THESE de DOCTORAT de l'UNIVERSITE PARIS 6

Spécialité :

INFORMATIQUE

présentée

par Monsieur **COUSIN Bernard**

pour obtenir le titre de **DOCTEUR DE L'UNIVERSITE PARIS 6**

Sujet de la thèse :

METHODOLOGIE DE VALIDATION DES SYSTEMES STRUCTURES EN COUCHES PAR RESEAUX DE PETRI

Application au Protocole Transport

soutenue le 9 AVRIL 1987 devant le jury composé de :

Messieurs G.BERTHELOT

M.DIAZ

M.ELIE

C.GIRAULT

G.PUJOLLE

à Daniéle

Méthodologie de validation des systèmes :

Je tiens à remercier tout particulièrement,

Monsieur C.Girault pour l'enseignement que j'ai reçu depuis plusieurs années, l'attention avec laquelle il a dirigé mes travaux, et les conseils qu'il m'a prodigués, aussi bien sur la forme que sur le fond, tout au long de cette thèse.

Monsieur G.Berthelot qui fut à l'origine de cette thèse et qui a accepté d'en être rapporteur. Je le remercie pour le temps qu'il a consacré à examiner ma thèse et pour les nombreuses remarques qu'il m'a adressées.

Monsieur M.Diaz qui malgré ses multiples occupations, a eu la grande obligeance de se déplacer de Toulouse, et a accepter la charge de rapporteur.

Monsieur G.Pujolle d'avoir accepté la présidence de ce jury.

Monsieur M.Elie qui me fait l'honneur de participer à ce jury.

Je n'oublie pas tous les chercheurs du laboratoire MASI qui, à l'occasion de débats, de séminaires ou de réunions, m'ont soutenu dans mon travail. Je veux citer tout spécialement P.Estraillier dont la collaboration a toujours été fructueuse.

Je remercie également une personne très dévouée, et les services de l'Institut de Programmation pour l'aide qu'ils m'ont apportée dans la réalisation pratique du document.

Méthodologie de validation des systèmes :

0.1 RESUME

Nous développons une méthode de modélisation et de validation adaptée aux systèmes parallèles structurés en couches hiérarchiques. Nous définissons deux notions : la concordance de modèle prouve que le modèle possède bien les propriétés dégagées par les spécifications; l'adéquation de service valide le protocole par rapport à son service.

Nous appliquons notre méthode à la modélisation du protocole de télécommunication de niveau Transport (la couche 4 d'après la norme ISO sur l'interconnexion des systèmes ouverts). Nous étudions tout particulièrement la gestion des désynchronisations du Service de la couche Réseau, et le contrôle de flux avec réquisition de crédit du Protocole de la couche Transport.

Nous utilisons les réseaux de Petri à prédicats pour décrire le modèle du service rendu par la couche Réseau sous-jacente et nous en servir pour construire le modèle du protocole de la couche Transport. Nous prouvons que la notion d'abstraction peut s'étendre aux réseaux de Petri à prédicats. La preuve du déroulement correct du protocole est apportée en utilisant les invariants issus du modèle.

MOTS CLEFS

Validation, Modélisation, Protocole de la couche Transport, Service de la couche Réseau, Réseau de Petri à prédicats, Contrôle de flux, réquisition de crédit, Désynchronisation.

Méthodologie de validation des systèmes :

P L A N de la THESE

Méthodologie de validation des systèmes :

0.3 FIGURES

Méthodologie de validation des systèmes :

Méthodologie de validation des systèmes :

0.4 INTRODUCTION

Nous développons une méthode permettant d'intégrer la modélisation et la validation des systèmes structurés en couches dans une approche globale de la conception des systèmes informatiques. Nous appliquons notre méthode à la preuve et la modélisation du protocole de communication de la couche Transport et du service de la couche Réseau. Nous utilisons comme outil de preuve et de modélisation les réseaux de Petri à prédicats.

La diffusion de l'informatique, et notamment celle des systèmes distribués, nécessite le développement d'outils et de méthodes permettant d'effectuer leur conception à moindre coût et surtout d'assurer leur qualité. La proposition de notre méthode s'inscrit pleinement dans le cadre de cette démarche.

Les protocoles de communication sont un bon exemple de coopération parallèle de processus et où le problème de preuve de bonne fonctionnalité pour assurer un bon fonctionnement est vraiment crucial (notre société repose sur les communications et dépend de plus en plus de la fiabilité des télécommunications). C'est pourquoi, nous avons choisi d'y appliquer notre méthode, afin d'en montrer la difficulté, la faisabilité et l'intérêt.

La normalisation des protocoles de télécommunication, en vue d'obtenir une interconnectabilité des systèmes informatiques, définit un modèle logique appelé "modèle de référence pour l'interconnection des systèmes ouverts" (OSI). Cette norme définit des concepts et une terminologie, avec notamment les notions de couche, protocole et service.

Le modèle de référence est structuré en sept couches fonctionnelles offrant des services de plus en plus sophistiqués. Chaque couche n'est en relation qu'avec les couches directement contiguës, elle utilise seulement le service de la couche

inférieure pour fournir le service à la couche supérieure. La normalisation munit chacune des couches : de son