames vers tous les modules qui lui sont rattachés.

5.3.1.2. Le module *pont*

Nous avons choisi de présenter un module `pont` avec trois ports pour généraliser les interactions à l'intérieur du pont de façon à mettre en correspondance le port d'entrée avec le port de sortie (lorsqu'on dispose d'un pont à deux ports, les trames entrant sur le premier port doit sortir sur le deuxième). Dans ce cas, un automate, ou un processeur, associé à un port d'entrée doit avoir à sa disposition la possibilité de copier un message multipoint sur les deux ports de sortie ou tout simplement aiguiller un message bipoint sur un seul port.

5.3.2. Le modèle de comportement de l'architecture

Une variété des techniques de modélisation de comportement se trouve dans la littérature. Nous citons quelques unes des plus connues :

- les machines à états finis [Kohavi 70],
- les réseaux de Petri [Petri 62],
- CCS[4] [Milner 80],
- CSP[5] [Hoare 85],
- la logique temporelle [Rescher 71].

Malheureusement, ces techniques se sont avérées insuffisantes, au moins dans leurs formes initiales, pour traiter des applications réelles. Il existe des extensions de ces techniques qui tendent à exprimer et vérifier les contraintes de temps, la communication et le parallélisme ou pour fournir une spécification générique et structurelle enrichie par des structures de données. Cela a donné naissance à : statechart [Harel 87], CRSM[6] [Shaw 92], les machines à états finis étendues, le HMS[7] [Gabrielian 91], les réseaux de Petri haut niveau, le RTTL[8] [Ostroff 89], et l'introduction du temps dans CSP [Lee 87].

De telles techniques, pour être efficacement appliquées au domaine de protocoles de communication, doivent fournir, selon [Blyth 87], un moyen pour :

- émettre/recevoir des messages
- décrire des structures de données (e.g., adresse, SDU, PDU, nom)
- exprimer des actions non observables
- exprimer la synchronisation pour définir le modèle de coopération entre les modules (synchrone/asynchrone)
- raffiner le comportement d'un système en fonction de ses sous-systèmes
- manipuler les modules (créer, détruire) ou des canaux (connecter, libérer)

4 Calculus of Communicating Systems ,
5 Communicating Sequential Processes ,
6 Communicating Real-time State Machines,
7 Hierarchical Multi-state Machine,
8 Real Time Temporal Logic,

- exprimer le non-déterminisme important pour rendre la spécification indépendante de l'implémentation mais qui peut être dangereux dans certains systèmes réactifs ou temps réel
- imposer des délais sur les actions.

A l'exception de la manipulation de modules et des canaux, l'outil EVAL, présenté dans le paragraphe suivant, permet, entre autres, de définir et de traiter tous les points précédents. En ce qui concerne l'analyse de faisabilité de délais, le respect des contraintes temporelles et évaluer la performance, des modèles de RdP stochastiques ou temporisés peuvent être utilisés.

5.3.2.1. Introduction à l'outil EVAL

En ce qui concerne le comportement, l'outil EVAL permet d'associer plusieurs modèles de comportement à un module : soit une simple machine à états, soit des RdP Prédicat-Transition Étiquetés, soit des RdP temporisés ou stochastiques. Nous avons choisi le modèle EPrT parce qu'il nous semble le plus adéquat pour définir des modules génériques et paramétrables d'une part, et pour la possibilité de colorer les jetons qui facilite la spécification d'autre part.

Nous rappelons que EVAL a une partie syntaxique qui est un sous-ensemble du langage de description formelle ESTELLE qui permet à la fois la description incrémentale et la spécification structurelle. La description incrémentale fournit une possibilité de raffinement de la spécification pour avoir une vue sur chaque niveau d'abstraction ainsi que pour profiter des notions de généricité et d'héritage. La spécification structurelle, quant à elle, améliore la lisibilité du système et la portabilité de ses composantes et permet la vérification ascendante.

Le corps d'un module élémentaire (dernier niveau d'abstraction, comme le module `mac` par exemple) selon le modèle RdP EPrT comporte les quatre parties suivantes :

1) Nous déclarons dans la première partie VAR les variables globales à tous les états d'un module. La notion de variable globale devient importante dans les protocoles de communication pour pouvoir mémoriser des valeurs de compteurs ou les derniers messages reçus en général ainsi que pour mémoriser le dernier identifieur véhiculé, le champ contrôle ou les files d'attente et les buffers pour le cas de FIP.
2) Dans une deuxième partie, nous initialisons les variables globales et nous configurons le marquage initial par INITIALIZE TO. Cette partie peut être définie dans un fichier à part pour obtenir un comportement réutilisable indépendamment de la configuration initiale (e.g., toutes les entités terminales ont le même comportement mais les ensembles de variables produites ou consommées ne sont pas les mêmes).
3) Dans la troisième partie TRANS, nous spécifions l'évolution des RdP selon les règles de tir de transitions propres aux réseaux PrT. Un état, dans un modèle PrT, est un ensemble de places marquées alors qu'une place, qui est de même un prédicat, sera marquée lorsque le prédicat est vrai. L'ordre des clauses dans une transition est :

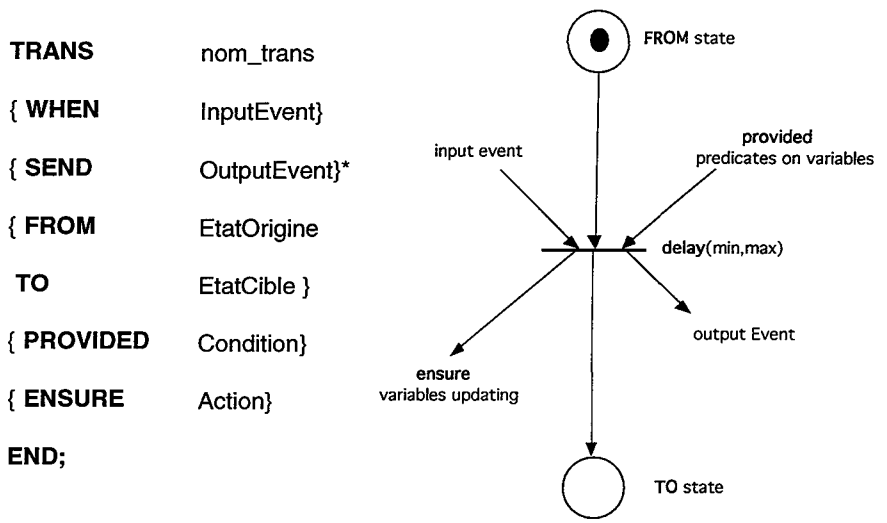


Fig. 5.2 — Spécification d'une transition selon EVAL

Une transition peut être tirée dans un état lorsque : les prédicats de la clause FROM correspondent aux prédicats de l'état courant, l'évaluation des clauses PROVIDED et ENSURE retourne une valeur vraie et les prédicats de la clause TO ne doivent pas correspondre à des prédicats de l'état courant (sauf si ce prédicat est déjà dans une clause FROM).

4) Dans la quatrième partie, nous pouvons définir un code PROLOG qui définit les faits ou les procédures qui doivent être appelés à l'intérieur des clauses PROVIDED ou ENSURE. Dans ces derniers, des opérateurs booléens, des expressions logiques ainsi que des structures de contrôle (IF THEN ELSE) peuvent être évalués en plus.

5.3.2.2. Comportement d'un segment FIP

1. COMPORTEMENT DU MODULE AB

Le cycle de vie de l'AB passe par une phase périodique et une phase apériodique. Pendant la première phase (dénotée par `static` dans le code suivant), l'AB reçoit les demandes d'accès au médium en provenance du LLC (`from_llc.rq`); envoie les identifieurs (`to_medium.trame_id`); confirme l'état de déroulement de l'émission pour indiquer au LLC qu'il est prêt pour l'identifieur suivant. La sous-couche LLC doit inclure dans chaque primitive de service le statut (`New_id =1` pour nouvel identifieur) de l'identifieur ainsi que la fin du microcycle (`last_id = 1`).

```
trans   id_data_send_st
when    from_llc.rq(Code, No_id, New_id, Last_id)  /* Code = champs contrôle */
send    to_medium.trame_id(Code, No_id)   /* No_id est le numero d'identifieur*/
from    actif, static, t2_on, b_reprise(0) /* New_id = 1 dénote un nouvel id. */
to      wait_data, static, t2_on, b_reprise(0)   /* = 0 dénote id. ordinaire */
provided is_id_dat(Code), bin(New_id), bin(Last_id),
         good_id_no(Code, No_id, New_id, LIST_IDS),
         LAST_ID=0
ensure  IF (New_id=1) THEN
          IF (\+(member(No_id, LIST_IDS))) THEN
            append([No_id], LIST_IDS, LIST_IDS')
          END
        END,
        ID'=No_id,
        LAST_ID'=Last_id
end;
```

N.B., un accent sur une variable globale permet de mettre à jour la valeur de la variable après le franchissement de la transition.

La clause `from` indique que l'AB se trouve dans l'état `actif`, la phase de scrutation est `statique`, le temporisateur `t2` est armé et le buffer de reprise `b_reprise` est vide. Dans la clause `provided`, nous vérifions que le champ de contrôle correspond bien à un identifieur de données, les valeurs de `New_id` et `Last_id` sont binaires, la combinaison de l'ensemble de `code`, `No_id`, `New_id` et la `List_IDS` produits, est valide. Dans la clause `ensure`, l'AB met à jour sa liste d'identifieurs s'il s'agit d'un nouveau, mémorise le dernier identifieur(`ID'` signifie que la valeur de `ID` après avoir franchi la transition en cours sera affectée par `No_id`) et de même pour la variable globale `LAST_ID`.

Après avoir envoyé l'identifieur, l'AB se trouvera dans l'état d'attente de la réponse qui peut être par exemple la suivante :

```
trans   rp_dat_rq1_msg_st  /* reponse data avec demande explicite */
when    from_medium.trame_rp_dat(Code,Value)
send    to_llc.conf('ok')
from    wait_data, static, t2_on
to      actif, static, f_idrq1(ID), f_idmsg(ID), t2_on
provided is_rp_dat_rq1_msg(Code), capacity_rq1(NBR_F_RQ1),
         capacity_msg(NBR_F_MSG)
ensure  NBR_F_RQ1'=NBR_F_RQ1+1, NBR_F_MSG'=NBR_F_MSG+1, ID'=0
end;
```

Dans ce cas, l'AB reçoit une trame réponse qui contient une demande explicite pour un message et pour une requête urgente (clause `provided is_rp_dat_rq1_msg`). l'AB sauvegarde la valeur de la variable globale `ID` dans chacune des files d'attente `f_idrq1` et `f_idmsg` (clause `TO`) et incrémente les compteurs associés aux files d'attente (clause `ensure`).

Dès que l'AB reçoit le dernier identifieur (`Last_id = 1`), il démarre la phase apériodique de messages. Il teste ainsi la file d'attente associée aux messages, et envoie `id_msg` tant que la file n'est pas vide et le temporisateur `t2` est armé. Lorsque la file de messages devient vide, l'AB exécute la transition suivante qui lui permet de démarrer la partie apériodique de requêtes urgentes :

```
trans   f_msg_vide
from    actif, aper_msg,t2_on, 0*f_idmsg(_)
to      actif, aper_rq1, t2_on
end;
```

On remarque ici que le test à zéro (arc inhibiteur) s'est avéré très pratique pour vérifier qu'une file d'attente est vide. Si par contre le temporisateur `t2` expire, un nouveau cycle élémentaire prendra place en laissant intactes toutes les files d'attente qui sont associées aux messages ainsi qu'aux requêtes. On peut en déduire que :

1- le trafic apériodique peut ne pas être demandé dans la phase périodique précédente,

2- l'AB utilise la fonction `capacity` (voir `provided` dans `trans rp_dat_rq1_msg_st`) pour ignorer une demande explicite au cas ou la file d'attente associée est pleine ce qui n'a pas été défini dans la Norme.

Les parties successives de requêtes seront exécutées d'une façon similaire à l'exception de l'interprétation de la liste des identifieurs comme montré par le code suivant :

```
trans   rp_rq1_list_ap1
when    from_medium.trame_rp_rq(Code, List_id)
from    wait_rq, aper_rq1, t2_on, f_rprq(0)
to      actif, aper_rq1, f_rprq(Id_nbr), t2_on
provided is_rp_rq1(Code), valid(List_id)
ensure   append(List_id, ID_RPRQ, ID_RPRQ'), length(List_id, Id_nbr)
end;
```

Nous supposons ici que l'AB vient de diffuser une trame requête `id_req1` et il se trouve dans l'état d'attente de la liste d'identifieurs associée (`wait_rq` dans la clause `from`), la phase est donc apériodique de requêtes urgentes (`aper_rq1` dans la clause `from`), la file d'attente de reprise (`f_rprq(0)` dans la clause `from`), qui indique le nombre d'identifieurs restant dans la file d'attente de reprise `ID_RPRQ`, doit retourner une file vide avant le franchissement de la transition. Ce compteur indiquera le nombre d'identifieurs demandés après le franchissement (`length` dans clause `ensure`).

Il convient de signaler ici la puissance descriptive des RdP PrT par lesquels nous avons pu passer du champ données de la trame réponse qui contient la liste d'identifieurs (clause `when`) à une suite de transactions `id_dat` et `rp_dat` correspondant à cette liste. Nous ajoutons que la réponse `rp_dat`, lorsqu'elle est scrutée apériodiquement, ne doit pas contenir de demandes explicites.

2. COMPORTEMENT DES MODULES MEDIUM ET FRAME_RECEP

Le module `medium` qui remplace la couche physique est responsable de diffuser les trames qui proviennent soit de l'AB vers les autres modules, soit du producteur. Il synchronise, par conséquent, tous les modules pour assurer le bon déroulement des transactions.

Le module `Frame_reception` assure que les trames identifieurs correctes soient envoyées vers le module de traitement des identifieurs `update_id` ainsi que les trames réponses soient envoyées vers le module `mac`.

3. COMPORTEMENT DU MODULE UPDATE_ID

Ce module assure la reconnaissance des identifieurs `id_dat`, `id_rqi` et `id_msg`, gère les demandes explicites en provenance des M_SAP_rq et M_SAP_msg et communique au

module mac le champs contrôle pour RP_DAT_XX (RP_DAT_RQi, RP_DAT_MSG ou RP_DAT_RQi_MSG).

Le code suivant montre l'invocation de la primitive d'une demande de transfert d'un message sur l'identifieur numéro : Id_no (clause when) :

```
trans   m_access_msg_req
when    from_m_sap_msg2.req(Id_no)        /* demande explicite de message */
send    to_m_sap_msg2.conf(Status)
ensure  IF (valid('msg',Id_no)) THEN
           Status='ok', update_list_id(Id_no, 'msg', LIST_ID, LIST_ID')
         ELSE Status='nok'
        END
end;
```

En fonction de la validité de l'identifieur pour supporter ce type de demande, ce module confirme localement cette demande et met à jour sa table d'identifieurs produits pour indiquer que l'identifieur Id_no doit émettre une demande de message apériodique.

Il faut noter que les demandes explicites sont asynchrones et ce module, en appelant la procédure update_list_id dans la clause ensure, associe à chaque identifieur la valeur du champ contrôle réponse à envoyer au module mac. Par exemple, pour le cas précédant, en réception de id_dat ayant Id_no comme valeur, le module update_id doit notifier au module mac que la réponse sera rp_dat_msg à moins qu'il y ait une demande explicite sur le même identifieur.

4. SPÉCIFICATION DU MODULE MAC

C'est le module principal d'une entité productrice/consommatrice (cf. §3.2.1.2.) qui est en charge d'attribuer à la sous-couche LLC l'accès au médium par l'intermédiaire des primitives de demande d'accès au médium, d'indication, de réponse et de confirmation niveau MAC. Il assure les services suivants :

1- En cas de réception d'un identifieur de données id_dat en provenance du module update_id avec une éventuelle demande explicite chez le producteur, le mac envoie une indication de production ou de consommation sur le M_SAP_Data et active la connexion multipoint entre le producteur et les consommateurs (voir code suivant).

```
trans    m_access_ind_data
send     to_m_sap_data.ind(ID_NO,CONTROL)
from     update_id
to       wait_rsp_data
provided is_id_dat(ID_CODE), CONTROL > 0
end;
```

A ce moment, le LLC chez le producteur doit envoyer le MA.Rsp(Value) qui sera concaténé avec le champ contrôle qui a été déjà constitué dans le module update_id. La confirmation d'envoi est localement générée chez le producteur.

```
trans    m_access_rsp_data
when     from_m_sap_data.rsp(Value)
send     to_medium.trame_rp_dat(CONTROL,Value)
send     to_m_sap_data.conf('Ok')
from     wait_rsp_data
to       wait_id
provided CONTROL > 0
ensure   ID_NO' = 100
end;
```

2- En cas de réception de ID_RQi, le mac chez le producteur seulement, envoie une indication au M_SAP_Req et il attend la réponse avec la liste d'identifieurs pour l'envoyer vers l'Arbitre de Bus.

```
trans    m_access_rsp_rq
when     from_m_sap_req1.rsp(List_id)
send     to_medium.trame_rp_rq(CONTROL,List_id)
from     wait_rsp_rq
to       wait_id
provided is_id_rq(ID_CODE)
ensure   CONTROL' = 40, ID_CODE' = 3, ID_NO' = 100
end;
```

3- En cas de réception de id_msg du module update_id, le mac, en tant que source, envoie une indication sur M_SAP_Msg et attend la réponse. S'il n'est pas source de message, il doit attendre le message en tant que destinataire potentiel. Le code suivant illustre l'envoi de MA.Rsp par la source d'un message sans acquittement.

```
trans    m_access_rsp_msg_noack       /* message sans acquittement */
when     from_m_sap_msg1.rsp(Code_msg,Msg_bdy,Adr_S,Adr_D)
send     to_medium.trame_rp_msg(Code_msg,Msg_bdy,Adr_S,Adr_D)
from     wait_rsp_msg
to       sending_rp_fin
provided is_id_msg(ID_CODE), CONTROL = 1,        /* rp_msg_noack! */
         is_rp_msg_noack(Code_msg),
         set_msg_bdy(Msg_bdy),set_adr_source(Adr_S),set_adr_dest(Adr_D)
end;
```

Nous montrons dans la suite le code exécuté chez le destinataire du message :

```
trans    m_access_ind_msg_noack_dest
when     from_frame_reception.trame_rp_msg(Code,Msg_bdy,Adr_S,Adr_D)
send     to_m_sap_msg1.ind(ID_NO,Msg_bdy,Adr_S,Adr_D)
from     wait_msg
to       wait_end
provided is_my_address(Adr_D), is_rp_msg_noack(Code)
end;
```

5.3.2.3. Comportement du pont

Ce module comporte trois sous modules identiques du point de vue comportement et partagant le même ensemble de buffers pour les variables produites/consommées alors qu'ils communiquent par l'intermédiaire de files d'attente FIFO pour tout ce qui relève de la messagerie. Un module mac_seg_a qui est attaché au segment a assure l'entrée/sortie des variables/messages vers ce segment, l'aiguillage et le filtrage des messages dans les deux autres modules ainsi que la lecture/écriture dans les buffers globaux.

Nous avons dit dans le chapitre 4 que les services rendus par le pont sont restreints au trafic identifié, la demande explicite pour les messages et la messagerie sans acquittement. Donc les fonctions assurées par les deux modules `update_id` et `frame_reception` peuvent être intégrées dans le `mac` du pont. Ce dernier est un sous ensemble du `mac` d'une entité terminale qui comporte les extensions suivantes :

- Définition de buffers contenant les variables partagées par tous les ports d'un pont,
- Puisque EVAL ne permet pas de définir une file d'attente commune entre plusieurs modules, nous aurons donc besoin de 6 files pour assurer l'échange de messages entre les trois modules comme montré dans la figure suivante :

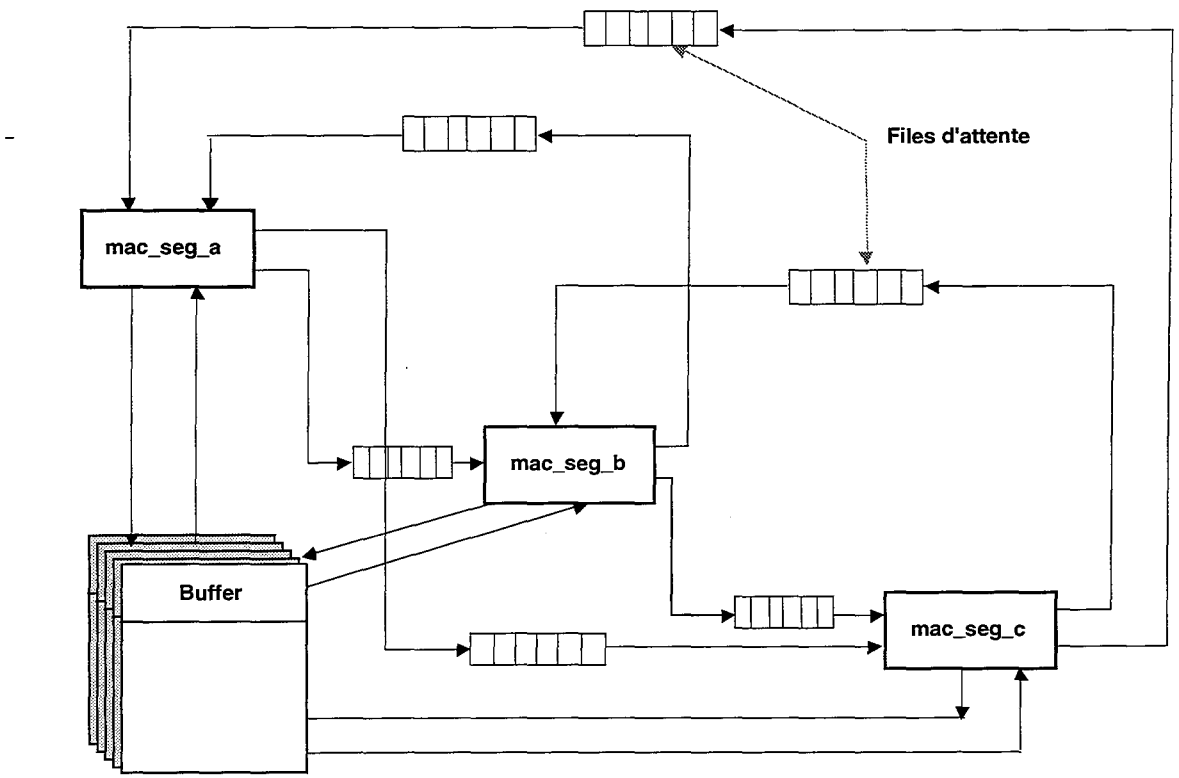


Fig. 5.3 — Les ressources utilisées dans un pont à trois ports

Il faut ajouter à chaque module (port) une file d'attente qui fusionne les deux files d'entrée en une seule pour respecter l'ordre d'arrivée (en fait, une implémentation est possible en associant deux files d'attente entrée/sortie à chaque port).

Nous distinguons deux comportements possibles du module qui gère l'accès à un port (celui du `segment_a` par exemple) : le premier concerne le trafic identifié pour lequel des buffers partagés sont à sa disposition de manière à ne pas avoir deux modules qui peuvent écrire dans le même buffer. En revanche, un module au moins doit lire ce buffer (sinon, on n'a pas besoin du buffer) et transmettre la valeur sur le segment qui lui est associé. Le deuxième comportement qui concerne la messagerie non acquittée consiste :

- en réception d'un message sur le port du segment "a" qui concerne l'un des deux autres ports au moins, il dépose ce message dans la file d'attente associée à ce port pour une éventuelle retransmission sur le port de sortie,

- ce port de sortie, en recevant une trame identifieur, si ce port est producteur, il enlève un message de sa file d'attente de sortie (s'il en existe un) et il l'envoie dans une trame réponse. Il envoie ensuite la trame RP_FIN.

Nous rappelons ici qu'une entité terminale peut avoir plusieurs files d'attente d'émission et chacune est associée à un identifieur cyclique. Nous n'avons pas envisagé cette configuration dans le pont pour assurer, d'une part, une équité entre les messages entrant (FIFO) et parce que nous ne connaissons pas auparavant lesquels parmi les identifieurs cycliques vont porter des messages à travers le pont d'autre part. Le code suivant est utilisé dans le module `mac_seg_a` (rattaché au segment "a") pour traiter la réception d'un identifieur de donnée produite dans le pont :

```
trans    rec_id_dat_produit(No_id)
when     from_seg_a.id_xx(Code_id_dat, No_id)
send     to_seg_a.frame(Code_rp_dat,Value)
from     wait_id,mac_seg_b.buffer(No_id,Value)    /* id_xx */
to       wait_id,mac_seg_b.buffer(No_id,Value)
provided is_id_dat(Code_id_dat),
          produced_var(No_id)
ensure
   IF (NON_REQ_MSG=0) THEN
      set_code_rp_dat(Code_rp_dat),NON_REQ_MSG'=0
   ELSE
       (NON_REQ_MSG'= NON_REQ_MSG-1, set_code_rp_dat_msg(Code_rp_dat))
   END,
   ID_NO'=100
end;
```

A la réception d'une trame identifieur de données `id_dat` (`Code_id_dat` dans la clause `when`) dont le port est producteur (clause `provided`), ce port envoie (clause `send`) une trame réponse (clause `ensure`, s'il existe un message dans la file alors envoyer `rp_dat_msg` sinon envoyer `rp_dat`) contenant la valeur comprise dans le buffer associé à l'identifieur (unification de la clause `from`).

Le moyen par lequel un pont acquiert l'accès au médium pour envoyer un message peut être soit dans la phase périodique de scrutation (l'identifieur cyclique est alloué au pont) ou suite à une demande explicite `rp_dat_msg` (le pont doit envoyer une trame `rp_dat_msg` lorsqu'il y a un message en attente au moins dans la file pour lequel le pont n'a pas encore envoyé une demande explicite antérieure). L'envoi d'un message par un port est effectué selon le code suivant :

```
trans   rec_id_msg_produit
when    from_seg_a.id_xx(Code_id_xx, No_id) réception trame identifieur de données
send    to_seg_a.frame(Code,S,D,Bdy)        émet message contenu dans la file: queue
from    wait_id,queue(Msg)                      enlève le message de la place queue
to      send_rp_fin,queue(NewMsg)la place queue contiendra NewMsg défini dans ensure
provided is_id_msg(Code_id_xx),           L'identifieur correspond aux messageries
         produced_msg(No_id)            L'identifieur(No_id) est produit par le port
ensure
    set_code_rp_msg_noack(Code),        Champ contrôle = trame message non acquitté
    dequeue_msg(Msg,NewMsg,S,D,Bdy),     Enlever un message de la file d'attente
       IF (QUEUE_L>0) THEN                          Tester la longueur de la file
            QUEUE_L' = QUEUE_L-1,                       Diminuer cette longueur
            IF (NON_REQ_MSG>0) THEN                   Pour indiquer le nombre de
               NON_REQ_MSG' = NON_REQ_MSG-1        demandes explicites à générer
       END
END
end;
```

A la réception d'une trame identifieur de message produit par le pont (clause `when` et `provided`), il enlève un message de sa file d'attente de sortie (clause `send, ensure` et `to`). Nous avons modélisé la file d'attente (queue) par une place et les messages par des jetons.

5.3.3. Validation du système

5.3.3.1. Techniques de vérification et de validation

Un système est valide si toutes les techniques de vérification qualitative et d'analyse quantitative sur la spécification formelle n'ont révélé aucune lacune et que l'implémentation est possible.

La vérification d'un protocole de communication est la démonstration qu'une spécification formelle est *correcte, complète et consistante* [Sajkowski 85].

- une spécification d'un protocole est *correcte* lorsqu'elle ne contient plus des erreurs fonctionnelles et que les services attendus seront éventuellement satisfaits,
- une spécification d'un protocole est *complète* lorsqu'elle considère tous les événements possibles et traite toutes les options remplies par les services,
- La *consistance* d'une spécification formelle signifie la conformité entre chaque niveau d'abstraction.

La base sur laquelle s'établit les critères précédents est la notion de propriété. Une propriété d'un protocole peut être *générale* dans le sens où n'importe quel protocole doit satisfaire cette propriété ou *spécifique* lorsqu'elle dépend du protocole traité. Nous citons parmi les propriétés générales l'absence d'interblocage, la vivacité, la fatalité, l'absence de sur-spécification et l'absence de réception non spécifiée. Alors qu'une propriété spécifique doit être définie en fonction du protocole. Par exemple, vérifier, pour le bus FIP, que l'accès au bus est exclusif à une seule entité à la fois ou vérifier qu'un identifieur portant une demande explicite sera éventuellement émis par l'AB dans une des phases apériodiques.

Pour ce faire, le modèle du comportement doit fournir une base formelle très rigoureuse à des fins de vérification. Celle-ci peut être effectuée de plusieurs façons :

- *l'analyse d'accessibilité* [Vuong 87] tend à générer exhaustivement tous les états globaux accessibles pour former un arbre d'accessibilité. Cette technique peut vérifier les

propriétés de *boundedness* pour les ressources, absence d'ambiguité d'états, sur-spécification, complétude, interblocage, reprise sur erreurs et la terminaison. En revanche, elle a l'inconvénient de l'explosion combinatoire de l'espace d'états.

- *L'analyse par des invariants* est utilisée souvent dans les réseaux de Petri (invariants des places ou des transitions). Cette technique sert pratiquement à vérifier les propriétés spécifiques, et peut servir également pour les propriétés de vivacité, d'absence d'interblocage, *boundedness*, reprise sur erreurs et l'équité.
- *La vérification algébrique* portant sur des expressions ou des grammaires par les moyens de la réduction, d'équivalence et de la restriction. Cette technique a été définie dans CCS de [Milner 80] et a été utilisée dans plusieurs outils algébriques comme AUTO [Lecompte 87], [Simone 89] et de même généralisée sur les graphes d'accessibilité comme pour EVAL. Les propriétés qui peuvent être vérifiées en utilisant cette technique sont le *boundedness* , la complétude, l'interblocage, le *livelock* (boucle non productive), la reprise sur erreurs, la sûreté, la vivacité et la conformité du service rendu.
- *La vérification par évaluation de théorèmes sur un modèle mathématique* comme le modèle de processus communicants d'Arnold-Nivat [Arnold 82] sur lequel est basé l'outil MEC [Crubille 89]. La vérification est faite en évaluant des fonctions, définies par l'utilisateur, sur le graphe global d'accessibilité.
- *La projection* d'un protocole selon une fonction donnée ou un ensemble d'actions peut réduire la taille du protocole en préservant en même temps quelques propriétés comme la vivacité, la sûreté et l'interblocage. [Juanole 90] a utilisé cette technique pour vérifier des propriétés spécifiques à l'élément du service application CCR[9] de l'ISO.
- *La logique temporelle* [Rescher 71] qui peut, par des moyens d'inférence logique ou par des opérateurs additionnels comme *henceforth* et *eventually*, raisonner sur des séquencements des éventuels chemins. Les propriétés de vivacité, de sûreté, la conformité à la description, l'interblocage, le *livelock* et l'équité peuvent être vérifiées. [Schwabe 85] a utilisé la logique temporelle pour spécifier un protocole d'accès à un réseau local et pour vérifier ainsi des propriétés de progression, de consistance et le bon fonctionnement.
- *La simulation* du comportement d'un protocole accroît la confiance sur la spécification mais elle ne la garantit pas. Les propriétés d'absence d'interblocage et la conformité peuvent être vérifiées durant cette phase. On peut se référer à [Groz 85] pour un exemple de l'utilité de la simulation aléatoire ou séquentielle pour la validation du système Galaxie du CNET.
- *L'exécution de la spécification* est possible dans certains cas où un interpréteur ou translateur sont fournis. Cette technique peut servir non seulement à vérifier les propriétés de consistance, de sûreté et de vivacité mais aussi à générer des prototypes rapides.
- *La vérification des propriétés temporelles* qui peuvent concerner la disponibilité, la validité, les délais, la temporisation, la date au plus tôt et la date au plus tard sont d'une grande importance surtout pour les protocoles (ou services) soumis à des contraintes temporelles strictes. Nous retrouvons ce type de raisonnement dans la logique temporelle [Ostroff 89], Timed CSP, les réseaux de Petri Temporels (*timed Petri Nets*) [Ramachandani 74] et temporisés TPN (*Time Petri Nets*) [Merlin 76], les CRSM [Shaw 92] et les HMS [Gabrielian 91].

9 Commitment Concurrency and Recovery

A partir de la puissance descriptive, du nombre et du degré d'importance des propriétés qui peuvent être vérifiées ainsi que de l'adaptation des techniques au type de protocole examiné, la technique appropriée doit être choisie. Nous citons dans la suite les techniques de vérification qui nous ont permis de valider (qualitativement) notre proposition

5.3.3.2. Validation d'un pont connecté à un segment FIP

Nous montrons dans le reste de ce chapitre les principales techniques de vérification qui nous ont aidés à valider notre spécification formelle.

La vérification porte aussi bien sur les comportements des modules élémentaires que celui de la composition parallèle à tous les niveaux d'abstraction, à savoir : composer les modules `mac`, `update_id` et `frame_reception` pour former une entité terminale `PC` qui sera composée à son tour avec l'AB et le medium pour former un segment. Les trois modules du pont seront composés également pour donner une vue abstraite du comportement du pont, qui sera composée avec un segment pour former un système interconnectant un pont à un segment. Les techniques de vérification qui ont été utilisées sont celles de la simulation, de la vérification de propriétés spécifiques et générales ainsi que l'évaluation des formules de la logique temporelle combinatoire `ctl` sur le graphe de marquage. Et enfin l'utilisation de la projection pour vérifier la conformité à différents niveaux d'abstraction ou pour vérifier le comportement par rapport à une fonction donnée ou à un ensemble d'actions.

1) VÉRIFICATION DU MODULE AB

La distribution centralisée du droit de parole rend la tâche de l'AB la plus critique dans un segment FIP. Nous ajoutons que l'AB en tant que destinataire de toute trame peut avoir un aperçu général de l'état global du réseau (un producteur qui ne diffuse pas sa valeur dans le délai est détecté par un temporisateur à l'intérieur de l'AB, les trames concernant les messages acquittés doivent passer par l'AB). Il faut noter enfin qu'en spécifiant le comportement de l'AB pendant la scrutation statique et dynamique, indépendamment de l'implémentation, nous aide à (i) vérifier des propriétés relatives aux demandes explicites, (ii) vérifier la saturation des ressources associées aux demandes explicites (iii) vérifier le déroulement des transactions pendant la fenêtre apériodique qui n'est pas toujours identique à celui d'une fenêtre périodique (e.g., on ne peut pas envoyer une demande explicite sur une trame réponse `rp_dat` dans la fenêtre apériodique autorisée dans la fenêtre périodique pour la même transaction).

Malheureusement, l'introduction de plusieurs phases de scrutation et la diversité de ressources utilisées qui n'ont pas de limites théoriques de taille peuvent provoquer rapidement l'explosion combinatoire en nombre d'états/transitions du graphe de marquage résultant. Pour faire face à ce problème classique des graphes d'accessibilité, nous avons associé des tailles relatives à chaque file d'attente. Le tableau suivant résume les différentes tailles du graphe en terme d'états/transitions et en fonction de nombre d'identifieurs et des tailles des files d'attente associées aux demandes explicites (req1, req2 et msg). Un nombre d'identifieurs égal à "1" signifie qu'une seule demande MA.req(id) est autorisée par le LLC (le paramètre `Last_id` doit être mis à un) et ensuite la fenêtre statique se termine. La première ligne (0, 0, 0) signifie que les réponses `rp_dat` ne peuvent pas porter des demandes explicites ce qui ressemble à la scrutation statique, alors que la deuxième ligne (Msg =1) signifie qu'une seule `rp_dat_msg` est autorisée, comme une alternative de `rp_dat` , dans la phase périodique et ainsi de suite pour les autres lignes.

Nombre d'identifieurs			1		2	
Rq1	Rq2	Msg	Nbr. états	Nbr. trans.	Nbr. états	Nbr. trans.
0	0	0	31	92	76	244
0	0	1	58	188	210	726
0	1	0	123	375	670	2214
0	1	1	231	775	1876	6678
1	1	1	455	1585	5578	20442
2	2	2	1505	5928		

Tableau 5.1 — Complexité en terme d'états-transitions du graphe de marquage de l'AB

Le tableau précédent montre que la taille du graphe est fonction de la permutation des identifieurs dans les files d'attente (msg, req1 et req2).

La simulation exhaustive pour les petits graphes ou par exécution des scénarios pour les grandes tailles ont révélé la plupart des erreurs syntaxiques, la sous-spécification ainsi que la bonne interprétation des prédicats Prolog utilisés pour définir des fonctions de mise à jour ou des conditions de test pour activer une transition (les clauses `provided` et `ensure`). Pour assurer ensuite les propriétés d'absence d'interblocage ou de la progression, nous avons utilisé des opérateurs prédéfinis. Enfin, les propriétés spécifiques ont été vérifiées en évaluant des formules de la logique temporelle combinatoire sur le graphe d'accessibilité résultant. Nous citons à titre d'exemple :

- lorsque l'AB reçoit une trame `rp_dat_rq` (ou `rp_dat_msg`), il va éventuellement envoyer une trame `id_rq` (ou `id_msg`) sur le même identifieur,
- il est inévitable de sortir d'une transaction de messagerie avec acquittement (ou sans acquittement) sans `rp_fin` ou défaillance de reprise (expiration temporisateur).
- la vérification qu'une trame `id_rq1` urgente doit être suivie par une trame `rp_rq1` urgente ou expiration du temporisateur associé a révélé la possibilité de recevoir une trame `rp_rq2` à la place; d'où la redondance inutile (sur-spécification) de `id_rq1` et `id_rq2` ainsi que `rp_rq1` et `rp_rq2` qui peuvent être remplacées par `id_rq` et `rp_rq`,
- la norme ne précise pas le comportement de l'AB dans le cas où une des ressources allouées est saturée.

2) VÉRIFICATION DU MODULE UPDATE_ID

Pour effectuer la vérification, nous avons choisi la configuration des identifieurs de la façon suivante :

Identifieur	100	101	102	104
free1		oui		
free2			oui	
spec	oui			
msg	oui		oui	
cons				oui

Tableau 5.2 — Exemple d'une configuration de quatre identifieurs

L'identifieur (100) peut être désigné spécifiquement pour porter soit une demande explicite de requêtes urgentes (`rp_dat_rq1`), soit une demande de transfert de message (trame réponse `rp_dat_msg`) soit les deux en même temps, l'identifieur (101) peut porter seulement des demandes libres (dynamiques) urgentes et l'identifieur (104) est consommé

par cette entité. Cette configuration a engendré un graphe de 544 états et 8160 transitions et ayant toutes les propriétés générales souhaitées comme l'absence d'interblocage et la réinitialisabilité.

La projection sur l'identifieur (100) a donné le graphe de la figure suivante dont on a supprimé quelques transitions (confirmation ou celles qui ne relèvent pas des demandes explicites comme `id_msg(100)`) pour la simplicité.

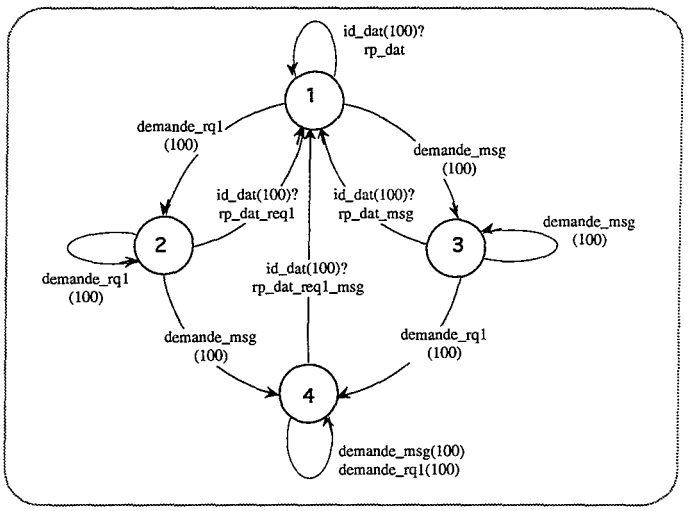


Fig. 5.4 — Vérification par projection

- L'état initial est l'état (1) où il n'existe pas de demande explicite associée à l'identifieur (100) et la réponse à une trame id_dat(100) sera rp_dat,
- l'état (2) correspond à une demande explicite de requêtes MA.req(req1, 100) et donc la réponse, suite à une trame id_dat(100), sera rp_dat_req1 et on revient à l'état initial,
- l'état (3) correspond à une demande explicite de message MA.req(msg, 100) et donc la réponse, suite à une trame id_dat(100), sera rp_dat_msg et on revient à l'état initial,
- l'état (4) correspond à deux demandes successives et dans n'importe quel ordre de requêtes et de messages et la réponse sera rp_dat_req1_msg suivi d'un retour à l'état initial.

Nous remarquons également que la projection a bien respecté la configuration de l'identifieur (100) pour les requêtes urgentes (`free1` et `spec` correspondent aux requêtes urgentes) et la messagerie.

3) VÉRIFICATION DU MODULE MAC

Nous avons envisagé pour le module `mac` (contrôle l'accès dans une station terminale) une configuration comportant un seul identifieur sur lequel tout type de demande peut circuler. Cela a engendré un graphe ayant 41 états et 335 arcs.

Quatre projections ont été effectuées sur ce graphe selon le trafic de variables, de messages et de requêtes :

- La projection sur le trafic identifié de variables a révélé un graphe réduit de 9/29 états/transitions illustré dans la figure suivante :

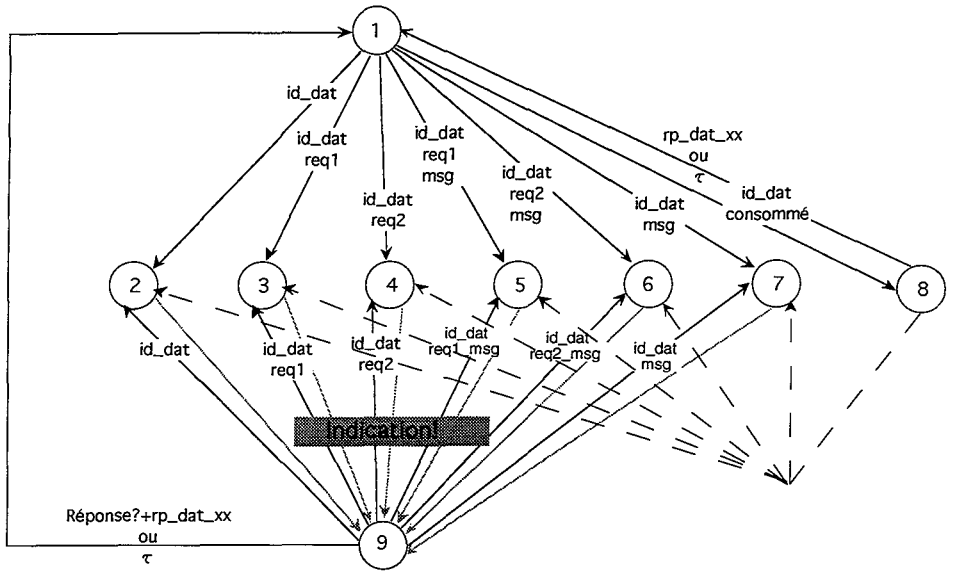


Fig. 5.5 — Projection sur le trafic de données

L'automate se trouve initialement dans l'état (1) correspondant à un état d'attente d'un identifieur et, éventuellement, une demande explicite en provenant du module `update_id`,

- l'état (2) correspond à la réception de l'identifieur sans demande explicite. La transition peut avoir lieu de n'importe quel état après avoir envoyé l'indication,
- l'état (3) correspond à la réception de l'identifieur avec une demande urgente de requêtes,
- les états {4, 5, 6, 7} correspondent à la réception de l'identifieur avec {requête normale, requête urgente et message, requête normale et message, message} respectivement,
- l'état (8) correspond à la réception d'un identifieur dont la station est consommatrice,
- la transition à l'état (9) se produit après avoir émis une indication au LLC pour envoyer la valeur associée à l'identifieur (il s'agit d'une trame `id_dat` suivie d'une MA.ind).

Nous avons procédé d'une façon similaire pour le trafic de demandes explicites de listes de variables et le trafic de messagerie.

4) VÉRIFICATION DU MODULE PC (MAC || UPDATE_ID || FRAME_RECEPTION)

Le graphe d'accessibilité global pour le cas d'un seul identifieur a révélé 2816 états et 22991 transitions.

Le test d'interblocage nous a aidé à localiser toutes les situations de réception non spécifiée, à spécifier la réaction du module `mac` en réponse à tout événement (trame) en provenance du module `frame_reception`. Par exemple, le cas lorsque le `mac` est en attente d'une réponse de donnée et il voit circuler une réponse de messagerie ou de requêtes, etc.

5) VÉRIFICATION DU MODULE PONT (MAC_SEG_A || MAC_SEG_B || MAC_SEG_C)

La complexité du module pont en terme de ressources utilisées et le critère asynchrone de comportement de ses composantes (communiquent par files d'attente) rendent la génération du graphe de marquage global très difficile, voire même impossible dans certaines conditions. Nous indiquons dans le tableau suivant la complexité du graphe généré pour une composante en fonction de la taille de la file d'attente de sortie qui lui est associée :

capacité de la file d'attente	0	1	2	3	4	5
complexité états/transitions	5/50	14/159	29/329	49/559	74/849	104/1199

Tableau 5.3 — Taille du graphe en fonction de la taille de la file d'attente

La vérification du comportement du pont a été effectuée en deux phases :

1. vérification du comportement du module `mac_seg_a` en interaction avec les autres ports du pont,
2. vérification du comportement du pont à un niveau supérieur d'abstraction : en faisant abstraction des modules contenus dans un segment FIP et ceux du pont, la vérification est portée sur la composition parallèle d'un segment avec le pont en fonction des interactions communes.

Dans la première phase, la vérification a concerné principalement les files d'attente et les buffers parce que le contrôle d'accès et la gestion de champs contrôle sont similaires à une station terminale. Des propriétés spécifiques au respect de la capacité des files d'attente, de la cohérence entre le transfert apériodique de messages et le transfert périodique.

Nous montrons dans le code suivant l'automate résultant de la spécification du pont qui décrit l'interaction avec le segment :

```
arc(1,[pont(from_seg_a)?id_xx(3,101),pont(to_seg_a)!rp_dat_xx(_)],
[pont(id_dat_prod)],1).
arc(1,[pont(from_seg_a)?id_xx(3,102)],[pont(id_dat_cons)],2).
arc(1,[pont(from_seg_a)?id_xx(5,101),pont(to_seg_a)!frame(_,_,_,_)],
[pont(id_msg_source)],3).
arc(1,[pont(from_seg_a)?id_xx(5,102)],[pont(id_msg)],4).
arc(2,[pont(from_seg_a)?frame(_)],[pont(rp_dat_cons)],1).
arc(3,[pont(to_seg_a)!frame(64)],[pont(send_rp_fin)],1).
arc(4,[pont(from_seg_a)?frame(_,_,_,1000)],
[pont(rp_msg_noack_dest)],5).
arc(5,[pont(from_seg_a)?frame(64)],[pont(recep_rp_fin)],1).
```

6) VÉRIFICATION DU MODULE SEGMENT (AB || MEDIUM || PC || RESPONDER)

Une génération d'un graphe représentant la composition parallèle de ces quatre modules n'a pas été possible à cause de l'explosion des états. La simulation exhaustive ou la génération de scénarios, sans générer le graphe de marquage, nous ont aidés à vérifier quelques propriétés spécifiques à un segment.

7) VÉRIFICATION DU MODULE INTERCONNEXION (PONT || SEG)

La vérification a porté seulement sur une spécification abstraite de chaque module en fonction des interactions communes. Cette spécification cache les détails concernant les interactions entre les sous-modules composants.

Nous récapitulons maintenant les principales techniques qui nous ont aidé à valider l'interconnexion du pont à un segment :

Vérification par simulation : toutes les erreurs syntaxiques ainsi que le comportement fini attendu ont été vérifiés dans cette étape. Nous ajoutons la possibilité de navigation dans les grands systèmes en termes de nombre d'états où l'explosion combinatoire d'états peut se produire pour vérifier sa conformité à un besoin donné ou pour enlever une ambiguïté. En outre, la simulation interactive permet la mise à jour dynamique des variables et des événements ce qui n'est pas rencontré dans l'analyse de graphe de marquage.

Vérification par l'analyse d'accessibilité sur le graphe de marquage : nous citons les techniques suivantes :

- l'utilisation des opérateurs prédéfinis pour vérifier l'absence d'interblocage, la réinitialisabilité et la vivacité qui donne une idée initiale du comportement interne du système.
- les techniques de projection (bisimulation forte ou faible de Milner) sont utilisées soit pour réduire la taille du graphe d'accessibilité selon les propriétés qui doivent être respectées, soit pour vérifier la conformité d'une spécification à deux niveaux voisins d'abstraction.
- l'exploration du graphe d'accessibilité pour vérifier des propriétés du type critère/service qui impliquent la recherche des chemins pour vérifier par exemple que chaque identifieur diffusé par l'AB sera suivi soit par la valeur de l'objet associé à l'identifieur soit par une expiration du temporisateur associé à l'AB.
- le test d'invariants, par exemple dans un état donné, une seule entité a l'accès exclusif au médium ou pour vérifier le respect du nombre de messages dans une file d'attente.

Nous récapitulons dans le tableau suivant les techniques et les propriétés que nous avons vérifiées :

propriété	Simulation	Analyse Accessibilité	Projection	Invariant	logique temporelle combinatoire
interblocage	X	X	X	X	
réinitialisation		X			
vivacité			X	X	X
boundedness		X	X	X	X
sûreté		X	X		X
ambiguîté	X	X			
surspécification	X	X			
complétude		X			
reprise erreurs	X	X		X	
terminaison		X			
équité				X	X
livelock		X			X
consistance	X		X		
prop. spécifiques	X	X	X	X	X

Tableau 5.4 — Vérification d'un pont interconnecté à un segment FIP : techniques et propriétés

L'explosion combinatoire des états est toujours considérée comme le problème le plus délicat pour l'analyse d'accessibilité. Car, lorsque le graphe d'accessibilité dépasse la taille de mémoire disponible, l'analyse n'est plus possible.

Nous rappelons que l'utilisation de l'arc inhibiteur (test lorsqu'il n'y a pas de jeton dans une place) est d'une grande importance, e.g. limiter la taille d'une file d'attente à zéro ou tester que la file est vide. Alors que la possibilité d'avoir des poids sur les transitions nous a

permis de bien traduire le fait d'envoyer une liste de variables pour laquelle le nombre de variables dénote le numéro de jetons à allouer dans la file d'attente.

5.4. Conclusion

Nous avons vu dans ce chapitre comment les réseaux de Petri étendus par des mécanismes d'étiquetage pour représenter la synchronisation avec l'environnement ainsi que par des prédicats pour permettre une spécification plus concise et générique peuvent être utilisés comme un langage de spécification formelle. Un point fort dans cette extension revient à préserver toutes les propriétés RdP en bénéficiant d'autant de techniques d'analyse que les réseaux de Petri élémentaires.

Les bases mathématiques de RdP qui leur fournissent une sémantique bien définie nous ont permis d'exclure toutes les ambiguïtés résiduelles dans les spécifications informelles des normes ainsi qu'à valider l'architecture par diverses techniques de vérification.

Ces résultats servent à implanter un pont dans des segments FIP. Ce pont peut gérer tous les services périodiques ainsi que les demandes apériodiques de messages sans acquittement en premier plan et ils peuvent servir également pour définir les frontières MAC/LLC conformément aux recommandations de l'ISO.

Chapitre 6

Algorithme de routage multipoint de plus courts délais et de coût minimal

Les résultats obtenus jusqu'à présent nous ont permis de définir et de valider l'opération d'un pont en interaction avec un réseau FIP. Nous avons dit également qu'un fonctionnement correct du pont est conditionné par l'existence de deux tables de routage concernant les messages point à point et multipoint ainsi que d'une liste d'objets produits. Nous ajoutons que la communication dans FIP est souvent du type multipoint et que la topologie d'interconnexion peut être maillée. Par conséquent, les tables et les listes de routage doivent être cohérentes et elles doivent garantir également le non bouclage. Il faut considerer aussi les délais de transfert et le nombre de ponts traversés pour la mise en œuvre de ce transfert.

Nous nous sommes fixés comme objectif de développer un algorithme qui tient compte des contraintes précédentes lors de la construction de ces tables.

Il est à noter que les sections : 6.1 à 6.5, sont indépendantes du réseau FIP. La section 6.6 définit un exemple de l'application de l'algorithme pour l'interconnexion de réseaux FIP.

6.1. Introduction

Nous avons montré dans le premier chapitre l'insuffisance des protocoles de diffusion multipoint qui sont actuellement utilisés ou en cours d'expérimentation pour répondre à toutes les contraintes utilisateur, notamment celles liées aux délais de propagation et de coût du transfert en matière de ressources et d'entités intermédiaires utilisées. Ces protocoles sont d'intérêt général et sont conçus particulièrement pour répondre aux besoins relatifs à la robustesse, la convergence rapide lors de modification de la relation d'appartenance aux groupes et enfin ils utilisent des protocoles de bas niveau point à point qui, à leur tour, ne garantissent pas cette catégorie de contraintes.

Le but principal de ce chapitre est de définir un algorithme de routage multipoint en mode sans connexion qui assure l'acheminement de données en provenance d'une source

spécifique et vers un groupe de destinataires tout en garantissant les contraintes précédentes et le non bouclage.

Pour ce faire, nous procédons de la manière suivante : une représentation du problème de routage en terme de graphes va nous permettre de formaliser notre problématique et d'étudier les algorithmes existants. La problématique et la solution fournie sous forme d'un algorithme qui a été mis en œuvre pour cette fin feront l'objet de la troisième et la quatrième section. Des analyses de performance ainsi que des études comparatives avec les autres algorithmes feront l'objet de la cinquième section. La sixième section est consacrée à un exemple de l'utilisation de l'algorithme pour le cas de FIP. Nous concluons par les domaines d'application de cet algorithme ainsi que les éventuelles extensions qui lui peuvent être ajoutées.

6.2. Représentation du routage multipoint

6.2.1. Définitions

1- Graphe : un graphe `G` est une paire `(V, E)`, où `V` est un ensemble fini et non vide d'éléments appelés `nœuds`, et `E` est un ensemble fini de paires distinctes et non ordonnées d'éléments distincts de `V` appelé `arcs` [Beineke 78].

Si nous enlevons de la définition précédente la restriction que les arcs doivent être distincts, nous retrouvons ainsi la définition d'un multigraphe (Fig. 6.1).

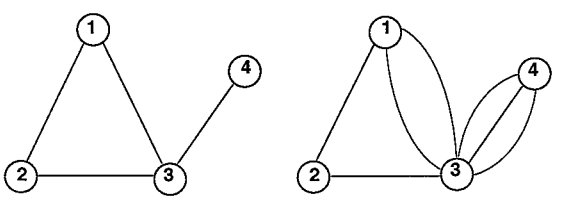


Fig. 6.1 — Graphe et multigraphe

Dans le cas où une fonction `coût : E` $\rightarrow \mathbb{R}^+$ associant à chaque arc son coût ou son poids (un débit d'une liaison par exemple) est définie, le coût d'un arbre sera la somme de coûts associés à ses arcs. Nous allons fixer, pour le reste de ce chapitre, cette fonction à l'unité (`coût = 1`). Par conséquent, le coût d'un arbre sera égal au nombre de ses arcs.

2- arbre : un arbre recouvrant un graphe connexe de `N` nœuds est un sous graphe connexe de `N` nœuds qui ne contient pas de circuits et par conséquent :

1) il contient `N-1` arcs, et
2) il existe un chemin unique entre n'importe quel couple de nœuds.

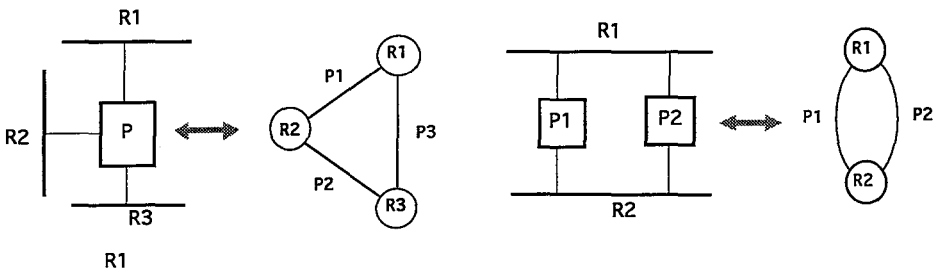
Lorsque l'arbre recouvre un sous ensemble `D` $\subseteq$ `V`, il est appelé un sous arbre recouvrant pour le quel la conséquence (1) est plus valide. En fait, si `D` contient `M` nœuds alors le coût (nombre d'arcs) du sous arbre sera compris entre `M` et `N-1`.

6.2.2. Représentation d'interconnexion par des graphes

Nous avons défini précédemment (cf. §1.7 et §1.8) l'adresse de groupe et le routage multipoint. Quant à la représentation de la topologie d'interconnexion, deux méthodes sont traditionnellement utilisées :

1) les nœuds V dénotent l'ensemble des sous réseaux et les arcs E dénotent l'ensemble des entités relais (pont/routeurs).
2) les nœuds V dénotent l'ensemble des entités relais et les arcs E dénotent l'ensemble des sous réseaux.

Nous avons retenu la première représentation parce qu'elle permet de traduire la plupart des topologies d'interconnexion possibles en respectant la définition précédente du graphe (Fig. 6.2) :



(a) cas d'un pont multiport (b) cas des ponts parallèles

Fig. 6.2 — Exemples de représentation de topologies d'interconnexion par des graphes

Nous remarquons que la topologie donnée dans la figure 6.2(a) n'est pas représentable par la deuxième méthode et de même pour une topologie réduite à un seul réseau.

Nous pouvons également représenter les sous-réseaux et les relais par des nœuds et les interconnexions par des arcs. Cette solution qui génère le même graphe que la première représentation peut mieux représenter une interconnexion longue distance par deux demi-ponts par exemple.

6.2.3. Routage en point à point et en diffusion

Le routage en point à point est souvent traité comme un problème de plus court chemin dans un graphe. L'algorithme de [Dijkstra 59] dans les graphes est de loin le plus utilisé à cette fin.

En ce qui concerne la diffusion de données vers tout nœud du graphe, les techniques suivantes ont été proposées dans la littérature :

- transmission par les moyens de routage en point à point d'une copie du paquet à chaque nœud séparément,
- inondation d'un paquet sur toutes les liaisons de sortie en utilisant l'algorithme de hot Potato par exemple,
- construction d'un arbre qui recouvre la topologie d'interconnexion et qui est partagé par tous les nœuds. En réception de paquet de diffusion, chaque nœud copie et retransmet ce paquet sur chaque branche de l'arbre sauf celle d'où le paquet provient. Dans le cas où l'arbre minimise la somme de coûts associés aux arcs, il est appelé MST[1],
- retransmission à la base de la source (source spécifique) : chaque nœud qui connaît la topologie globale d'interconnexion maintient une table qui associe à chaque source

1 Minimum Spanning Tree,

l'ensemble de liaisons de sortie appartenant à un arbre de plus courts chemins entre la source et le reste des nœuds. Cet arbre est appelé SPT[2],

- retransmission par chemin inverse RPF[3] [Dalal 78] : en réception, sur une liaison d'entrée, d'un paquet de diffusion, le nœud retransmet ce paquet sur toutes les liaisons de sortie sauf celle d'où le paquet provient, si et seulement si ce nœud estime que le plus court chemin pour atteindre la source est via la liaison d'entrée.

6.2.4. Routage en multipoint

Le routage multipoint (la diffusion est restreinte ici à un sous ensemble de nœuds $D \subseteq V$) est un problème plus général que la diffusion et par conséquent les algorithmes de diffusion sont parfois non applicables ou peuvent provoquer une perte d'efficacité.

La solution qui a été retenue pour ce faire est de trouver un sous arbre recouvrant un sous ensemble de nœuds.

Le type du sous arbre peut être `source-spécifique` ou `partagé` par tous les nœuds du groupe [Wei 94]. Dans le premier cas, on cherche à construire un sous arbre de plus courts chemins d'une source (ou racine) spécifique vers l'ensemble de nœuds `D`. Par contre dans le deuxième cas, le sous arbre doit être partagé par tous les membres du groupe. Par conséquent :

- dans le premier cas, un sous arbre doit être associé à chaque couple (`source, groupe`) alors que dans le deuxième cas, on associe un sous arbre par groupe,
- le premier cas convient au cas où la source n'appartient pas au groupe de destinataires comme par exemple la communication entre un producteur et un groupe de consommateurs ou entre un client et un groupe de serveurs alors que le deuxième cas convient à une communication dans un groupe fermé y compris la source.

En fonction du type de sous arbre et du critère d'optimisation choisi pour trouver le sous-arbre, nous pouvons identifier trois familles d'algorithmes optimisant :

6.2.4.1. Coût de l'arbre

Cette famille d'algorithmes cherche un sous arbre partagé d'un coût[4] minimal couvrant l'ensemble `D`. Ce problème est connu dans la littérature comme le problème de l'arbre de Steiner dans les graphes SMT[5] [Gilbert 68] et a été prouvé NP-complet par Karp [Karp 72].

Nous citons parmi les algorithmes polynomiaux les plus efficaces, le MST [Bharath 83], l'algorithme de Rayward-Smith [Rayward 83], [Rayward 86], celui inventé par Kou, Markowski et Berman (connu par l'algorithme KMB) [Kou 81] et d'autres [Tanaka 90] et [Tode 92].

L'application de ces algorithmes à des protocoles réels n'a pas été faite à cause de la complexité des calculs qui entraînent un overhead significatif ainsi qu'à cause de la difficulté de développer des versions distribuées.

2 Shortest Path Tree,
3 Reverse Path Forwarding,
4 un coût d'un arbre est la somme de coûts associés à ses arcs (nombre d'arcs dans notre cas),
5 Steiner Minimal Tree,

6.2.4.2. Les plus courts chemins

Dans cette famille d'algorithmes, un arbre du type `source-spécifique` composé des plus courts chemins (SPT) entre la source et tous les destinataires du groupe multipoint doit être trouvé. Nous citons parmi les algorithmes de cette famille : l'algorithme de diffusion RPF et son extension proposée dans [Deering 90a] et [Deering 91]. Ces algorithmes dérivent l'arbre de plus courts chemins à partir des plus courts chemins individuels et ils ont donc l'inconvénient de ne pas optimiser le coût total de l'arbre et d'utiliser la diffusion qui est coûteuse.

Bien que cette stratégie n'offre pas une solution optimale du point de vue du coût total, elle s'est avérée très pratique en raison de sa simplicité ainsi que de la possibilité de répartir l'algorithme sur plusieurs sites. Le MOSPF et le MBone [Castner 92] sont des exemples de protocoles réels implémentant ce type d'algorithmes d'une façon répartie.

6.2.4.3. Compromis entre le coût total et les longueurs des chemins

Une variété d'algorithmes ayant comme mission de trouver un compromis entre le coût total de l'arbre avec les longueurs de chemins, ont été proposés dans [Ballardie 94]. L'attention est faite ici pour trouver un centre du graphe et relier ainsi ce centre au groupe de destinataires par les plus courts chemins.

Bien que ce type d'algorithmes améliore les longueurs des chemins dès que le centre du graphe a été trouvé, il complique les calculs par des surcoûts supplémentaires dûs à la recherche du centre qui est un problème NP-complet et il concentre le trafic sur un sous-ensemble de liaisons conduisant vers le centre. La seule application de cet algorithme se trouve dans PIM[6] [Deering 94] qui peut être utilisé dans le MBone pour le routage multipoint interdomaine avec des groupes de taille réduite.

6.3. Problématique

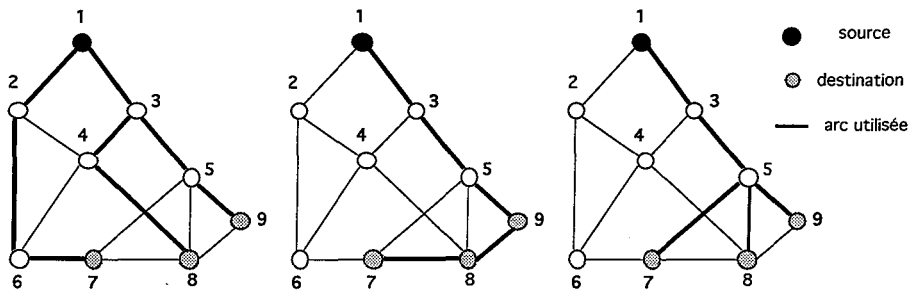
Notre objectif principal, dans le reste de ce chapitre, est de trouver un sous arbre du type `source-spécifique` recouvrant un ensemble de nœuds (groupe de destinataires) qui vérifie les conditions suivantes:

1) chaque chemin de la source vers un nœud destinataire est le plus court chemin en nombre d'arcs (on suppose qu'un coût égal à un est associé à chaque arc),
2) le coût total (en nombre d'arcs) du sous arbre résultant doit être minimal; cela est équivalent à dire que le nombre des nœuds intermédiaires est minimal,

Il s'agit, du point de vue interconnexion des réseaux, de réduire autant que possible le nombre des relais traversés sans que les longueurs des chemins (ou délais de transmission) soient perturbées.

La figure suivante montre notre objectif en comparaison avec les résultats qui peuvent être rendus par les autres familles d'algorithmes:

6 Protocol Independant Multicast,



(a) SPT (b) MST et SMT (c) sous arbre souhaité

Fig. 6.3 — La problématique et les algorithmes existants

Nous remarquons que la famille (a) d'algorithmes calcule les plus courts chemins séparément du coût total de l'arbre et peut rendre des coûts très élevés (coût égal à 8 ici). La deuxième famille (idem. pour la troisième famille) ne nous intéresse pas parce qu'elle ne fournit pas les plus courts chemins (le chemin entre 1 et 7 a une longueur de 5) d'une part, par ailleurs elle utilise des arbres du type partagé où la source doit faire partie du groupe de destinataires.

6.4. Présentation de l'algorithme MSPST[7]

Les études précédentes montrent qu'aucun algorithme ne répond à notre problématique. Les algorithmes optimisant le coût total de l'arbre ne font pas la distinction entre la source et le groupe de destination et peuvent produire des chemins très longs inadaptés à un trafic temps réel. Les algorithmes de plus courts chemins satisfont la première condition de notre problématique mais en revanche ils ne tiennent pas compte du nombre de relais (ressources) qui doivent être utilisés; ils vont servir ainsi comme plates-formes de test pour évaluer la performance de notre algorithme.

Pour le reste de ce chapitre, nous définissons, dans un graphe $G = (V , E)$ non dirigé et sans poids (coût = 1), les variables suivantes:

- $S \in V$ dénote la source,
- $D \subset V$ dénote le groupe de destinataires,
- un chemin $Path(V_0,V_n)$ entre V_0 et V_n est un ensemble ordonné de nœuds $\{V_0,\ldots,V_i\ldots,V_n\}$ tel que : $\forall\ 0 \leq i \leq (n-1),\ (V_i,V_{i+1}\ ) \in E$
- $distance(V_i,V_j)$ est la longueur du plus court chemin entre V_i et V_j en nombre d'arcs par rapport à G,
- un arbre T comporte :
  - un ensemble de nœuds distincts $N_T = \{N_0,\ldots,N_i,\ldots,N_n\}$
  - une racine (ou source) $S \in N_T$
  - une fonction "Pred" qui associe à chaque nœud son prédécesseur dans l'arbre :
    $Pred : N_T \rightarrow N_T$, $(Pred(S) = S)$ de façon que :
    $$\forall\ N_i \in N_T - \{S\},\ (Pred(N_i)\ ,\ N_i) \in E$$

Application à l'exemple de la figure 6.3.c :

$S = 1,\ D = \{7,\ 8,\ 9\},$

$Path\ (1,\ 8) = \{1,\ 3,\ 5,\ 8\},$

7 Minimal-cost Shortest Path Spanning Tree,

distance (1, 8) = 3,
L'arbre recherché a les composantes suivantes :
N_{MSPST} = {1, 3, 5, 7, 8, 9}, la racine est 1,
Pred(7) = 5, Pred(8) = 5, Pred(9) = 5, Pred(5) = 3, Pred(3) = 1

L'objectif principal de l'algorithme [Saba 95a] est le suivant:

Construire un sous arbre de coût minimal recouvrant l'ensemble des destinataires D de telle façon que la racine soit la source S et le chemin entre la source et chaque destinataire soit le plus court chemin en nombre d'arcs par rapport au graphe initial.

Entrées : un graphe G, une source $S \in V$, un groupe de destinataires $D \subset V$.

Résultats : un arbre MSPST pour lequel :

- $\forall d \in D / d \in N_{MSPST}$,
- $S \in N_{MSPST}$,
- $\forall d \in D$ / si $S = Pred^k(d)$ alors $k = distance(S,d)$,
- coût (MSPST) qui vaut $card(N_{MSPST})-1$ est minimal

Idée de l'algorithme : chercher les nœuds qui sont importants pour la solution finale; il s'agit ici des nœuds partagés par le plus grand nombre des plus courts chemins entre la source et les destinataires. Pour représenter cette importance, pour un groupe de N destinataires, nous allons utiliser une fonction *occurrence* qui associe à chaque nœud un entier entre 0 et N. Cette occurrence est égale à 0 si aucun des plus courts chemins de la source vers chaque destinataire ne passe par ce nœud, alors qu'une occurrence égale à N signifie que pour chaque destinataire, il y a au moins un plus court chemin qui passe par ce nœud (le nœud 5 dans l'exemple précédent).

Le tableau suivant donne les occurrences respectives des nœuds de la figure 6.3 (l'occurrence de la source est facultative) :

nœud	1	2	3	4	5	6	7	8	9
occurrence	0	2	3	1	3	1	1	1	1

Tableau 6.1 — Calcul global de la fonction heuristique

Nous sélectionnons en deuxième temps le chemin qui maximise, pour chaque destinataire, la somme d'occurrences de ses nœuds intermédiaires.

Par exemple, pour le destinataire 7, il y a 2 plus courts chemins possibles: {1,2,6,7} avec une somme d'occurrences de 4 et {1,3,5,7} avec une somme d'occurrence de 7. Dans ce cas, le chemin {1,3,5,7} sera sélectionné ainsi que les occurrences de ses nœuds seront incrémentées par 1 pour privilégier ces nœuds. Donc, occ(3) := 4, occ(5):= 4 et occ(7):=2. Et ainsi de suite pour les autres nœuds.

algorithme MSPST (G:graphe, S:source, D:Ens_dest, T:arbre)

Etape 1 (calcul de distances entre les nœuds) :

calculer la matrice M des plus courtes distances pour tous les couples des nœuds (V_i,V_j) [Floyd 62].

$\forall$ V_i,V_j (deux nœuds de G)/ M(V_i,V_j) est la distance la plus courte en nombre d'arcs entre le nœud V_i et le nœud V_j,

{Ce calcul nous permet de modifier la source ou les destinataires sans avoir recours à ré-exécuter l'algorithme de Floyd}

Etape 2 (la fonction heuristique globale) :

pour chaque $V_i \in D$ faire

1. calculer l'ensemble V des nœuds intermédiaires (i.e., situés sur, au moins, un plus court chemin entre S et Vi),
2. pour tout $e \in V$ faire
   occurrence(e) := occurrence(e)+1 (initialement l'occurrence est nulle)

Etape 3 (construction de l'arbre) :

pour tout $V_i \in D$ et $V_i \notin T$ faire *(initialement T := {S})*

1. calculer tous les plus courts chemins entre V_i et S (un chemin est le plus court si le nombre de nœuds intermédiaires = distance(S,V_i)+1),
2. sélectionner le plus court chemin P_{max} = {S,…,I,…,V_i} qui maximise* :

$$\Sigma \text{ occurrence(e)} \;/\; e \in P_{max}$$

3. pour chaque nœud $e \in P_{max}$ faire
   a) occurrence(e) := occurrence(e) + 1
   b) T := T $\cup$ {e}

finalg

(*) un chemin P_d, entre un nœud d et la source S, peut être soit un ensemble vide lorsque $d \in T$ ou soit l'ensemble ordonné de nœuds $\{V_0,\ldots,V_i,\ldots V_n\}$ de manière que V_0=d, $\forall$ $0 \le i \le (n-1)$, $(V_i,V_{i+1}) \in E$ (i.e., il existe un arc dans G qui relie V_i à V_{i+1}), et $\forall$ $0 \le i \le (n-1)$, $V_i \notin T$, $V_n \in T$ (informellement, un chemin vers la source s'arrête dès qu'on rencontre un nœud qui a été déjà ajouté à l'arbre courant T). Puisque la source se trouve sur tous les chemins, son occurrence est facultative.

A titre d'exemple, nous avons exécuté l'algorithme sur le graphe suivant :

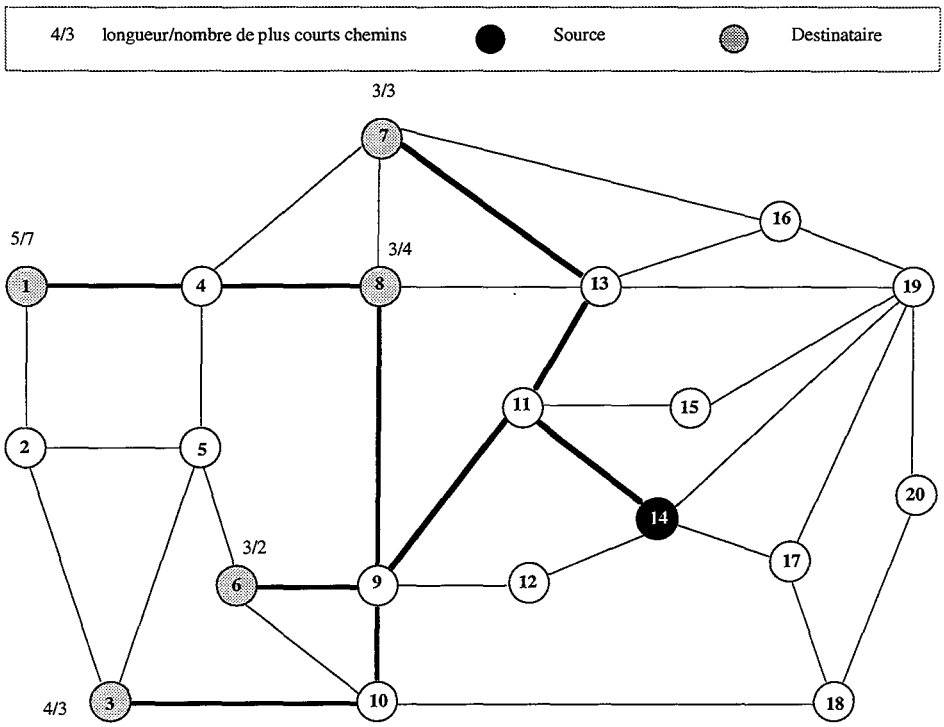


Fig. 6.4 — Exécution de l'algorithme MSPST sur un graphe de 20 nœuds

La table suivante montre la construction progressive de l'arbre minimal des plus courts chemins:

Etape	nœud	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	occ.	1	0	1	1	0	1	2	2	4	1	5	4	3	0	0	2	1	1	3	0
3	Dest=1	**2**	0	1	**2**	0	1	2	**3**	**5**	1	**6**	4	3	**0**	0	2	1	1	3	0
3	Dest=3	2	0	**2**	2	0	1	2	3	**5**	**2**	6	4	3	0	0	2	1	1	3	0
3	Dest=6	2	0	2	2	0	**2**	2	3	**5**	2	6	4	3	0	0	2	1	1	3	0
3	Dest=7	2	0	2	2	0	2	**3**	3	5	2	**6**	4	**4**	0	0	2	1	1	3	0

Tableau 6.2 — Un scénario d'exécution de l'algorithme

Nous indiquons dans la première ligne le numéro du nœud et dans la deuxième ligne les occurrences respectives de chaque nœud. Ces occurrences seront modifiées au fur et à mesure de la construction de l'arbre (e.g., `occ(13)=3` signifie que le nœud `13` est situé sur trois plus courts chemins entre la source `14` et les destinataires, à savoir : ceux reliant les destinataires `{1,7,8}`).

Dans la troisième ligne, nous sélectionnons le plus court chemin entre la source `14` et le nœud `1`. C'est le chemin composé des nœuds `{1,4,8,9,11,14}` parce que la somme des occurrences de ces nœuds (égale à `13`) est le maximum par rapport aux `6` autres plus courts chemins. Nous ajoutons maintenant les nœuds `{1,4,8,9,11,14}` à l'arbre courant et nous incrémentons leurs occurrences par `1`.

Dans la quatrième ligne, le plus court chemin vers `3` sera `{3,10,9,11,14}` maximisant la somme d'occurrence de ses nœuds. Et parce que le nœud `9` a été déjà ajouté à l'arbre courant, le chemin s'arrête au nœud `9` et nous incrémentons les occurrences successives

des nœuds {3,10} par 1 pour les ajouter à l'arbre courant. Et ainsi de suite pour les nœuds 6,7 et 8.

Il reste à signaler que l'exécution de l'algorithme sur plusieurs types de topologies a montré que des résultats plus optimaux peuvent être obtenus si nous exécutons l'algorithme deux fois : en premier temps, nous construisons seulement les plus courts chemins qui sont uniques et le deuxième temps est consacré pour le reste. La figure suivante montre un cas d'amélioration (nous devons atteindre {4,7,10,13} à partir de {1}):

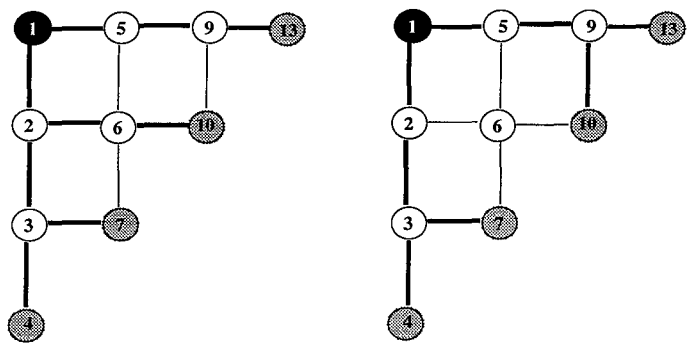


Fig. 6.5 — Exemple de la version améliorée de l'algorithme

6.5. Analyse de performance

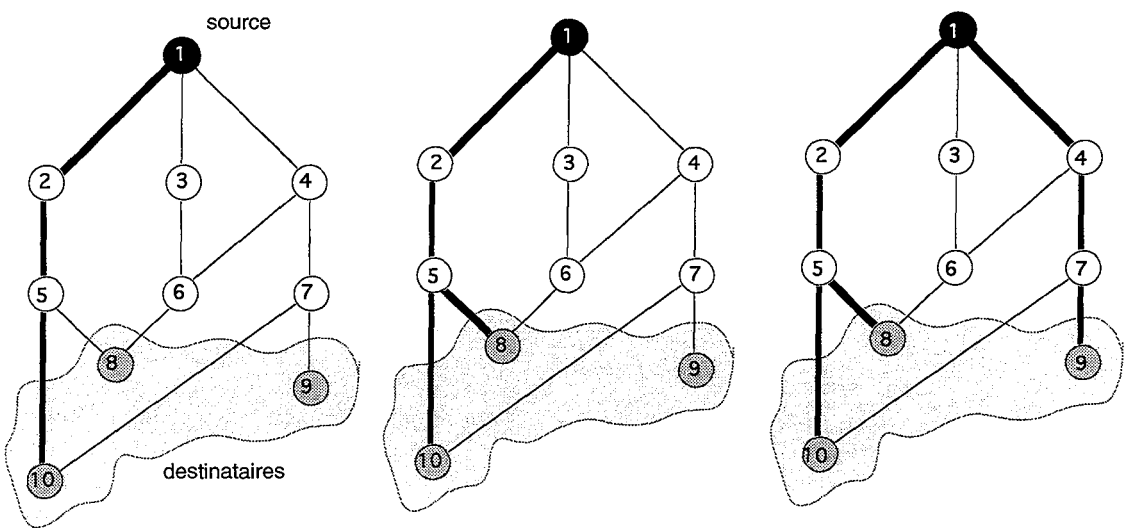
6.5.1. Plates-formes de tests

Puisque les algorithmes d'énumération exhaustive qui donnent les résultats exacts sont NP_complets, nous sommes amenés à implémenter le SPT et un algorithme testbed : le SPT construit un arbre recouvrant de plus courts chemins ayant la source comme racine et tous les autres nœuds du graphe comme destinataires; nous éliminons ainsi , à partir des feuilles, tous les nœuds qui ne font pas partie du groupe de destinataires (voir Fig. 6.7 pour un exemple). Le SPT est de loin le plus utilisé dans les protocoles de routage multipoint actuels comme dans le MBone par exemple. Les avantages principaux de SPT viennent du fait qu'il est facile d'implémenter une version répartie de cet algorithme, que sa complexité est de l'ordre de (N LOG N où N est le nombre total de nœuds du graphe) et enfin on peut profiter de routage en point à point pour l'intérêt du routage multipoint (voir MOSPF de §1.8.3.). Mais malheureusement les avantages précédents sont au détriment de nombre de routeurs traversés parce qu'aucune considération n'est faite pour minimiser le nombre total de routeurs traversés (coût de l'arbre). Bien que le SPT ne tient pas compte du coût total de l'arbre, il a été choisi comme plate-forme de test pour les autres algorithmes [Wei 94]. Le deuxième algorithme *testbed*, inspiré de [Takahashi 80], est une version améliorée de SPT parce que nous construisons l'arbre de façon progressive mais en tenant compte du coût total. Nous avons choisi cet algorithme parce qu'il correspond bien à la mise à jour dynamique des arbres. Pour ajouter un nouveau destinataire, il suffit de trouver le nœud le plus proche à ce destinataire et ajouter le chemin entre ces deux nœuds; de même pour supprimer un destinataire, il suffit de supprimer le nœud correspondant à ce destinataire et supprimer ainsi le prédécesseur de ce nœud dans l'arbre s'il n'est pas prédécesseur d'autre

nœuds; et ainsi de suite. Un pseudo-code de cet algorithme ainsi qu'un exemple d'exécution sont donnés dans la suite :

algorithme Testbed

0. T := {S} et D := groupe de destinataires,
1. tant que D ≠ ϕ faire
   - sélectionner arbitrairement un nœud V ∈ D
   - calculer un plus court chemin P entre V et la source S qui a le moins d'arcs à ajouter à T,
   - T := T ∪ P
   - D := D - V



1) ajouter un plus court chemin entre 10 et T (= {1})
   T devient {1,2,5,10}

2) ajouter un plus court chemin entre 8 et T
   T devient {1,2,5,8,10}

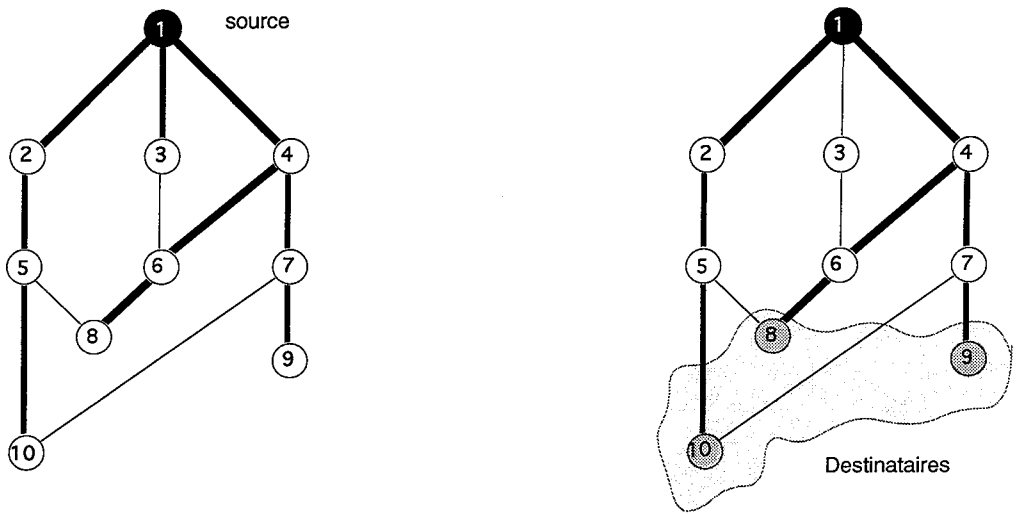
3) ajouter un plus court chemin entre 9 et T
   T devient {1,2,4,5,7,8,9,10}

Fig. 6.6 — Exemple d'une exécution de l'algorithme testbed

Si nous voulons supprimer le destinataire 10 de l'arbre, il suffit de supprimer ce nœud (avec la branche) et nous examinons le nœud 5 = Pred(10) si ce nœud est prédécesseur d'un autre nœud (nœud 8 dans l'exemple) nous arrêtons la suppression si ce n'est pas le cas on supprime ce nœud et on examine son prédécesseur...

Dans tous les cas, nous ne sommes pas partisans de la mise à jour dynamique de l'arbre parce qu'au bout de quelques modifications, nous nous éloignerons de la solution optimale (si on ajoute le nœud 6 via le nœud 4 et on ajoute ainsi le nœud 3 via le nœud 1, on obtient l'arbre de diffusion totale).

Quant à l'algorithme SPT, nous donnons dans la figure 6.7 un exemple d'exécution de cet algorithme :



1) construire un arbre de diffusion

2) supprimer les nœuds et les branches non utiles

Fig. 6.7 — Exécution de SPT sur le même graphe

Les exemples précédents montrent bien le comportement non déterministe de testbed parce qu'il dépend de l'ordre de choix des nœuds. Et par conséquent, nous pouvons nous rapprocher de la solution exacte en exécutant cet algorithme autant de fois qu'il existe de combinaisons possibles de destinataires. Par contre, l'algorithme SPT qui est déterministe une fois l'arbre de diffusion construit mais il consomme beaucoup de nœuds intermédiaires comme nous allons le voir dans les études comparatives.

Les comparaisons ont été effectuées sur des graphes générés arbitrairement en suivant le modèle donné dans la section suivante.

6.5.2. Modèle de graphes aléatoires et sans poids

Le modèle de génération aléatoire de graphe qui a été introduit par [Waxman 88] et qui est utilisé souvent pour la fin de test permet d'obtenir des variétés de topologies d'interconnexion sous forme de graphes mais malheureusement il ne convient pas pour des graphes non valués. Pour ce faire, nous avons défini un modèle de graphe qui, en se basant sur le nombre des nœuds et des arcs ainsi que de leurs localisations géographiques, permet d'associer un arc à deux nœuds voisins de la façon suivante :

- les nœuds sont supposés répartis sur une grille rectangulaire ayant `L` comme longueur et `W` comme largeur ,
- le nombre des arcs d'un graphe est une fonction de `L` et de `W`,
- un arc est permis uniquement entre deux nœuds voisins. Ceci permet de représenter un coût (distance) de `1`,
- la `source` est un nœud qui est arbitrairement choisi dans la grille précédente,
- le groupe de destinataires peut varier entre `2` et `L*W-1` (la `source` ne peut pas être incluse),
- la relation de voisinage peut avoir une des deux formes suivantes :

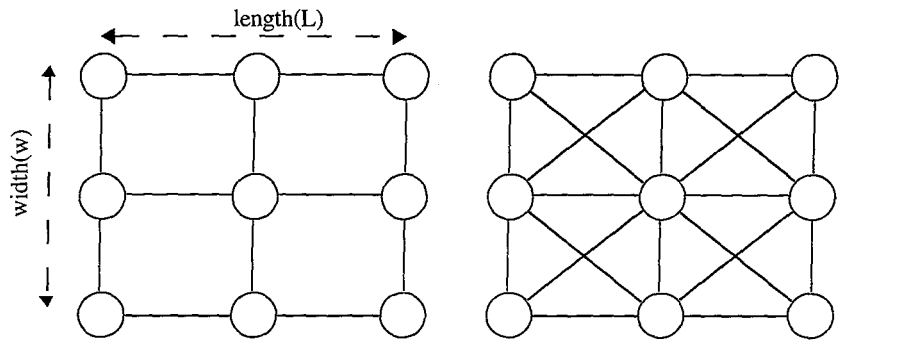


Fig. 6.8 — Les deux relations de voisinage possibles

Il faudrait bien noter ici que la forme (a) de voisinage représente l'interconnexion de réseaux par l'intermédiaire des ponts bi-ports seulement parce que, si un nœud n_1 relié à n_2 par un pont; n_2 relié à n_3 par un autre pont; il n'est pas possible d'avoir un pont entre n_1 et n_3 dans ce type de voisinage. Par contre, la forme de voisinage donnée dans (b) permet de représenter des ponts multi-ports.

6.5.3. Résultats de comparaison

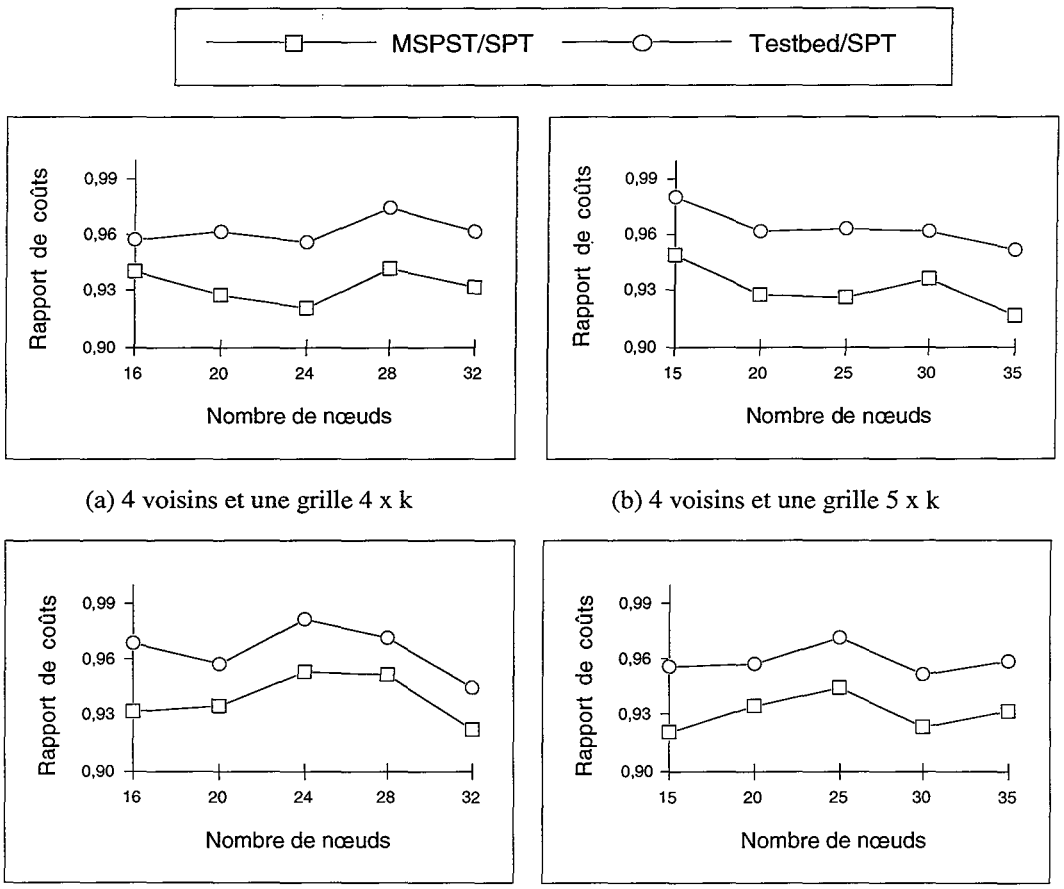
6.5.3.1. La performance en fonction de nombre de nœuds

Nous avons exécuté les trois algorithmes (SPT, Testbed et MSPST) sur `50` graphes aléatoires, pour chaque mesure, qui ont les caractéristiques suivantes :

- la taille de groupe de destinataires est fixée à `(L+W)/2`, où `L` est la longueur de la grille et `W` est sa largeur
- les deux relations de voisinage ont été considérées,
- pour chaque relation de voisinage, deux types de graphe ont été utilisés :
  1) graphes peu denses : `nombre d'arcs = min + (max - min)/3` où `min` dénote le nombre minimal d'arcs, à partir duquel le graphe perd sa connexité, et `max` dénote le nombre maximal d'arcs pour construire un graphe complet,
  2) graphes denses : `nombre d'arcs = max - (max - min)/3`,
- deux type de grilles ont été considérés :
  1) des grilles de type : `4 x k` où `k` varie dans l'intervalle suivant $4 \leq k \leq 8$,
  2) des grilles de type : `5 x k` où `k` varie dans l'intervalle $3 \leq k \leq 7$

1- Graphes peu dense

Nous montrons à l'aide des courbes suivantes les rapports de coûts moyens `Testbed/SPT` (par cercles) et `MSPST/SPT` (par rectangles) pour des graphes peu dense. Deux types de grille sont utilisés : 4 x k pour le premier colonne et 5 x k pour le deuxième colonne. Nous rappelons que le coût d'un arbre est le nombre de branches utilisées (qui est égal au nombre de nœuds moins `1`). Et donc, minimiser ce coût est équivalent à minimiser le nombre de nœuds intermédiaires entre la source et les destinataires.



(a) 4 voisins et une grille 4 x k

(b) 4 voisins et une grille 5 x k

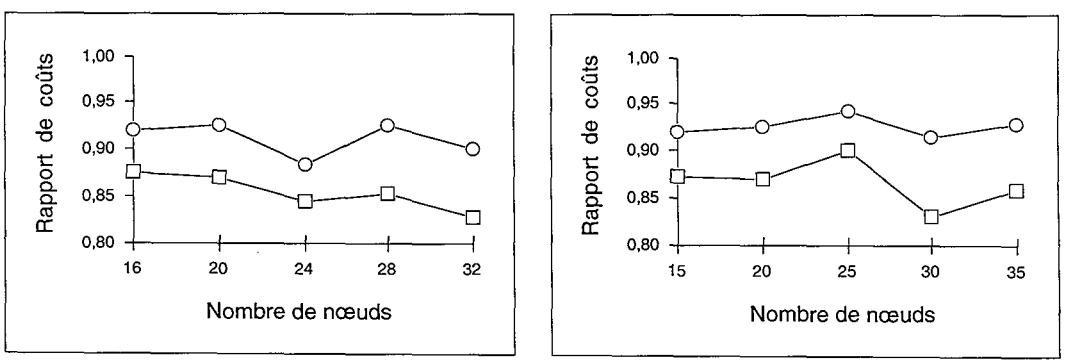
(c) 8 voisins et une grille 4 x k

(d) 8 voisins et une grille 5 x k

Fig. 6.9 — La performance pour des graphes peu denses

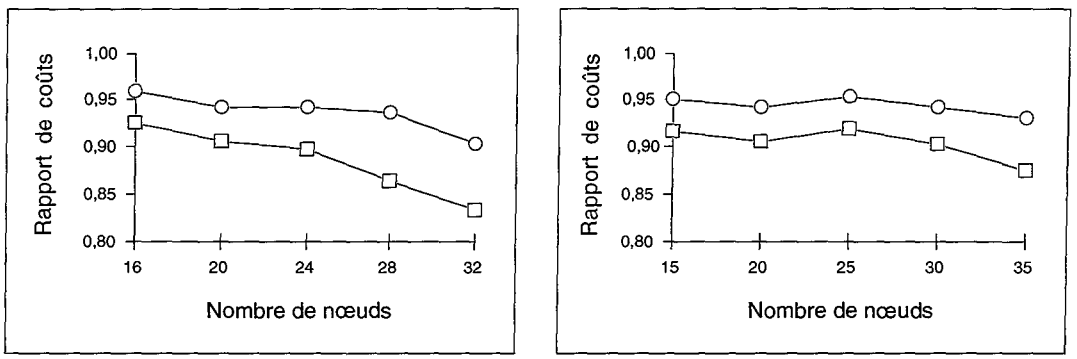
2- *Graphes denses*

Nous montrons à l'aide des courbes suivantes les rapports des coûts moyens `Testbed/SPT` (dénoté par cercles) et `MSPST/SPT` (dénoté par rectangles) pour des graphes denses :



(a) 4 voisins et une grille 4 x k

(b) 4 voisins et une grille 5 x k



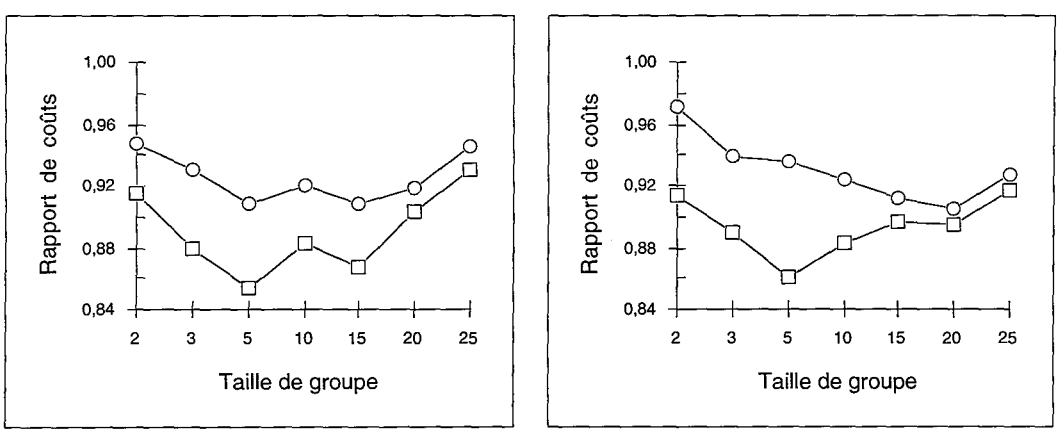
(c) 8 voisins et une grille 4 x k (d) 8 voisins et une grille 5 x k

Fig. 6.10 — La performance pour des graphes denses

Nous remarquons que le gain devient plus important dans le cas des graphes denses parce que les alternatives pour les plus courts chemins deviennent plus nombreuses.

6.5.3.2. La performance en fonction de la taille de groupe

Nous fixons maintenant les dimensions de la grille à 8x6 nœuds (une taille de 50 nœuds est la plus utilisée dans ce type de graphe) et nous faisons varier la taille de groupe de destinataires entre 2 et 25 (la moitié de la taille du graphe) pour les deux cas suivants :



(a) Cas de 4 voisins (b) Cas de 8 voisins

Fig. 6.11 — La performance en fonction de la taille de groupe de destinataires

Nous remarquons que le meilleur gain se produit pour des tailles de groupes situées entre 3 et 15 pour le cas de 48 nœuds. Nous justifions ces résultats par le fait que l'optimalité d'une solution dépend de deux facteurs antagonistes :

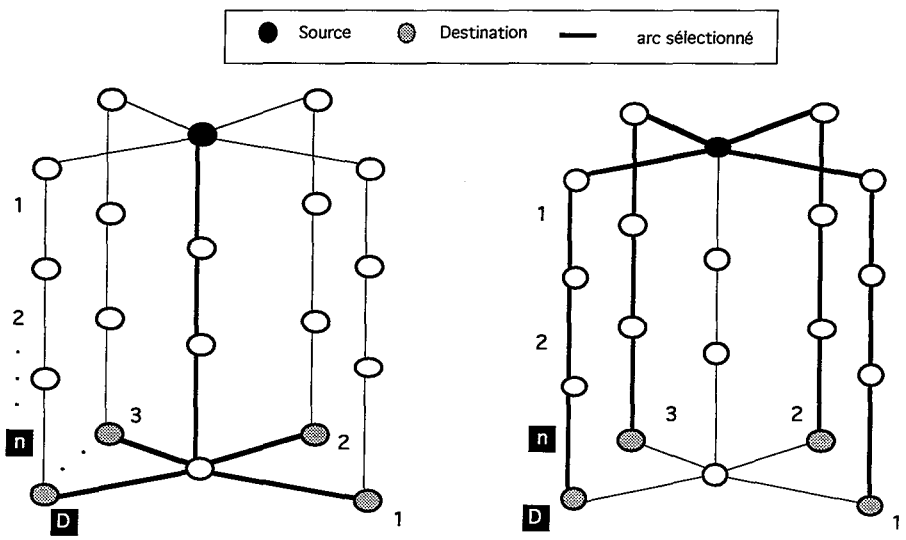
1) le nombre de nœuds intermédiaires. Un nœud est dit intermédiaire s'il n'appartient pas au groupe de destinataires et s'il est important pour la solution finale. Par conséquent, plus la taille de groupe est grande moins il y a de choix pour les nœuds intermédiaires,
2) le nombre d'alternatives (arbres) possibles des plus courts chemins. Ce nombre est par contre proportionnel à la taille de groupe.

Donc, pour une taille de groupe de 2 nœuds, les nœuds intermédiaires potentiels sont très nombreux mais le nombre des arbres de plus courts chemins est réduit alors que pour une

taille de groupe de 25 nœuds, le nombre des nœuds intermédiaires est réduit alors que celui des alternatives (arbres) est grand.

6.5.4. Le cas de plus mauvaise performance

Bien que les courbes précédentes révèlent des gains en nombre d'arcs (relais) qui peuvent atteindre 20% en comparaison avec SPT et de 12% pour le Testbed, il nous semble utile de montrer des cas où ce rapport peut dépasser largement ces limites. La figure suivante indique une distribution spéciale des nœuds et du groupe pour laquelle le rapport est proportionnel avec la taille de groupe et au paramètre n.



(a) MSPST (b) Testbed ou SPT

Fig. 6.12 — La plus mauvaise performance de SPT et Testbed

L'arbre des plus courts chemins obtenu par l'algorithme MSPST est d'un coût de n + D alors que l'arbre obtenu par SPT ou Testbed peut avoir un coût qui atteint (n+1)*D dans le pire des cas.

- (n+D)/((n+1)*D) ≈ 1/D pour des grandes valeurs de n par rapport à D ou
- (n+D)/((n+1)*D) ≈ 1/(n+1) pour des grandes valeurs de D par rapport à n

Application:

Pour n=7 et D=15, l'arbre obtenu par MSPST est de coût 22 arcs alors que celui qui peut être obtenu par SPT (ou Testbed) est de coût 120 arcs.

6.6. Exemple d'interconnexion de réseaux FIP

Nous présentons dans cette section un exemple d'interconnexion de cinq bus FIP. L'objectif principal de l'exemple est d'illustrer la construction des tables de routage dans chacun des ponts pour permettre l'acheminement des objets identifiés, des messages point-à-point et des messages multipoint.

Nous montrons dans la figure suivante la topologie d'interconnexion qui a été choisie pour expérimenter nos algorithmes de routage. Cette topologie doit être connue à la configuration et nous ne cherchons pas, par ailleurs, à l'optimiser.

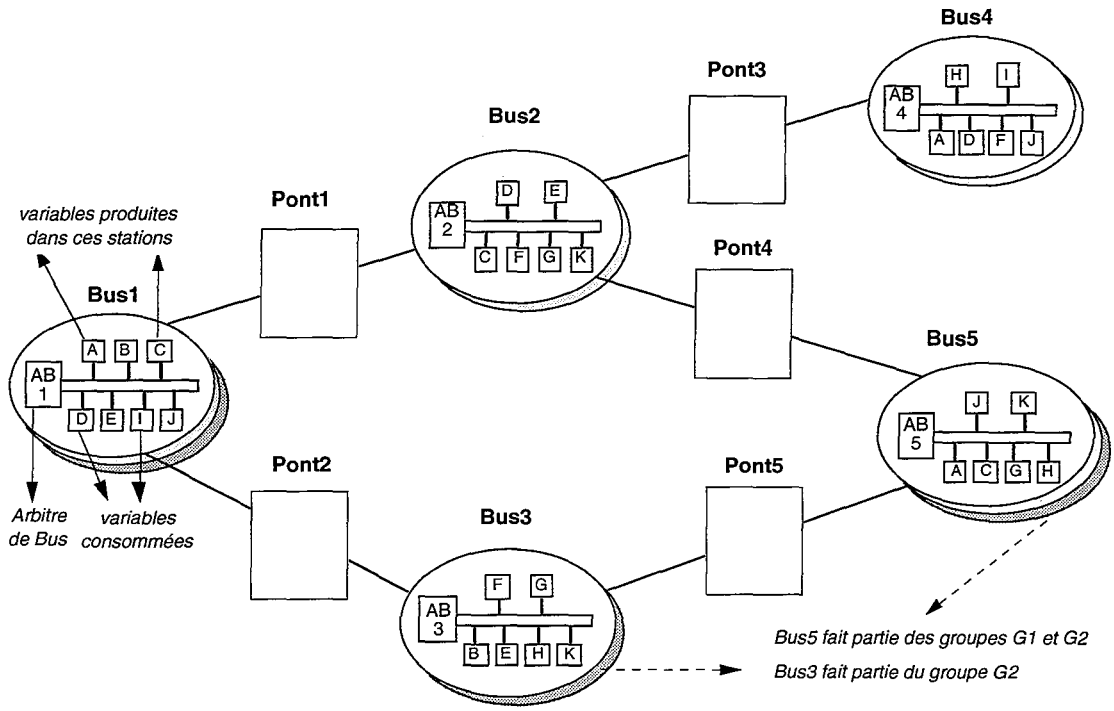


Fig. 6.13 — La topologie d'interconnexion choisie

Il s'agit donc d'interconnecter cinq bus FIP par l'intermédiaire de cinq ponts simples (bi-ports). Chaque bus B(I) se compose de :

1- un arbitre de bus AB(I),

2- un support de communication,

3- des stations raccordées sur le support et gérées par AB(I). Dans chaque station sont configurées les fonctions suivantes :

- un ensemble des variables produites,
- un ensemble des variables consommées; chacune doit être produite dans un bus quelconque du réseau,
- une adresse source composée de l'adresse de cette station dans le bus et du numéro (I) de bus dans le réseau global,
- un ensemble d'adresses de groupes dont cette station est membre : elle doit recevoir une copie de chaque message envoyé à l'un des groupes dont elle fait partie. Il faut bien noter que toutes les stations du réseau global peuvent envoyer un message de groupe mais seulement les membres de ce groupe vont recevoir une copie de ce message. Nous avons choisi le motif clair pour le groupe G1 et le motif foncé pour le groupe G2 (i.e., les membres du groupe G1 sont répartis sur Bus1, Bus2, Bus4 et Bus 5; les membres du groupe G2 sont répartis sur Bus1, Bus3 et Bus5),
- puisque l'interconnexion est concernée uniquement par l'ensemble des variables globales (c-à-d, produites dans un bus et consommées dans d'autres bus)

indépendamment des stations productrices (un bus produit l'ensemble des variables globales produites par ces stations), la figure 6.13 indique seulement ces variables,

4- des demi-ponts raccordés à ce bus. Chaque demi-pont est considéré comme une station mais qui ne possède pas d'adresse source (c-à-d, on ne peut pas adresser un message pour un pont et le pont ne modifie pas les adresses des messages qui y passent). Les deux moitiés d'un pont ont deux accès à deux bus différents et chacune doit donc avoir ses propres tables de variables produites ou de variables consommées, etc. Mais, parce qu'une variable/message consommé dans une moitié sera produit à l'autre moitié, nous allons réduire chaque pont aux tables suivantes : table des variables consommées/produites, table de routage pour les messages point à point et table de routage pour les messages multipoint; nous indiquerons pour chaque table le sens de transfert, i.e. le port de consommation et le port de production pour chaque pont.

Puisque l'objectif principal de l'expérimentation est de construire les différentes tables de routage utilisées dans les différents ponts, nous allons atteindre cet objectif en trois étapes :

- construction des tables de routage des variables,
- construction des tables de routage des messages multipoint,
- construction des tables de routage des messages point à point.

6.6.1. Construction des tables de routage de variables

Nous construisons dans cette étape la table des variables qui traversent chacun des ponts. Ceci nous amène à construire pour chaque variable V l'arbre d'acheminement entre le bus où se trouve le producteur initial et les bus où se trouvent les consommateurs. Les branches de cet arbre dénotent les différents ponts auxquels nous devons ajouter la variable V. Par exemple, pour la variable A qui est produite dans B1 et qui est consommée respectivement dans B4 et B5, l'arbre trouvé par l'algorithme MSPST est constitué des ponts P1, P3 et P4. Nous ajoutons ainsi la variable A à la table de ces ponts en respectant le sens d'échange de cette variable. La table finale de chacun des ponts est donnée dans la figure suivante :

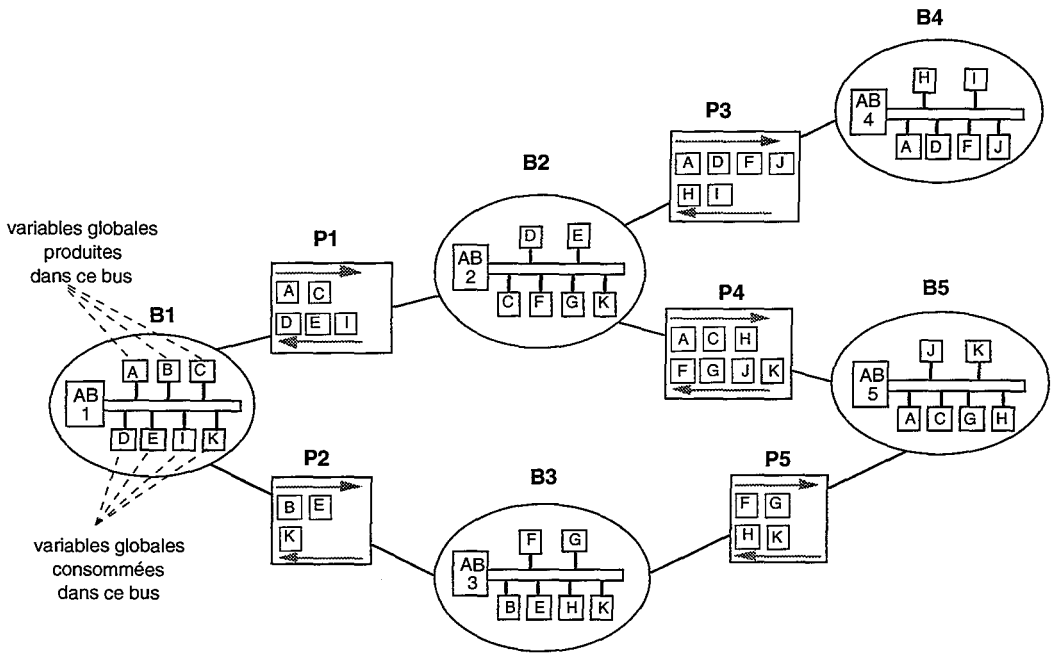


Fig. 6.14 — Tables de routage des variables

En ce qui concerne l'acheminement de chaque variable à travers les ponts, nous avons suggéré deux solutions (voir §4.4.1.) :

La première solution consiste à ajouter les identifieurs des variables globales aux tables de scrutation de chacun des bus traversés (e.g., l'identifieur de la variable A sera ajouté aux tables de scrutation de AB(2), AB(4) et AB(5)). Ensuite, des suites de ID_DAT(ID), RP_DAT(Valeur) vont permettre de transmettre la valeur vers ses destinataires finaux. Pour transmettre la variable A vers ses consommateurs qui se trouvent dans les bus B4 et B5, trois opérations seront initiées :

- une trame ID_DAT(ID(A)) suivie d'une trame réponse RP_DAT(Valeur(A)) permettent au pont P1 de recevoir la valeur de la variable A,
- une trame ID_DAT(ID(A)) suivie d'une trame réponse RP_DAT(Valeur(A)) permettent aux ponts P3 et P4 de recevoir cette valeur,
- une trame ID_DAT(ID(A)) suivie de RP_DAT(Valeur(A)) dans chacun des bus B4 et B5 permettent aux consommateurs finaux de recevoir la valeur de A.

Pour illustrer le fonctionnement des ponts qui sont basés sur la deuxième solution, nous allons considérer le cas des variables A et C :

- une trame ID_DAT(ID(A)) suivie d'une trame RP_DAT(Valeur(A)) diffusée dans B1 permet au pont P1 de recevoir la valeur de la variable A et d'ajouter l'identifieur ID(A) dans une liste L1 (initialement la liste est vide),
- une trame ID_DAT(ID(C)) suivie d'une trame RP_DAT(Valeur(C)) diffusée dans B1 permettent au pont P1 de recevoir la valeur de la variable C et d'ajouter l'identifieur ID(C) dans la liste L1 (L1 contient maintenant ID(A) et ID(C)),
- une trame ID_RQ(ID(L1)) suivie d'une trame RP_RQ([ID(A), ID(C)]) permettent au pont P1 de demander à l'AB2 la diffusion des deux identifieurs précédents,
- une trame ID_DAT(ID(A)) suivie d'une trame RP_DAT(Valeur(A)) permettent aux ponts P3 et P4 de recevoir la valeur de A et d'ajouter ID(A) dans leur listes (L3 et L4),

- une trame ID_DAT(ID(C)) suivie d'une trame RP_DAT(Valeur(C)) permettent au pont P4 de recevoir la valeur de C et d'ajouter ID(C) dans sa liste L4 (L4 contient maintenant ID(A) et ID(C)) et permettent également au consommateur de C situé dans B2 de recevoir la valeur associée,
- une trame ID_RQ(ID(L4)) suivie d'une trame RP_RQ([ID(A), ID(C)]) permettent au pont P4 de demander à l'AB5 la diffusion des identifieurs contenus dans cette liste,
- on continue d'une façon analogue jusqu'à ce que chacun des consommateurs finaux reçoit la valeur désirée.

6.6.2. Construction de tables de routage de messages multipoint

Contrairement au routage de variables où la source est bien connue, la source d'un message multipoint n'est pas connue auparavant parce que n'importe quelle entité, indépendamment de sa localisation physique, peut émettre un message vers un groupe de destinataires. Ceci nous a conduit à considérer chaque bus comme comportant des sources de messages pour tous les groupes. Nous avons à construire, par conséquent, N x M arbres où N est le nombre de bus et M est le nombre de groupes (5x2 arbres dans notre exemple). Il convient de signaler ici que la complexité de cette solution devient proportionnelle au nombre de stations qui composent le réseau global au cas où des adresses logiques (adresse plates) sont utilisées pour désigner la source comme c'est le cas dans les réseaux locaux.

Nous montrons dans la figure suivante les tables de routage multipoint pour les deux groupes G1 et G2 :

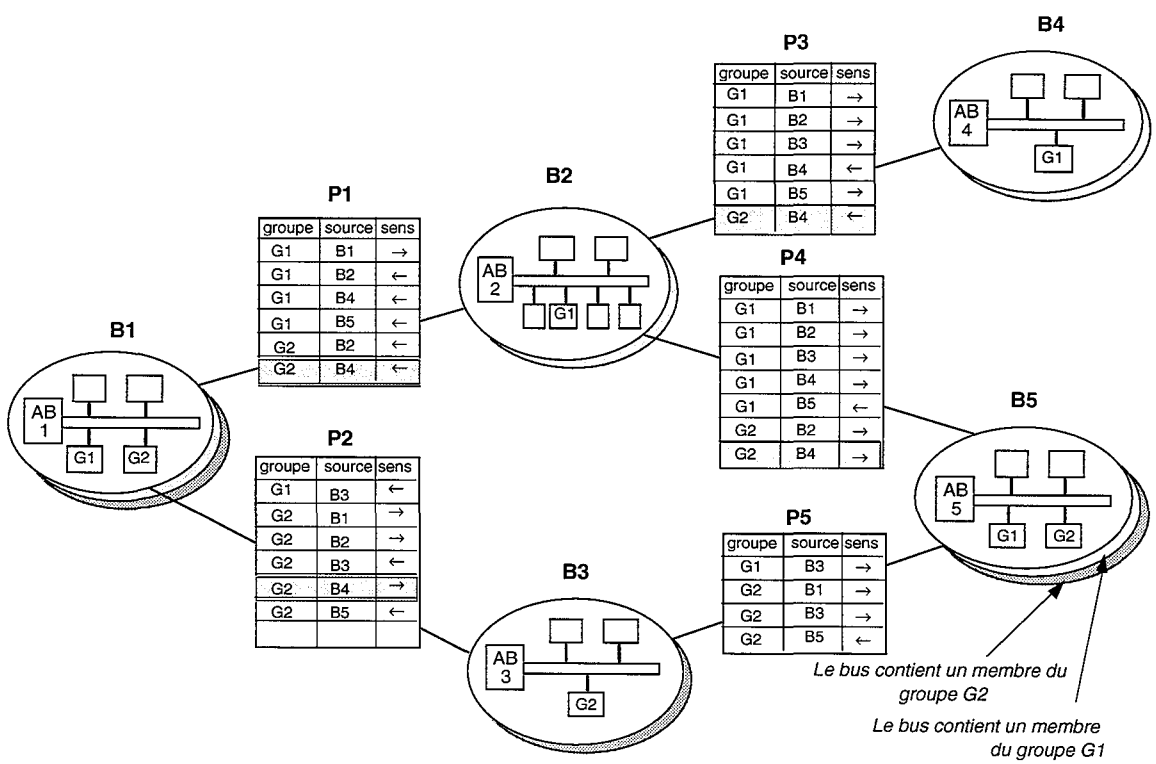


Fig. 6.15 — Tables de routage des messages multipoint

Si une station raccordée à B4 émet un message vers le groupe G2, à savoir les membres de G2 sont partagés par B1, B3 et B5, et que la trame émise contient le suffixe B4 dans l'adresse source, les opérations suivantes sont initiées :

- le pont P3 qui trouve une entrée correspondant au couple (G2, B4) va émettre ce message sur le bus B2 sans modifier l'adresse source,
- chacun des ponts connectés à B2, à savoir P1 et P4, va consulter sa table de routage et trouver une entrée correspondant au couple (G2, B4). P1 et P4 vont donc stocker et transmettre ce message sur les bus B1 et B5.
- les membres du groupe G2 dans B1 et B5 ainsi que le pont P2 reçoivent ce message; le pont P5 ignore ce message car il ne trouve pas une entrée correspondant à (G2, B4),
- le pont P2 transmet le message sur le bus B3.

N.B. : le sens de la flèche qui est utilisée dans chaque table désigne le port de réception et le port d'émission à respecter lors de la consultation de la table.

6.6.3. Construction des tables des messages point à point

Le routage de messages point à point se fait à la base de la portion "numéro de bus" de l'adresse de destinataire. Ceci réduit la complexité de la taille des tables de routage et la rend proportionnelle au nombre de bus.

Nous montrons dans la figure 6.16 les différentes tables de routage qui ont été construites en fonction de la distance minimale en nombre de ponts entre la source et le destinataire :

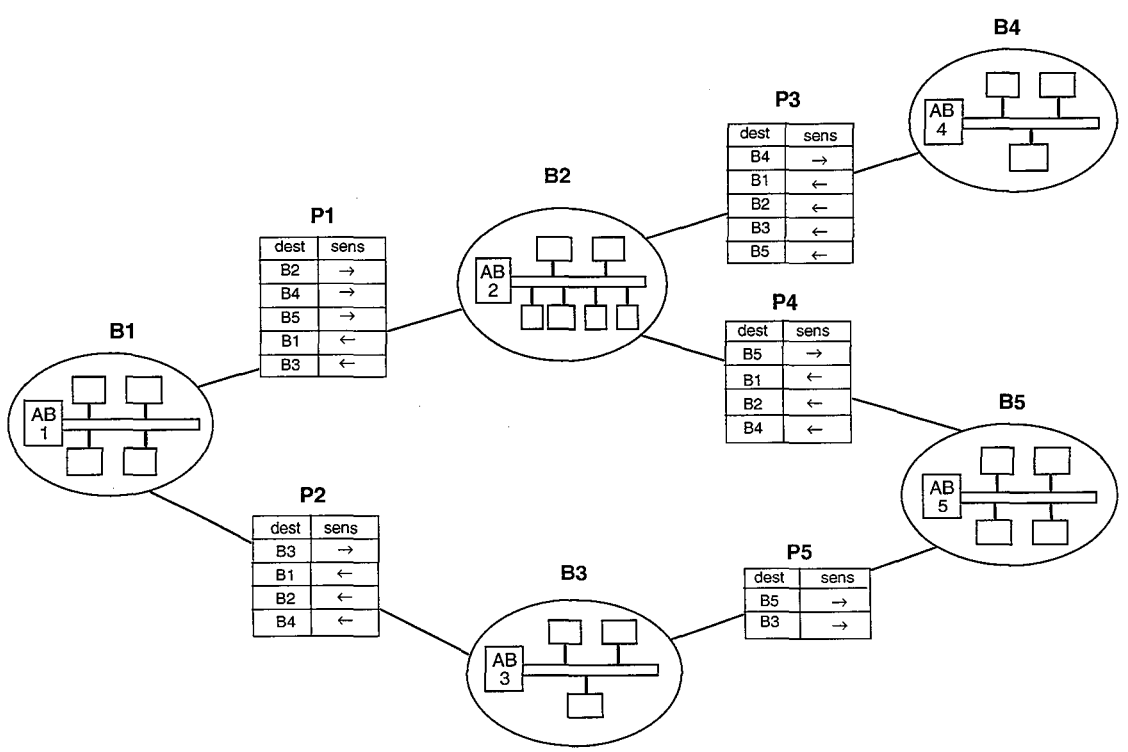


Fig. 6.16 — Table de routage de messages point à point

L'exemple précédent montre que les messages émis par une station appartenant au bus B1 et destinés à des stations qui appartiennent à l'un des bus B2, B4 ou B5 seront acheminés par le pont P1. Le pont P2 achemine seulement les messages de B1 vers B3. En ce qui concerne les messages transmis par les ponts P1 et P2 et traversant le bus B1 (voir le sens

de la flèche dans la table de routage associée à chaque pont), le pont P1 transmet à travers B1 les messages destinés soit au bus B1 lui-même, soit au bus B3; le pont P2 transmet à travers le bus B1 les messages destinés à l'un des bus suivants B1, B2 ou B4.

En conséquence, si une station de B3 émet un message vers une station de B4, les opérations suivantes sont initiées :

- contrairement au pont P5 qui ignore ce message lorsqu'il ne trouve pas d'entrée, au sens de la flèche, correspondant à la destination B4, le pont P2 en trouve une et retransmet ce message sur le bus B1,
- le pont P1 trouve par la suite une entrée correspondant à B4 et retransmet le message sur le bus B2,
- P4 ignorera ce message et ce sera P3 qui le retransmet vers le destinataire final B4.

6.7. Conclusion

Les domaines d'application de l'algorithme que nous venons de proposer peuvent concerner :

- tout protocole utilisant le routage par l'état de liaison où une carte globale de la topologie est fournie à chaque routeur (voir OSPF et IS-IS). Dans ce cas, une construction d'arbre de plus courts chemins doit être exécutée à chaque modification d'un groupe,
- application multimédia et plus particulièrement la vidéo-conférence où les messages (vu ses tailles) doivent traverser le moins de relais possibles tout en respectant des chemins minimaux,
- application aux machines multiprocesseurs. Étant donnée une machine multiprocesseurs (un processeur est représenté par un nœud dans un graphe) et des canaux point à point de communication entre ces processeurs (un canal est représenté par un arc). Le problème d'envoyer un message d'un processeur vers un groupe de processeurs est équivalent à notre problématique.
- application au réseau de terrain FIP.

En ce qui concerne la contrainte du coût qui est supposé égal à l'unité, nous rappelons qu'une transformation d'un graphe avec des coûts entiers en un graphe avec des coûts unitaires peut s'effectuer en remplaçant chaque arc ayant `k` comme coût par `k` arcs ayant un coût égale à `1`.

La deuxième contrainte qui porte sur l'appartenance statique aux groupes entraîne la ré-exécution de l'algorithme à chaque modification d'un groupe. Ceci permet de maintenir un coût minimal de l'arbre qui, dans le cas de mise à jour dynamique de l'arbre, se dégrade rapidement lors de modification d'un groupe.

Conclusion et perspectives

L'objectif principal du travail mené dans ce mémoire était l'interconnexion de réseaux FIP par l'intermédiaire des ponts. Cet objectif nous a amené à traiter le problème par étapes :

Nous avons commencé par un état de l'art consacré à l'interconnexion de réseaux et aux différents algorithmes qui permettent le routage inter-réseaux ou l'extension d'un réseau local. Nous avons montré ainsi comment la généralisation de ces algorithmes, conçus initialement pour le cas d'un seul destinataire, pour le transfert des paquets multi-destinataire s'avère inadaptée pour les réseaux FIP.

Nous avons poursuivi notre démarche en structurant la couche Liaison de FIP pour pouvoir identifier les services rendus par la sous-couche MAC à la sous-couche LLC en fonction des points d'accès aux services SAPs. Ces résultats fournissent un support de base pour la définition de fonctionnalités d'un pont qui agit habituellement au niveau de sous-couche MAC.

Nous avons proposé ainsi quelques solutions permettant de remédier aux spécificités inhérentes du réseau FIP pour former une interconnexion de réseaux par l'intermédiaire des ponts.

Pour accroître la confiance à notre proposition, nous avons utilisé les techniques de spécification formelle. Ces techniques nous ont aidés à valider le comportement du pont et la nouvelle structure de la couche Liaison. Nous avons montré comment les réseaux de Petri étendus par des mécanismes d'étiquetage et par des prédicats peuvent être utilisés comme un langage de spécification formelle. Un point fort dans cette extension revient à préserver toutes les propriétés de RdP en bénéficiant d'autant de techniques d'analyse que les réseaux de Petri élémentaires. Les bases mathématiques de RdP qui leur fournissent une sémantique bien définie nous ont permis d'exclure toutes les ambiguïtés résiduelles dans les spécifications informelles ainsi qu'à valider l'architecture par diverses techniques de vérification.

Il convient de noter que les solutions définies utilisent des tables de routage qui ont été mises à la disposition de chaque pont. Ces tables peuvent être construites à la main lorsque la topologie d'interconnexion est assez simple. Mais, par contre, des problèmes se posent lorsque la topologie devient maillée et fait apparaître des chemins multiples. Pour ce faire,

nous avons développé enfin un algorithme de routage multipoint construisant un arbre des plus courts chemins ayant comme racine une source quelconque et recouvrant le groupe de destinataires. Cet algorithme, dit MSPST, tend également à optimiser le nombre total des ponts (ou des routeurs) traversés (ce qui est équivalent à minimiser le nombre de réseaux passifs qui ne sont pas concernés par les paquets). La performance de cet algorithme par rapport à l'algorithme classique des plus courts chemins SPT (Shortest Path Tree) qui est de loin le plus utilisé dans ce domaine d'application a fait l'objet d'une étude complémentaire.

Quant aux domaines d'application de notre algorithme, ils peuvent se classer dans :

(1) tout protocole utilisant le routage par l'état de liaison où une carte globale de la topologie est fournie à chaque routeur (voir OSPF et IS-IS). Dans ce cas, une construction d'arbre des plus courts chemins doit être exécutée à chaque modification d'un groupe,

(2) les protocoles multimédia et plus particulièrement les vidéoconférences où les paquets (vu leurs tailles) doivent traverser le moins de relais possibles tout en respectant des délais minimaux,

(3) les machines multiprocesseurs : étant donnée une machine multiprocesseurs (un processeur est représenté par un nœud dans un graphe) et des canaux de communication point à point entre ces processeurs (un canal est représenté par un arc). Le problème d'envoyer un message d'un processeur vers un groupe de processeurs est équivalent à notre problématique,

(4) application au réseau de terrain FIP.

Nous sommes conscients que les résultats obtenus dans ce mémoire ne peuvent pas couvrir tous les besoins attendus, que ce soit en matière de routage multipoint ou de services rendus par le pont. C'est pourquoi, les travaux à venir peuvent concerner les deux axes suivants :

(1) Le fonctionnement du pont FIP

Un certain nombre de problèmes restent encore à résoudre pour prendre en compte la globalité des types de trafics offerts par FIP. Il s'agit d'étudier notamment :

- les statuts de rafraîchissement qui sont élaborés à la fois par le producteur initial et par tous les ponts traversés. Nous avons proposé dans [Saba 93] (voir annexe A) une machine à états généralisée qui peut être utilisée au niveau du pont pour qualifier la validité temporelle des données transmises et qu'on peut inclure dans le pont,
- les acquittements pour le trafic de messages en point-à-point,
- les listes de variables produites et/ou consommées sur des réseaux différents,
- la cohérence spatiale et temporelle des listes de variables.

Il s'y ajoute l'interconnexion avec d'autres types de réseaux soit de même niveau (niveau de terrain) comme Profibus ou de niveau supérieur (réseau local) comme Ethernet. Des études quantitatives concernant l'évaluation de performance du pont qui n'est pas abordée dans ce mémoire peut faire également l'objet d'un travail complémentaire.

(2) Le routage multipoint

Nous avons proposé dans le chapitre 6 un algorithme de routage multipoint qui a été initialement forgé pour le cas de FIP et pour lequel nous avons trouvé une diversité d'applications. Mais si on désire adapter cet algorithme à d'autre domaines d'application, les points suivants peuvent être étudiés :

- utiliser un graphe avec des poids associés sur les nœuds qui désignent les sous-réseaux,
- trouver un compromis entre les longueurs des chemins et le coût total de l'arbre,
- avoir une version répartie dans les relais (pont ou routeur),
- traiter le cas des groupes dynamiques pour lesquels une mise à jour de l'arbre d'acheminement doit être effectuée en fonction de modifications des membres des groupes ou pour remplacer un relais défectueux par un autre par exemple,

La prise en compte d'une (ou d'une combinaison) des adaptations précédentes dépend de l'application en question et peut être ainsi effectuée dans une étude à long terme.

Annexe

Space and Time Consistency Qualifications in Distributed Communication Systems

G. SABA, J.P. THOMESSE, Y.Q. SONG

CRIN-ENSEM
2- Avenue de la Forêt de Haye
54516 - Vandœuvre-Lès-Nancy, France
E-mail : {saba, thomesse, song}@loria.fr

Abstract

In this paper, we outline first some factors that characterise the communication quality in a distributed system with respect to strict timing constraints. We recall then that actual standards and models such as Client/Server one are insufficient. Thus, we introduce the PDC model which defines time-critical information broadcasting and attributes time and space consistencies to this information. For this model, some enrichments have been done by adding some general attributes. A new formalism: inequalities' representation, rather than the traditional evaluation nets is used to describe and qualify adequately the notion of time. Our propositions have been validated using MEC, a verification tool for internal behaviour of a distributed system.

Keywords: Time Consistency, Space consistency, General Promptness, Distributed System, Timing Constraints, Producer/Distributor/Consumers model, Time-Critical communication

1. Introduction

Recent applications tend to be more and more distributed, and require increasingly the respect of strict timing constraints. These constraints are not handled sufficiently in the contexts of International Standards. These lead us to study and develop some attributes which characterise the communication quality in a time constrained distributed system. Some of them are defined in the PDC[1] [1,2] model that supports Time-Critical information broadcasting/multicasting and takes into account time and space consistencies related to exchanged information.

To achieve this end, we define the following terms as they appear into French Standards [3] according to the requirements specified by the ISO[2] [4]:

time window: This is the application requirement that is characterised by absolute minimum and maximum values which are dependent on the application and the implementation of the application,

Application Entity (AE): This is the part of the application process that deals with the communication system,

Time-Critical data transmission: This notion of time is used in the data transmission that have constraints related to time window,

Space Consistency: This property indicates whether or not all the copies are identical within a given time window,

Temporal (or time) Consistency: This property when related to a variable indicates whether the value of this variable has been produced and/or transmitted and/or received in a given time window or not.

The reminder of this paper is organised as follows: in the section two, we outline some factors and parameters that seem essential for a time-critical model. We introduce in section three the PDC model adopted by the FieldBus FIP [5] and French Standards [6] which define the broadcasting protocols supporting time-critical communications [7]. This model serves as a basis of all studies realised in the section four that consist of developing, redefining, or extending existing attributes by means of more general attributes or a representation by inequalities.

---

[1] Producer/Distributer/consumers,

[2] International Organization For Standardization,

2. Factors and parameters related to a time-critical model

This section identifies time-critical models, in which not only safety constraints are to be met but also data validity with respect to time intervals is to be guaranteed. Thus, We can only attribute the term Time-critical to such a model when it has the following criteria:

- capability to associate to one or to a set of events, latest start date and deadline,
- capability to distinguish between periodic and sporadic (*event is said to be sporadic when minimum period between any two aperiodic events is defined*) information; and to attribute then periods to periodic ones,
- capability to distinguish Time-Critical information from non-Time-critical ones,
- ability to determine the respect of the time bounds whether on the production phase or on the transmission phase,
- check the feasibility of meeting timing requirements [8],
- attributing properties to information that are locally available on different communication entities,
- attributing of space consistency properties to information that are available on a set of entities so-called consumer entities,
- capability to recovering from errors and to derive the system to some stable states.

Those factors don't appear in the most existing models and more particularly those which are standardised such as Client/Server one adopted by MMS[3] services set which make part of International Standards OSI of ISO [9].

However, these standards have identified three basic time parameters associated with the communication procedures. These are send time, transfer time, and receive time. Moreover, three optional time parameters could be associated depending on the QoS (*Quality of Service*). These are acknowledge time, confirmation time without acknowledge, and confirmation time with acknowledge.

- *Send time* T_S*:* This is the time taking from the transmitter requesting a transfer to the first item of information being placed on the transmission media. It is the message handling time at the sending Entity.
- *Transfer time* T_t*:* This is the time taking from the first item of information being placed on the transmission media to the last item of information being received by the receiver's communication system. It is the time taking to transmit a message across the network.
- *Receive time* T_r**:** This is the time taking from the last item of information being received by the receiver's communication system to the last item of information being received by the receiver. It is the message handling time at the destination Entity.

By means of these three time parameters, we can define the following parameters that may be useful when transaction constraints related to the selected QoS are to be carried out:

- *Acknowledge time* T_a: This is the time taking from the last item of information being received by the receiver to the acknowledge item being received by the sender· It consists of the message validity testing in the receiver Entity, acknowledge send time, acknowledge transfer time, and acknowledge receive time.
- *Confirmation time without acknowledgment* T_{cna}: This is the time taking from the last item of information being sent by the sender to the confirmation item being received by the sender (initiator of the message exchange in some cases).
- *Confirmation time with acknowledgment* T_{ca}: This is the time taking from the acknowledge item being received by the sender to the confirmation item being received by the requester initiator (*request source*).

To each one of the previous parameter should be assigned a time window. Thus, three principals' time windows can be defined by an application in order to satisfy Time-critical communications constraints; for which the bounds are to be defined according to the latest start date and deadlines for executing actions:

- $\mathbf{TW_w}$**:** Time window in which a message must be sent from one AE to another,

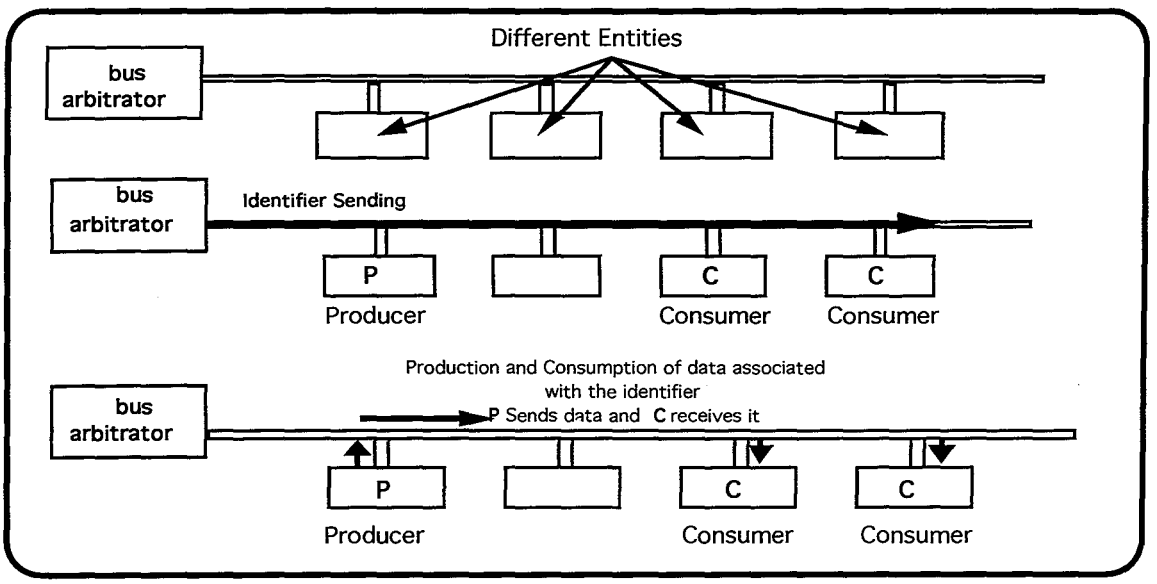
$$\mathbf{TW_w > T_S + T_t}$$

- $\mathbf{TW_r}$**:** Time window in which a message must be sent from one AE to another and an acknowledgment or other response received by the originating AE,

$$\mathbf{TW_r > T_S + T_t + T_r + T_a + T_{ca}}$$

- $\mathbf{TW_t}$**:** Time window in which a Time-critical transaction must be completed.

[3] Manufacturing Message Specification,



Principles of PDC model

3. PDC model

PDC model which is adopted by FIP FieldBus is data broadcasting/multicasting model that guarantees time and space consistencies. The major principle of this model is the following:

- For each information, only one Application Entity is allowed to produce it, it is called "Producer",
- The information can be consumed by one, many, or all other Entities, such entity is called "Consumer",
- Only one Entity is responsible to manage the access right on the communicating medium; it is called here "Bus Arbitrator". This entity designates by sending information identifiers, the successive producers of these identifiers.

Each updated information is subject to general broadcasting to all consumer equipment that are connected to the network. This mechanism for information exchanges can be assimilated to a real-time distributed data base between many sites.

Transmission errors' detection can be realised whether at the data link layer level or at the application layer level. That a life cycle management could be carried out for each received information which will be studied in details later.

End-user, when receiving the information, must know whether this information has been produced in time, transmitted in time, and consumed in time; otherwise production's latest start time and consumption's deadline.

For this end, PDC model provides various mechanisms to check for the production validity (*refreshment*) in parallel with the consumption one (*promptness*) [10]. It allows end-Entity as well to know if all other consumer Entities have the same value of information and, if provided, the same information list closed to time window. This latest is very useful when broadcasting information to many users at the same time.

Bus Arbitrator makes use of time-out mechanisms allowing it to avoid deadlock situations and to bound End-entities medium access time to a predefined delay (*response time should be within this bound*). That ensures fairness access to the Bus (*See* [11] *for more detailed properties*). Since each entity can occupy the bus in the information production phase, and each entity is producer of one information at least, network is starvation free.

4. Metrics for time qualification in distributed system

We have already mentioned that information reception related to time window is our major interest. So, attributes related to information reception will be defined with respect to time window bounds (opening and closing time) in such a way that attributes should take into account the following situations:

— was the information reception within the time window?, before it?, or after it?,

— does time window still open or not?.

Besides these attribute states, absolute aspects related to lifetime of an information (i.e., does last information validity still on?) will be studied.

In the following, we will deal with subset of these attributes (time consistency) whereas the rest of this paper deals with combination of all possible states.

Time consistency as defined into the French standards [3] deal with the absolute aspect rather than a subset of those related to a time window in a separate manner. Thus, actual attributes are the following:

Asynchronous Promptness: When it has the value TRUE, this Boolean information indicates in a consumer entity that the variable value has been updated by the network for a delay that is smaller than the variable consumption period.

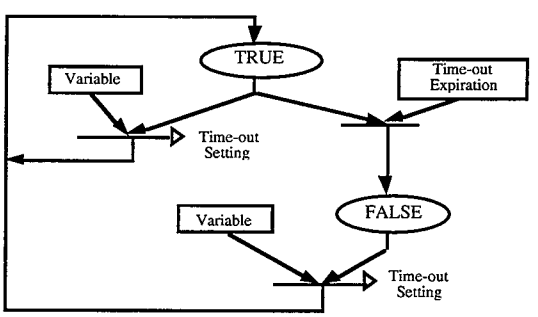


Figure 1 : Asynchronous Promptness

Synchronous Promptness: When it has the value TRUE, this Boolean information indicates in a consumer entity that the synchronisation order has been received and has been followed by (at least) one updating of the associated variable by the network, and that the updating was within the consumption period.

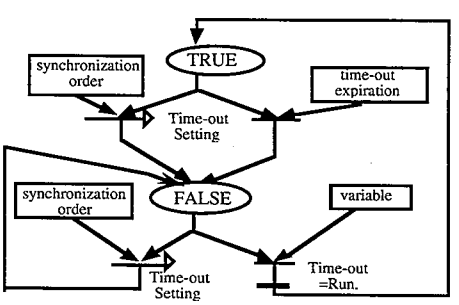


Figure 2 : Synchronous Promptness

Punctual Promptness: When it has the value TRUE this Boolean informs a consuming application entity that the variable value has been updated by the network (at least) once in the time slot following the reception of a synchronisation order associated with the variable. It indicates also that the variable value has not been updated outside the time slot.

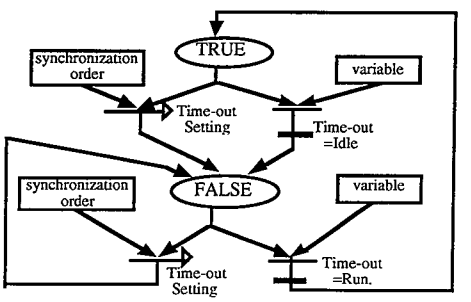


Figure 3 : Punctual Promptness

Previous promptness evaluation nets make use of a *time-out* resource that is given in the following figure:

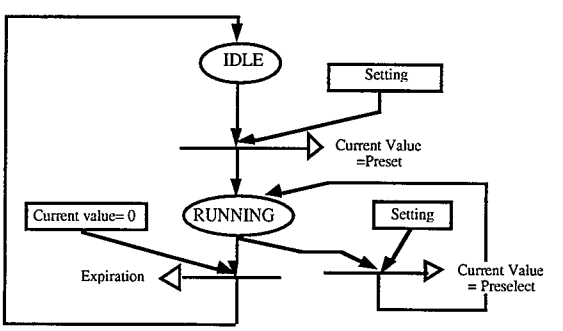


Figure 4 : Time-out

We do not consider here attributes which are relevant to production information (refreshment) because they could be derived easily in an analogue way from promptness when substituting variable reception by variable production.

In order to use more general notations, open time window will be used for synchronisation order, close time window for time-out expiration, and information reception for variable value reception. Note that, old notations will be used when reference to French standards is to be made and others for newly developed.

4.1. Analysing the status of Synchronous and punctual consistencies

It should be noted that the different previous status of promptness (*resp. refreshment*) indicate when they have the value "TRUE" at a consumer entity (resp. producer) that time constraints on transmission (resp. production) have been met. Assuming now that the Boolean value is "FALSE", in this case no hypothesis can be made regarding the reason of this status because the following possibilities could occur:

1) time window was closed without receiving (resp. producing) the information by the consumer (resp. producer) entity,
2) Time window has been opened, information has not been received (resp. produced), and time window has not been closed yet. Promptness status (resp. refreshment) stills waiting whether closing time window or information reception(resp. production); let's call it "IN".
3) Time window has been opened, information has been successively received (resp. produced) one time at least. Promptness status (resp. refreshment) is "TRUE" because all conditions are met.
4) Time window has been opened, closed successively without receiving (resp. producing) information, or information has been received (resp. produced) outside the time window. Which indicates that Promptness (resp. refreshment) is in "FALSE" state.

In the following figure, we focus not only on the information reception but also on the time bounds currently in use. So, two new states are added to reflect that time window stills open "IN" or has been closed "OUT". Moreover, promptness system doesn't affect timer any more because we use the closing time window instead of time-out expiration. In this context, we omit dependency between opening and closing time window and attempt furthermore to give a more general formalism.

In this representation method, "False" state denotes the information reception outside the time window or closing the time window without receiving the information. Whereas the "TRUE" state denotes the reception of the information once at least within the time window. "IN" state stands for an opened time window and "OUT" for a closed one.

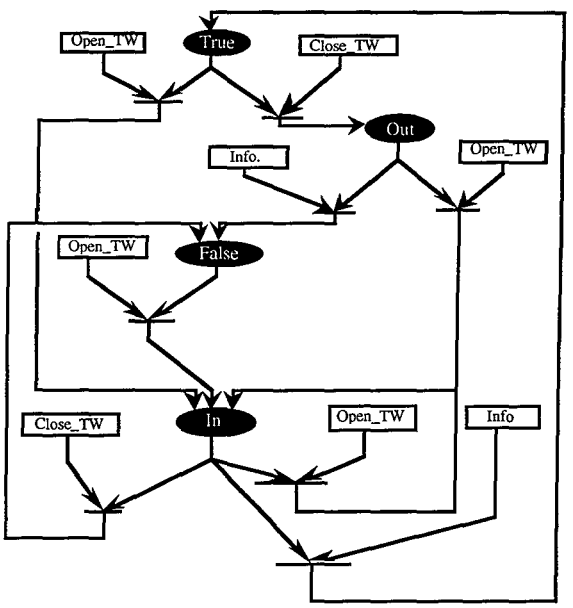


Figure 5: Evaluation net for the General Promptness

From this form of promptness, Synchronous and Punctual states can be derived easily as indicated in the following table:

General	Synchronous	Punctual
IN	FALSE	FALSE
OUT	FALSE	TRUE
FALSE	FALSE	FALSE
TRUE	TRUE	TRUE

Table 1 — General promptness states through synchronous and punctual ones.

Equivalence relationship of table 1 can be validated as follows:

General through synchronous promptness:

- we omit first time-out setting action and rename then time-out expiration by closing time window,
- we substitute in the General promptness "OUT" and "IN" by "FALSE",
- we omit then all internal actions among these three states (OUT, IN, FALSE), and distinguish two kinds of information reception (inside or outside the time window). This distinction is justified by the fact that information reception will be considered as an internal

action outside the time window and as an external one inside it.

• We can use now the strong equivalence according to the Milner's bissimulation [12, 13] to prove the equivalence of the two previous automata (that have been done using AUTO [14, 15] tool that allows us to test such equivalence). Proving accordance with punctual one have been done by an analogous manner but "OUT" state was considered as "TRUE".

4.2. Space Consistency

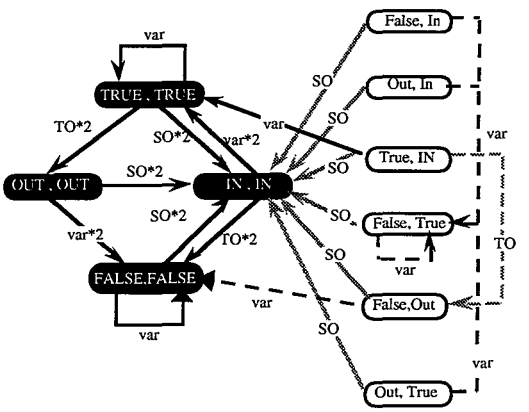
Space Consistency in a distributed system can be now examined by means of the parallel product of all automaton components.

To show now whether all entities belonging to a distributed system have an identical copy of information at a given time window; we need then to compose several automata together to constitute a global system automaton. Thus, Space consistency will be easily obtained when synchronizing and exploring global system as well reachable state analysis. Synchronising and composing several components have been realised using the tool MEC [16, 17] which is a verification tool for internal behaviour of a distributed system by means of searching and exploring reachable states in the global graph. Resulting global graph for two entities with respect to one single information consists of 25 states and 125 transitions. Of course, all errors' possibilities have been introduced and recovered upon, as shown in the figure 4.

In this case, we have adopted space consistency for one information reception with respect to two entities. So, following assumptions have been made: correct transitions are denoted by continuous lines, recovering transitions in shadowed lines and error to error transitions or those which can imply ambiguity in dotted lines. State assumptions are: correct in black and erroneous in white.

N.B.: We have not introduced symmetric states or transitory ones, and some error transitions for simplicity.

1- when no error can produce (i.e., both entities receive the same action at the same time). Thus, global system behaviour can be viewed as one single entity behaviour.



SO = Synchronization Order reception (*open_TW*)
TO = Time-out (*Close_TW*)
var = Variable (*information*) reception
***2** = Two simultaneous receptions

Figure 6: Space consistency for two entities

2- When one error can produce within a fixed time window, we are in the white states, thus, the recovering will be immediately after the reception of synchronisation order; otherwise, we still in the error states with two exceptions:

(True, IN) |— var_r —> (True, True) and (False, Out) |— var_r —> (False, False). These two exceptions are justified by the fact that when the system is on the "False" or "True" states, two successive receptions of variable do not change its value.

3- Deadlock situations can be occurred if and only if no more occurrence of synchronisation order can take place in the system.

Example: If first entity has the value "Out" and the second has "IN"; that's mean second entity has not received the synchronisation order.

Thus, space consistency is equivalent to say that last fired transition was between (IN , IN) and (True , True) by the reception of variable simultaneously in the two entities. Other kinds of consistency can be easily introduced (e.g., the same state, the same true state ,...).

4.3. Using inequalities

In some applications, when time constraints become more stringent; it should be necessarily to deal about exact values of time (e.g., time elapsed since the last information reception).

Because several components (*timer, promptness, buffers)* are to be composed together to build one single

system and this system could be composed with others so far to show global consistency (*N Entities for a list of M information*), makes verification concern more complicated (*e.g., one information with respect to two Entities produces an automaton with 25 states and 125 transitions*). Moreover, we realised that problem becomes more simplified if we add time at the transition level which is the case of many specification tools like EVAL [18, 19] and even some synchronous language like Esterel [20]. That leads us to adopt an other formalism which has the capacity of describing all previous attributes by means of inequalities. We assume here the existence of a global conceptual clock [21] that is the basis of all measurements. Each entity can read global current time with some level of accuracy. So, we can now introduce the following notations:

$\mathbf{T_c}$: denotes current time (i.e., time giving by the global clock),

$\mathbf{T_r}$: denotes the time at which information has been received (local to an end-entity),

$\mathbf{T_o}$: denotes the time at which time window has been opened (local to an end-entity),

$\mathbf{W_1}$: denotes the time window length which is an alternative of using closing window, and

$\mathbf{C_p}$: Consumption period related to the information (information lifetime).

Considering the above notations, we define two variables:

$\theta_1 = T_r - T_o$: it denotes the time being spent between the reception of last information and the time at which window was opened, and

$\theta_2 = T_c - T_o$: which stands for progressing delay since the beginning of the time window to current time.

The following invariants hold always:

$$T_c \geq T_o$$
$$T_c \geq T_r$$
$$\text{So } \theta_2 \geq \theta_1.$$

In terms of the previous variables, we can define promptness status as follows:

Asynchronous promptness: has the value "True" whenever $t_c - t_r < C_p$ that implies:

$$\theta_2 - \theta_1 < C_p$$

Synchronous promptness has the value "True" when

$$0 < \theta_1 < W_1 \text{ and } \theta_2 < W_1$$

Punctual promptness has the value "True" when

$$0 < \theta_1 < W_1$$

Depending on the value of θ_1 and θ_2 the following inequalities can be pointed-out:

1)	$\theta_1 < 0$	and	$\theta_2 < W_1$	"IN",
2)	$\theta_1 < 0$	and	$\theta_2 > W_1$	"FALSE",
3)	$0 < \theta_1$	and	$\theta_2 < W_1$	"TRUE",
4)	$0 < \theta_1 < W_1$	and	$\theta_2 > W_1$	"OUT",
5)	$\theta_1 > W_1$			"FALSE".

Note that condition (3) denotes the synchronous promptness, both (3) and (4) denote punctual promptness, and all together denote General promptness. These states are shown in the next figure and map all the definition space in terms of θ1 and θ2.

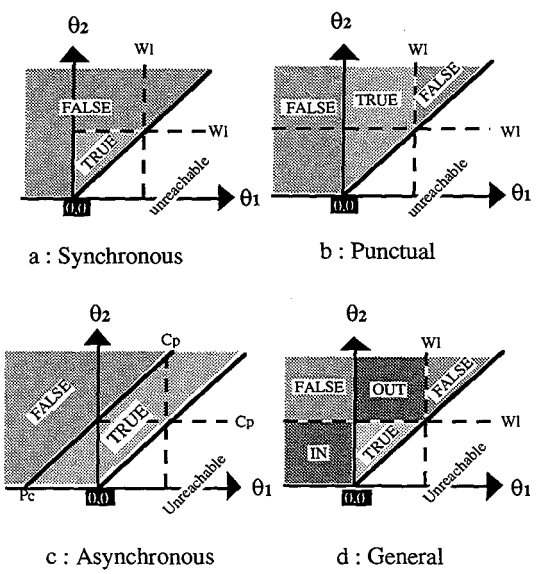


Figure 7: Graphical representation of promptness status

This form of representation can inform several properties:

— That General promptness = TRUE implies that Asynchronous one =True if $W_1 \leq C_p$

— that we can always deduce value of synchronous or punctual promptness from the value of General one,

— that the two regions where General value is "False" could be distinguished as information reception beyond the time window and the time window closing with non reception of information.

In this representation, we consider the time issue (*in the two axes*) as fundamental and all actions are to be considered according to it. Thus, considering the previous graphics, we can deduce easily automata related to each promptness taking into account the following points:

1- Time progressing ($T_C := T_C + n$) is mapped by moving in vertical sense,
2- information reception ($Tc = T_r$, thus $\theta_2=\theta_1$) is mapped by the projection on the line $\theta_2=\theta_1$, and
3- time window opening ($T_C=T_O$, thus, $\theta_1:=\theta_1-\theta_2$ and $\theta_2:=0$) is mapped by the projection on the abscise axis θ_1.

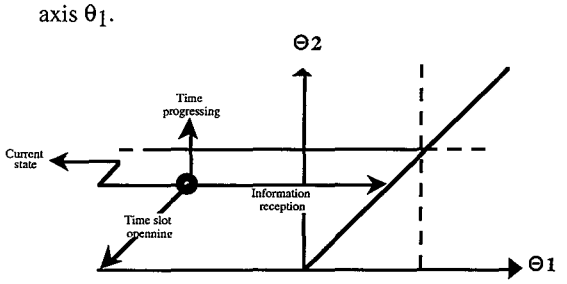


Figure 8: actions simulation in the graphical representation

With these assumptions all previous promptness status can be drawn, verified, and relationships between them can be pointed-out. Furthermore, asynchronous and General promptness can be fold together to give an eight, nine, or ten states' automaton depending on the values of W_1 and C_p.

According to these rules, ambiguity could be avoided and tracing states' machines could be easily done. More particularly, an explicit use of time bounds was used and more significant values could be obtained.

5. Conclusion

In this paper, we have shown that current models and standards for dealing with timing constraints are too ad hoc; we have introduced hence an adequate model PDC that serves as a basis of all suggestions. Thus, General form of promptness has been proposed which improves interoperability between remote entities when negotiating Space consistency, give more indications regarding information reception, and makes an explicit use of time window's notion. The last section has been devoted to give real amounts of time whether locally to each entity or to be circulated to test Space consistency. Henceforth, depending on the importance of the circulated information, the attribute will be negotiated.

The result of this work could be useful when constructing a time constrained distributed system. More particularly in a real time industrial systems in which the safety depends not only on the correct execution of the functions but also on the time at which results have been produced.

References,

[1] J.P. Thomesse, P. Noury, "Client-server model versus, Producer-distributor-consumers model".Contribution to ISO/TC184/SC5/WG2 - TCCA working group, October 1989.

[2] J.P. Thomesse, "Les services du bus de terrain FIP et l'intégration dans les systèmes automatisés". Congrès CIM'90. Bordeaux, 12-14 June 1990.

[3] Norme FIP de la "couche application", Union Technique de l'Electricité, Courbevoie.

[4] Interim report of the TCCA Rapporteurs group of ISO/TC 184/SC 5/WG2 on time-critical Communications Architecture and System, ISO/TC 184/SC 5/WG 2 N254, 24 April 1991.

[5] J.P. Thomesse, P. Lorenz, "Fieldbus - Communications models - Real Time Data Base", ANIPLA, Milano, 6 juin 1991.

[6] "Bus FIP pour échange d'informations entre transmetteurs, actionneurs et automates. Couche Application. Services Périodiques et Apériodiques" Norme Pr C 46-602- Union Technique de l'Electricité, avril 1990.

[7] P. Chatelet, T. Laine : "Les performances temps critique du réseau de terrain FIP", RGE - N° 2/91 - Février 1991.

[8] André M. van Tilborg, Gary M. Koob: "Foundations of Real-Time Computing, Formal Specifications and Methods", Kluwer academic publishers, Chapter 10, pp. 251-290, 1991.

[9] ISO 9506: "Manufacturing Message Service: Industrial Automation Systems - Systems Integration and Communications - Manufacturing Message Specification, Part 1: Service Definition, Pub. 88, 2) Protocol Specofocation, Part 2, Pub. 88.

[10] J.-P. Elloy : "Les contraintes du temps réel dans les systèmes industriels répartis", RGE - N° 2/91 - Février 1991.

[11] M. Sajkowski: "Protocol Verification Techniques: Status Quo and Perspectives", Proceedings of the IFIP WG 6.1 Fourth International Workshop on Protocol Specification, Testing, and Verification Skytop Lodge, Pennsylvania, U.S.A., June 11-14, 1984, pp. 697-720.

[12] R. Milner, "A calculus of communicating systems", Lecture Notes Computer Science 92, Springer-Verlag, 1980.

[13] R. Milner, "Calculi for Synchrony and Asynchrony", Theoretical Computer Science 25, pp267-310, 1983.

[14] Valérie Lecompte, Eric MADELINE, Didier Vergamini : "un systeme de Vérification de processus paralleles et communicants", AUTO, Rapp. Tech. INRIA, N° 83, Mars 1987.

[15] Robert de SIMONE, Didier VERGAMINI:"Programmation, Calcul Symbolique et Intelligence Artificielle", ABOARD AUTO, Rapports Techniques-INRIA , N° 111, Octobre 1989.

[16] P. Crubille: "Réalisation de l'outil Mec : Spécification Fonctionnelle et Architecture" Thèse, à l'univérsité de Bordeaux I, nov, 1989, N° d'ordre 390.

[17] A. Arnold, "Mec: a system for constructing and analysing transition system", Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, 1989.

[18] K. Jensen (Ed.): "Application and Theory of Petri Nets 1992", 13th International Conference Sheffield, UK, June 1992, Proceedings, pp. 379-383.

[19] Verilog 91 EVAL: System Modelling and Analysis. Technical Presentation, EVAL/NT.A/1.0, Verilog, Toulouse.

[20] G. Berry, P. Couronne, G. Gonthier : "Synchronous Programming of Reactive Systems : an Introduction to Esterel", INRIA, Rapports de Recherche, Unité de Recherche INRIA-Sophia-Antipolis, N° 647, mars 1987.

[21] André M. van Tilborg, Gary M. Koob: "Foundations of Real-Time Computing, Formal Specifications and Methods", Kluwer academic publishers, Chapter 9, pp. 217-250.

Bibliographie

[Amer 89] P.D. Amer and F. Cecli. Estelle Formal Specification of ISO Virtual Terminal. *Computer Standards and Interfaces,* pp. 87-104, North-Holland, 1989.

[Arnold 82] A. Arnold and M. Nivat. Comportements de processus, *Colloque AFCET*, Les mathématiques et l'informatique, pp. 35-68, 1982.

[Backes 88] F. Backes. Transparent Bridges for Interconnection of IEEE 802 LANs. *Special issue: "Bridge and Routers", IEEE Network,* 2(1):5-9, Jan. 1988.

[Ballardie 94] A. J. Ballardie, P. F. Francis, and J. Crowcroft. Core based trees. *Proceedings of the ACM SIGCOMM 94*, London, September 1994.

[Beineke 78] L. W. Beineke and R. Wilson. *Selected Topics in Graph Theory*, Academic press, 1978.

[Berntsen 88] J. Berntsen, J. R. Davin, D. A. Pitt and N. G. Sullivan. MAC LAYER. *Computer Network and ISDN Systems,* (10):259-273, December 1985.

[Bharath 83] K. Bharath-Kumar and J. M. Jaffe, "Routing to multiple destinations in computer networks," *IEEE Trans. comm.*, vol. COM-31, pp. 343-351, March 1983.

[Blyth 87] D. Blyth, E. Dubuis, H. Hansson, G. Juanol, M. Kapus-Kolar, H. Kemer, G. Leduc, G. Le Moli, A. Lombardo, S. Marchena, W. Orth, J. Pavon, B. Pehrson, M. Tienari and F. Vogt. Architectural and Behavioural Modelling in Computer Communication, *COST 11 ter group on FDT/ABM Project,* Juillet 1987.

[Bowen 93] J.P. Bowen and V. Stavridou. Safety-Critical Systems, Formal Methods and Standards. *Software Engineering Journal*, 8(4):189-209, July 1993.

[Bowen 94a] J.P. Bowen and M.G. Hinchey. Seven More Myths of Formal Methods. *Technical Report PRG-TR-7-94*, Programming Research group, Oxford University Computing Laboratory, June 1994.

[Bowen 94b] J.P. Bowen M.G. Hinchey. Ten Commandments of Formal Methods. *Technical Report*, Programming Research group, Oxford University Computing Laboratory, 1994.

[Bruno 92] G. Bruno, A. Castella, G. Macario and M. Pescarmona. Scheduling Hard Real Time Systems Using High-Level Petri Nets. In K. Jenson, editor, *Application and Theory of Petri Nets*, pp. 93-112, Lecture Notes in Computer Science, Springer-Verlag, 1992.

[Burg 89] F. M. Burg and N. Di Iorio. Networking of Networks: Interworking According to OSI. *IEEE Journal on Selected Areas in Communications*, 7(7):1131-1142, September 1989.

[Cali 94] D. Cali, G. Zany. *Technologie de l'interconnexion de réseaux.* Techniques de l'information, Hermès Paris, 1994.

[Cantrell 92] P. E. Cantrell. Gateways between OSI and Proprietary Networks. *ISA*, pages 1425-1430, 1992.

[Castner 92] S. Castner. Second IETF internet audiocast. *Internet Society News*, 1(3):23, 1992.

[Comer 92] D. Comer. TCP/IP : *Architecture, Protocoles, applications*. InterEditions 1992. (Version originale: Internetworking with TCP/IP, volume 1: Principles, Protocols, and Architecture, Second Edition, Prentice Hall, Inc. 1991).

[Corr 91] P. Corr. Multi-LAN Approach to LAN/WAN Internetworking. *International Journal of Network management*, :22-38, September 1991.

[Courtiat 87] J.P. Courtiat, P. Dembinski, R. Groz and C. Jard. Estelle: un langage ISO pour les algorithmes distribués et les protocoles. *Technique et Science Informatiques*, 6(2):89-101, 1987.

[Courtiat 89] J.P. Courtiat, M. Diaz and M. Ayoub Dit Ayadi. Description Formelle de Protocoles OSI en Estelle. *Séminaire Framco-Brésilien*, pp. 47-55, Floianopolis-SC, Brésil, August 1989.

[Creusot 91] D. Creusot, J.-P. Elloy, S. Kachkachi, J.-P. Thomesse. Interconnexion de réseaux: FIP-FIP et FIP-MAP, *Actes du Colloque A2RP*, MRT, Paris 15-16 Jan. 1991.

[Crubille 89] P. Crubille. *Réalisation de l'outil Mec : Spécification Fonctionnelle et Architecture*, Thèse de Doctorat, N° d'ordre 390, univérsité de Bordeaux I, novembre 1989.

[Dalal 78] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040-1048,1978.

[Deering 90a] S. Deering. Multicast Routing in Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2),1990.

[Deering 90b] S. Deering and D. Cheriton. Multicast Routing in datagram internetworks and extended LANs. *ACM Transactions on Computer Systems*, pages 85-111, May 1990.

[Deering 91] S. Deering. *Multicast routing in a datagram internetwork.* PhD thesis, Stanford University, Dec. 1991.

[Deering 94] S. Deering, D. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei. An architecture for wide-area multicast routing. *Proceedings of the ACM SIGCOMM 94*, London, September 1994.

[Diaz 86] M. Diaz. Petri Net based models in the specification and verification of protocols. In G. Goos, J. Hartmanis, editors, Lecture Notes in Computer Science, pages 135-170, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Springer-Verlag, Bad Honnet, September 1986.

[Dijkstra 59] E. W. Dijkstra, "A note on two problems in connection with graph," *Numerishe Mathematik*, 1, pp. 269-27, 1959.

[Dixon 88] R. C. Dixon, D. A. Pitt. Addressing, Bridging, and Source Routing. *Special issue: "Bridge and Routers", IEEE Network*, 2(1):25-32, Jan. 1988.

[Elloy 90] J.-P. Elloy. Les contraintes du temps réel dans les systèmes industriels répartis. *Journée SEE Grenoble*, Juin 90.

[Faure 90] C. Faure. *Interconnexion de réseaux informatiques et réseau numérique à intégration de services : modélisation et vérification d'architecture de communication.* Thèse de l'université Paul Sabatier de Toulouse, 1990.

[Fip 88] Club FIP - LAAS, Couche liaisons de données FIP — Modèle en Réseaux de Petri Labellés à Prédicats, *Dossier Technique*, Octobre 1988

[Floyd 62] R. W. Floyd, "Shortest path Algorithm 97, *CACM5*, 345, 1962.

[Gabrielian 91] A. Gabrielian and M. Franklin. Multilevel specification of real-time systems, *Comm. of ACM* 34, pp. 51-60, May 1991.

[Galara 84] D. Galara et J.P. Thomesse. *Proposition d'un système de transmission série multiplexé pour les échanges d'informations entre des capteurs, des*

actionneurs et des automates réflexes, Ministère de l'Industrie et de la recherche, 1984.

[Genrich 86] H. J. Genrich. Predicate/Transition nets. In G. Goos, J. Hartmanis, editors, *Lecture Notes in Computer Science*, pp. 207-247, Petri Nets: Central Models and their Properties, Springer-Verlag, Bad Honnet, 1986.

[Gilbert 68] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees," *SIAM J. Appl. Math.*, vol. 16, No. 1, pp. 1-29, 1968.

[Grant 92] K. Grant. Users requirements on time Critical Communications Architectures. *Technical Report ISO TC184/SC5/WG2/TCCA*. April 1992.

[Groz 85] R. Groz, C. Jard and C. Lassudrie. Attacking a complex distributed algorithm from different sides: An experience with complementary validation tools. In Y. Yemini, R. Strom and S. Yemini, editors, *Protocol Specification, Testing, and Verification,* IV, pp. 3-17, IFIP, Elsevier Science, North-Holland, 1985.

[Harel 87] D. Harel. Statecharts: a visual formalism for complex systems. *Science of Computer programming*, Vol. 8, 1987.

[Heuser 92] C.A. Heuser and G. Richter. Constructs for Modeling Information Systems with Petri Nets. In K. Jensen, editor, Application and Theory of Petri Nets, pp. 224-243, *Lecture Notes in Computer Science*, Springer-Verlag, 1992.

[Hoare 85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall International, 1985.

[Huitema 95] C. Huitema. *Le routage dans l'Internet*, Editions Eyrolles, 1995.

[Jensen 86] K. Jensen. Coloured Petri Nets. In G. Goos, J. Hartmanis, editors, *Lecture Notes in Computer Science*, pp. 248-299, Central Models and Thier Properties, Springer-Verlag, Proccedings of an Advanced Course, Bad Honnet, September 1986.

[Juanole 90] G. Juanole and M. Ayoub Dit Ayadi. ISO CCR Formal Modelling and Verification via Projections. In M. Cosnard and Girault, editors, *Decentralized Systems,* pp. 387-398, IFIP, Elsevier Science, North-Holland, 1990

[Kachkachi 92] S. Kachkachi. *Interconnexion de réseaux FIP simulation d 'un pont.* Thèse, INPL-ENSEM, Nancy, April 1992.

[Karp 72] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations,* R. E. Miller and J. W. Thatcher, Eds., Plenum Press, New York, pp. 85-104, 1972.

[Kohavi 70] Z. Kohavi. *Switching and Finite automata Theory.* 1970.

[Kou 81] L. Kou, G. Markowski, and L. Berman, "A fast algorithm for Steiner trees" *Acta Informatica*, vol. 15, pp. 141-145, 1981.

[Lecompte 87] V. Lecompte, E. Madeline, D. Vergamini : un systeme de Vérification de processus paralleles et communicants, AUTO, *Rapp. Tech. INRIA*, N° 83, Mars 1987.

[Lee 87] I. Lee and S.B. Davidson. Adding Time to Synchronous Process Communications. *IEEE Transactions on Computers,* 36(8):941-948, September 1987.

[LeLann 91] G. Le Lann. Designing real time dependable distributed systems, *Rapport INRIA,* N° 1425, Avril 1991.

[Lloret 90] J.C. Lloret. *Réseaux Prédicat/Transition Etiquetés pour la modélisation et la vérification de systèmes informatiques répartis.* Thèse de doctorat, LAAS du CNRS, Toulouse, France, July 1990.

[Lloret 92] J. C. Lloret and J.L. Roux. Modelling and Evaluation of a Satellite System Using EVAL, a Petri Net Based Industrial Tool. In K. Jensen, editor, Application and Theory of Petri Nets, pp. 379-383, *LNCS*, Springer-Verlag, 1992.

[Mammeri 93] Z. Mammeri et J.P. Thomesse, *Réseaux locaux industriels - FIP et MAP dans les systèmes automatisés.* Editions Eyrolles, 1993.

[Merlin 76] P. Merlin and D.J. Faber. Recoverability of communications protocols. *IEEE Trans. Comm.*, vol. 24, September 1976.

[Milner 80] R. Milner. A Calculus of Communicating Systems, *LNCS 92*, Springer-Verlag, 1980.

[Moy 89] J. Moy. The OSPF specification. *RFC 1131*, SRI Network Information Center, October 1989.

[Ostroff 89] J.S. Ostroff. *Temporal Logic for Real-Time Systems.* Research studies press Ltd. John Wiley & Sons Inc. England, 1989.

[Pansiot 95] J.J. Pansiot, D. Grad, S. Marc-Zwecker. Towards a Logical Addressing and Routing sublayer for Internet multicasting. *PROMS '95, Second Workshop on Protocols for Multimedia Systems, "Mozart on Multimedia Highways",* pp. 438-451, Salzburg, Austria, Octobre 1995.

[Perlman 85] R. Perlman. An Algorithm for Distributed Computation of a Spanning Tree in an Extended LAN. *ACM,* pages 44-53, September, 1985.

[Perlman 88] R. Perlman, A. Harvey, G. Varghese. Choosing the Appropriate ISO Layer for LAN Interconnection. *Special issue: "Bridge and Routers", IEEE Network,* 2(1):81-86, Jan. 1988.

[Perlman 94] R. Perlman. *Interconnexions Ponts et routeurs.* Titre original: Interconnections: Bridges & Routers Ed. Addison-Wesley France, 1994.

[Petri 62] C.A. Petri. Kommunikation mit Automaten. In German, University of Bonn, Germany, Reprinted in English as *Communication with Automata,* Suppl. 1 to Tech. Report RADC-TR-65-377, Vol. 1, Griffis AFB, New-York, 1966.

[Poo 90] G. S. Poo and W. Ang. OSI addressing strategies for interconnected LANs. *Interconnected LANs,* 13(5):290-297, June 1990.

[Ramachandani 74] C. Ramachandani, *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets.* PhD thesis, Massachussets Inst. Technol., 1974.

[Rayward 83] V. J. Rayward-Smith, "The computation of nearly minimal Steiner trees in graphs," *Int. J. Math.* Ed. Sci. Tech., vol. 14, No. 1, pp. 15-23, 1983.

[Rayward 86] V. J. Rayward-Smith and A. Clare, "on finding Steiner vertices," *Networks,* vol. 16, pp. 283-294, 1986.

[Reisig 86] W. Reisig. Place/Transition Systems. *LNCS,* pp. 117-141, Petri Nets: Central Models and their Properties, Springer-Verlag, Bad Honnet, 1986.

[Rescher 71] N. Rescher, A. Urquhart. *Temporal Logic.* Springer-Verlag, 1971.

[Roffinella 87] D. Roffinella, C. Trinchero and G. Frecschi. Interworking Solutions for a Two-Level Integrated Services Local Area Network. *IEEE Journal on Selected Areas in Communications,* Sac-5(9):1444-1453, December 1987.

[Saba 93] G. Saba, J.P. Thomesse, Y.Q. Song. Space and Time Consistency Qualifications in a Distributed Communication System, *IMACS/IFAC,* pp. 383-391, Brusseles, Belgium, April, 1993.

[Saba 95a] G. Saba. An Approximate Solution to Minimal-cost Shortest Path Multipoint Routing Problem, *PROMS '95, Second Workshop on Protocols for Multimedia Systems, "Mozart on Multimedia Highways",* pp. 459-466, Salzburg, Austria, Octobre, 1995.

[Saba 95b] G. Saba, Z. Mammeri, J.P. Thomesse. Some Solutions for FIP Network Interconnection, *WFCS '95, First IEEE Workshop on Factory Communication Systems,* pp. 13-20, October 1995.

[Saba 95c] G. Saba, Z. Mammeri, J.P. Thomesse. Interconnexion de réseaux FIP. *2ème Congrès biennal de l'AFCET,* pp. 519-528, Toulouse, Octobre 1995.

[Sajkowski 85] M. Sajkowski. Protocol Verification techniques: Status Quo and perspectives. In Y. Yemini, R. Strom and S. Yemini, editors, *Protocol Specification, Testing, and Verification, IV,* pp. 697-719, IFIP, Elsevier Science, North-Holland, 1985.

[Schwabe 85] D. Schwabe, A. R. Cavalli and L. I. T. P. . Temporal logic specification of a virtual ring LAN access protocol. In Y. Yemini, R. Strom and S. Yemini, editors, *Protocol Specification, Testing, and Verification, IV,* pp. 79-91, IFIP, Elsevier Science, North-Holland, 1985.

[Seifert 88] W. M. Seifert. Bridges and Routers. *Special issue: "Bridge and Routers", IEEE Network,* 2(1) :57-64, Jan. 1988.

[Shaw 92] A. C. Shaw. Communicating Real-Time State Machines. *IEEE Transactions on Software Engineering,* 18(9):805-816, September 1992

[Simone 89] R.D. Simone, D. Vergamini: Programmation, Calcul Symbolique et Intelligence Artificielle, ABOARD AUTO, *Rapports Techniques-INRIA* , N° 111, Octobre 1989.

[Soha 88] M. Soha, R. Perlman. Comparison of Two LAN Bridge Approaches. Special issue: "Bridge and Routers", *IEEE Network,* 2(1):37-43, Jan. 1988.

[Soulas 89] B. Soulas. Modèle d'adressage FIP, *Rapport technique, EDF-DER* 1989.

[Sunshine 90] C. A. Sunshine. Network Interconnection and Gateways. *IEEE Journal on Selected Areas in Communications,* 8(1):4-11, January 1990.

[Svobodova 90] L. Svobodova, P. A. Janson and E. Mumprecht. Heterogeneity and OSI. *IEEE Journal on Selected Areas in Communications,* 8(1):67-79, January 1990.

[Takahashi 80] H. Takahashi and A. Matsuyma, "An approximate solution for the Steiner Problem in graphs," *Math. Japonica 6,* pp. 573-577, 1980.

[Tanaka 90] A. Tanaka and J. Mizusawa, "Routing Algorithm of Multiple-Destination Telecommunication Services," *Elecronics and Communications in Japan,* Part 1, vol. 73, No. 7, 1990.

[Tanenbaum 90] A. Tanenbaum. *Réseaux, Architectures, protocoles, applications,* InterEdition, 1990, (titre original: Computer Networks by Prentice-Hall, 1989).

[Thiagarajan 86] P. S. Thiagarajan. elementary net systems. In G. Goos, J. Hartmanis, editors, Lecture Notes in Computer Science, pp. 26-59, *Petri Nets: Central Models and their Properties*, Spring-Verlag, Bad Honnet, 1986.

[Thomesse 89a] J.P. Thomesse et Ph. Leterrier, Le bus de terrain FIP, *Mini-Micros*, Oct. 1989.

[Thomesse 89b] J.P. Thomesse, P. Noury. Communication models. Cleint-server vs Producer-Distributor-Consumers. *Contribution ISO TC 184/SC5/WG2-TCCA*, Oct. 89.

[Thomesse 90] J.P. Thomesse. Les réseaux locaux à temps critique. *Colloque : Le Temps Réel,* Nantes, pp. 29-40, Oct. 1990.

[Thomesse 93] J.P. Thomesse, Le réseau de terrain FIP. *Revue Réseaux et Informatique Répartie*, Vol. 3, No. 3, pp. 287-321, 1993.

[Tode 92] H. Tode, Y. Sakai, M. Yamamoto, H. Okada, and Y. Tezuka, "Multicast Routing Algorithm for Nodal Load Balancing," *Proceedings of the Eleventh Annual Conference of the IEEE Computer and Communications Societies*, pp. 2086-2095, IEEE Infocom, Florence, Italy, May 1992.

[Varghese 90] G. Varghese and R. Perlman. Transparent Interconnection of Incompatible Local Area Networks Using Bridges. *IEEE Journal on Selected Areas in Communications,* 8(1):42-48, January 1990.

[Verilog 91] Verilog 91 *EVAL, System Modelling and Analysis.* Technical Presentation, EVAL/NT.A/1.0, Verilog, Toulouse.

[Vuong 87] S. T. Vuong and D. D. Cowan. VALIRA - A tool for protocol validation via reachability analysis. In B. Sarikaya and G. V. Bochmann, editors, *Protocol Specification, Testing, and Verification, VI,* pp. 35-41, IFIP, Elsevier Science, North-Holland, 1987.

[Waxman 88] B. M. Waxman. Routing of multiple connections. *IEEEJ., Sel. Areas comm.*, SAC-6(9):1617-1622, 1988.

[Wei 94] L. Wei, D. Estrin. The Trade-offs of Multicast Trees and Algorithms. *Proceedings of the 1994 International Conference on Computer Communications and Networks*, 1994.

[Zatti 88] S. Zatti and P. Janson. Interconnecting OSI and Non-OSI Networks using an Integrated Directory Service. *Computer Networks and ISDN Systems,* 15:269-283, 1988.

[Zimmerman 80] H. Zimmerman, OSI reference model. The OSI model of architecture for open system interconnection, *IEEE Trans.* COM-28 N°4, pp. 425-432, April 1980.

[Zwecker 93] S. Zwecker. *Représentation et spécification d'architectures d'interconnexion de réseaux hétérogènes : vers la définition d'une base de conaissances.* Thèse du Laboratoire d'Automatique et d'Analyse des Systèmes (LAAS du CNRS), 1993.

Normes

[CCITT 84] CCITT/SGXI/WP3-1: Specification and Description Language, CCITT Recommendations Z100-Z104, 1984.

[CCITT X.500] ISO and CCITT information Processing Systems - Open Systems Interconnection - The Directory ISO DIS 9594-1 to 9594-8, CCITT X.500-X.521, Geneva, Switzerland, 1988.

[CCITT X200] CCITT. Réseaux de communications de Données, Interconnexion de Systèmes Ouverts (OSI), Technique de Description du Système. Volume 8.5, CCITT, Récommandations X.200 à X.250 edition, 1985.

[EMUG 89] European MAP Users Group. User requirements for communications in time critical applications. réf. ISO/TC184/SC5/WG2-N180, Feb. 1989.

[IEEE 802.1] IEEE Project 802, Local and metropolitan area network Standard overview, interworking, and system management, Draft Standard 802.1, July 1987.

[IEEE 802.2] ANSI/IEEE Standard, Draft International Standard 802.2, Logical Link Control. IEEE, 1985.

[IEEE 802.5] IEEE Computer Soc., Token ring access method and physical layer specification, ANSI/IEEE Standard 802.5 (ISO/DP 8802/5), IEEE, 1985.

[IEEE 802.6] IEEE Document 802. 6A-90/02, Multiport Bridge, June, 1990.

[ISO 7498-3] Naming and Addressing, Reference Model of Open Systems Interconnection, March 1989.

[ISO 8348/AD1] Information processing systems - Data communications - Network Service definition, Addendum 2: Network layer addressing,, ISO 8348/AD1.

[ISO 9542] Information processing systems - Data communications - End systems to intermediate systems routing exchange protocol for use in conjunction with the protocol for the provision of the connectionless-mode network service, ISO 9542, 1989.

[ISO 10747] Information processing systems - Data communications - Intermediate System to Intermediate System Interdomain routeing exchange protocol, ISO 10747.

[ISO DP 10028] Information processing systems - Data communications - Definition of the relaying functions of the network layer intermediate system, ISO DP 10028, 1988.

[ISO DP 10589] Information processing systems - Data communications - Intermediate System to Intermediate System routeing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service (ISO 8473), ISO DP 10589, 1990.

[ISO/IEC JTC1/SC6] Secretariat USA (ANSI). Intermediate System to Intermediate System Intra-Domain Routeing Exchange Protocol for Use in Conjunction with ISO 8473, October 1989.

[ISO/IEC802.1] ISO/IEC 802.1 MAC Bridge, DP 10038 MAC sublayer.

[ISO/IEC8348] Draft amendment 5 (group network addressing) to the Network service definition ISO/IEC 8348 (Information Technology - Telecommunications and information Exchange Between Systems - Network Service Definition for Open Systems Interconnections) 1993, which is identical to ITU-T Recommendation X.213.

[ISO/IEC8473-1] Draft amendment 1 (multicast extension) to ISO 8473-1 (Information Technology - Protocol for Providing the Connectionless-mode Network Service: Protocol Specification), 1993.

[ISO/IEC9542] Draft amendment 2 (multicast extensions) to ISO 9542 (Information processing systems - Telecommunications and information exchange between systems - End system to Intermediate System routeing exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service), 1993.

[ISO3] ISO 9074. ESTELLE - A Formal Description Technique Based on an Extended State Transition Model, 1989.

[ISO4] ISO 9074/PDAM1, Amendment1: ESTELLE Tutorial.

[ISO5] ISO 8807. LOTOS - A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour, 1989.

[ISO8802-5] ISO 8802-5/DPA D4. Bridging-Routing.

[ITU 93] ITU: Z.100 ITU Specification and Description Language (SDL), Geneva, 1993.

[RFC 1058] C. L. Hedricks, «Routing Information Protocol», Novembre 1987.

[RFC 1075] S. Deering, C. Partridge, D. Waitzman, « Distance Vector Multicast Routing Protocol », 1988.

[RFC 1112] S. Deering, « Host extensions for IP multicasting », 1988.

[RFC 1584] J. Moy, « Multicast extensions to OSPF », Mars 1994.

[RFC 1585] J. Moy, « MOSPF : Analysis and experience »

[UTE 89] UTE NF C46-605, Gestion de bus de terrain FIP. Union Technique de l'Electricité, Paris, 1990.

[UTE 90a] UTE C46-601, Architecture et présentation générale du bus de terrain FIP. Union Technique de l'Electricité, Paris, 1990.

[UTE 90b] UTE NF C46-602, Couche Application - services périodiques et apériodiques. Union Technique de l'Electricité, Paris, 1990.

[UTE 90c] UTE NF C46-603, Couche Liaison de données. Union Technique de l'Electricité, Paris, 1990.

[UTE 90d] UTE NF C46-604, Couche physique en bande de base sur paire torsadée blindée. Union Technique de l'Electricité, Paris, 1990.

[UTE 91a] UTE NF C46-607, Couche physique en bande de base sur fibre optique. Union Technique de l'Electricité, Paris, 1990.

[UTE 91b] UTE C46-603/ADD_1, Bus FIP pour échange d'informations entre transmetteurs, actionneurs et automates. Couche liaison de données. Union Technique de l'Electricité, Mars 1991.

[UTE 92] UTE NF C46-606, Couche Application - services de messagerie. Union Technique de l'Electricité, Paris, 1990.

Glossaire des sigles

AB ..... Arbitre de Bus
AFI..... Authority Format Identifier
ANSI..... American National Standardization Institute
BPDU..... Bridge Protocol Data Unit
CCITT ..... Comité Consultatif International Télégraphique et Téléphonique
CCR ..... Commitment Concurrency and Recovery
CCS ..... Calculus of Communicating Systems ,
CEI..... Commission Electronique Internationale
CEP..... Connection End Point
CLNS..... ConnectionLess-mode Network Service
CONS..... COnnection-mode Network Service
CRSM ..... Communicating Real-time State Machines
CSP..... Communicating Sequential Processes
DIT ..... Directory Information Tree
DSP..... Domain Specific Part
DVMR ..... Distance Vector Multicast Routing
ES ..... End System
FDT..... Formal Description Techniques,
FIP..... Factory Instrumentation Protocol
HMS ..... Hierarchical Multi-state Machine
IDI..... Initial Domain Identifier
IDP..... Initial Domain Part
IEEE..... Institute of Electrical and Electronics Engineers
IETF ..... Internet Engineering Task Force
IGMP..... Internet Group Management Protocol
IS..... Intermediate System
ISO..... International Standardization Organization
ITU ..... International Telecommunication Union
LSP ..... Link State Packet
MAC ..... Medium Access Control

MMS ..... Manufacturing Message Specification
MOSPF ..... Multicast Open Shortest Path First
MPS ..... Message Périodique/apériodique Specifications
MSPST ..... Minimal-cost Shortest Path Spanning Tree
MST ..... Minimum Spanning Tree
PDU ..... Protocol Data Unit
OSI ..... Open Systems Interconnection
OSPF ..... Open Shortest Path First
RDN ..... Relative Distinguished Name
RPF ..... Reverse Path Forwarding
RTTL ..... Real Time Temporal Logic
SAP ..... Service Access Point
SDL ..... Specification and Description Language
SDU ..... Service Data Unit
SMT ..... Steiner Minimal Tree
SNPA ..... SubNetwork Point of Attachment
SPF ..... Shortest Path First
SPT ..... Shortest Path Tree
TCP/IP ..... Transmission Control Protocol/Internetwork Protocol

Titre : Protocoles multipoint et interconnexion de réseaux FIP

Résumé : Ce travail est une contribution au problème de routage multipoint d'une part et à l'interconnexion de réseaux FIP (Factory Instrumentation Protocol) par l'intermédiaire de ponts d'autre part.

Conçues dans une perspective générale, les solutions classiques d'interconnexion de réseaux, y compris les algorithmes de routage, se sont révélées inadaptées pour traiter les spécificités inhérentes au réseau FIP. Nous citons parmi ces dernières la gestion d'accès au bus qui n'est pas séparée du contrôle logique de liaison, la coexistence de plusieurs types d'adresses et de trames, l'absence de routage de bout en bout et la communication en mode multipoint pouvant être assujettie à des contraintes temporelles très strictes.

Pour atteindre notre but, nous proposons tout d'abord une nouvelle structure de la couche Liaison de données de FIP. Cette structure, outre la séparation entre la gestion d'accès au bus et le contrôle logique de liaison, nous fournit un adressage global de différentes entités de la sous-couche LLC (Logical Link Control) qui est fonction des points d'accès aux services (SAPs) et des extrémités de connexion (CEPs). De cette structuration, nous avons déduit des solutions d'interconnexion de réseaux FIP qui, en réservant des ressources chez les ponts et de la bande passante (identifieurs cycliques) chez l'Arbitre de Bus, fournissent un moyen permettant de mieux respecter les contraintes temporelles.

Les mécanismes décrits informellement peuvent comporter des erreurs fonctionnelles et/ou des incohérences. Aussi, nous avons utilisé des techniques de spécification formelle et de vérification pour valider le pont et la nouvelle architecture du réseau FIP.

Pour remédier enfin aux problèmes de bouclage et d'optimalité de chemins, nous avons développé un algorithme de routage multipoint bien adapté à la communication en mode multipoint et temps réel.

Ce travail permet donc de mettre en évidence le fait que les mécanismes de réservation de ressources renforcés par des algorithmes efficaces de routage sont deux facteurs essentiels qui sous-tendent la réussite d'une interconnexion de réseaux dans un environnement temps réel et multipoint.

Mots clés : interconnexion de réseaux, pont, routage multipoint, bus de terrain, FIP, temps réel, spécification formelle

---

Title : Multipoint protocols and FIP networks interconnection

Abstract : The work presented in this thesis is a contribution to multipoint routing problem and more specifically to FIP (Factory Instrumentation Protocol) networks interconnection by bridges.

Unfortunately, the usual interconnection strategies, including routing algorithms, turned out unsatisfactory to cope with the specific problems of FIP. For these reasons, we have developed a novel solution that considers the following aspects: the Medium Access Control (MAC) is not separated from the Logical Link Control (LLC), several styles for addressing and frames format are used, the lack of end-to-end routing, and the multicast communication capabilities that are submitted to real-time constraints.

To attend our goal, we rebuild first the data link layer of FIP in order to separate MAC functionality from LLC ones, and to provide global addressing of LLC entities by means of Service Access Points (SAPs) and Connection End Points (CEPs). Thus, some interconnection solutions that, by providing resources reservation in the bridges and throughput allocations (by means of cyclic identifiers) in Bus Arbitrators, enable the respect of some real-time constraints.

To avoid now functional errors, inconsistency, and incompleteness of the previous informal solutions, we are led to use formal specification technique. This later enables us to validate both the behaviour of a bridge and the new architecture of FIP network.

To garantee further that paths are optimal and cycle-free, we have developed a multipoint routing algorithm that, in addition to handling previous problems, is well adapted to multipoint and real-time communication.

As a conclusion, we recall that a good strategy for resources reservation and an efficient routing algorithm are the key-elements behind the success of networks interconnection in a real-time and multipoint environment.

Keywords : networks interconnection, bridge, multipoint routing, fieldbus, FIP, real-time, formal specification

INSTITUT NATION
POLYTECHNIQ
DE LORRAI

AUTORISATION DE SOUTENANCE DE THESE DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

o0o

VU LES RAPPORTS ETABLIS PAR :
Monsieur PANSIOT Jean-Jacques, Professeur, Université Louis Pasteur Strasbourg,
Monsieur SAHRAOUI Abd-El-Kader, Professeur, Laboratoire d'Automatique des Systèmes du CNRS Toulouse,
Monsieur SCHAFF André, Professeur, CRIN/LORIA Université Nancy I.

Le Président de l'Institut National Polytechnique de Lorraine, autorise :

NANCY BRABO
2, AVENUE DE L
FORET-DE-HAY
BOITE POSTALE
F - 5 4 5 0
VANDŒUVRE CEDE

Monsieur SABA Ghassan

à soutenir devant l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE, une thèse intitulée :
"Protocoles multipoint et interconnexion de réseaux FIP".

en vue de l'obtention du titre de :

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : **"INFORMATIQUE"**

Fait à Vandoeuvre le, **21 Décembre 1995**
Le Président de l'I.N.P.L.,
M. LUCIUS

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

Pour le Président
Le Vice-Président
J. Ch. CHEVRIER

TEL. 33/83.59.59.
FAX. 33/83.59.59.

Titre : Protocoles multipoint et interconnexion de réseaux FIP

Résumé : Ce travail est une contribution au problème de routage multipoint d'une part et à l'interconnexion de réseaux FIP (Factory Instrumentation Protocol) par l'intermédiaire de ponts d'autre part.

Conçues dans une perspective générale, les solutions classiques d'interconnexion de réseaux, y compris les algorithmes de routage, se sont révélées inadaptées pour traiter les spécificités inhérentes au réseau FIP. Nous citons parmi ces dernières la gestion d'accès au bus qui n'est pas séparée du contrôle logique de liaison, la coexistence de plusieurs types d'adresses et de trames, l'absence de routage de bout en bout et la communication en mode multipoint pouvant être assujettie à des contraintes temporelles très strictes.

Pour atteindre notre but, nous proposons tout d'abord une nouvelle structure de la couche Liaison de données de FIP. Cette structure, outre la séparation entre la gestion d'accès au bus et le contrôle logique de liaison, nous fournit un adressage global de différentes entités de la sous-couche LLC (Logical Link Control) qui est fonction des points d'accès aux services (SAPs) et des extrémités de connexion (CEPs). De cette structuration, nous avons déduit des solutions d'interconnexion de réseaux FIP qui, en réservant des ressources chez les ponts et de la bande passante (identifieurs cycliques) chez l'Arbitre de Bus, fournissent un moyen permettant de mieux respecter les contraintes temporelles.

Les mécanismes décrits informellement peuvent comporter des erreurs fonctionnelles et/ou des incohérences. Aussi, nous avons utilisé des techniques de spécification formelle et de vérification pour valider le pont et la nouvelle architecture du réseau FIP.

Pour remédier enfin aux problèmes de bouclage et d'optimalité de chemins, nous avons développé un algorithme de routage multipoint bien adapté à la communication en mode multipoint et temps réel.

Ce travail permet donc de mettre en évidence le fait que les mécanismes de réservation de ressources renforcés par des algorithmes efficaces de routage sont deux facteurs essentiels qui sous-tendent la réussite d'une interconnexion de réseaux dans un environnement temps réel et multipoint.

Mots clés : interconnexion de réseaux, pont, routage multipoint, bus de terrain, FIP, temps réel, spécification formelle

---

Title : Multipoint protocols and FIP networks interconnection

Abstract : The work presented in this thesis is a contribution to multipoint routing problem and more specifically to FIP (Factory Instrumentation Protocol) networks interconnection by bridges.

Unfortunately, the usual interconnection strategies, including routing algorithms, turned out unsatisfactory to cope with the specific problems of FIP. For these reasons, we have developed a novel solution that considers the following aspects: the Medium Access Control (MAC) is not separated from the Logical Link Control (LLC), several styles for addressing and frames format are used, the lack of end-to-end routing, and the multicast communication capabilities that are submitted to real-time constraints.

To attend our goal, we rebuild first the data link layer of FIP in order to separate MAC functionality from LLC ones, and to provide global addressing of LLC entities by means of Service Access Points (SAPs) and Connection End Points (CEPs). Thus, some interconnection solutions that, by providing resources reservation in the bridges and throughput allocations (by means of cyclic identifiers) in Bus Arbitrators, enable the respect of some real-time constraints.

To avoid now functional errors, inconsistency, and incompleteness of the previous informal solutions, we are led to use formal specification technique. This later enables us to validate both the behaviour of a bridge and the new architecture of FIP network.

To garantee further that paths are optimal and cycle-free, we have developed a multipoint routing algorithm that, in addition to handling previous problems, is well adapted to multipoint and real-time communication.

As a conclusion, we recall that a good strategy for resources reservation and an efficient routing algorithm are the key-elements behind the success of networks interconnection in a real-time and multipoint environment.

Keywords : networks interconnection, bridge, multipoint routing, fieldbus, FIP, real-time, formal specification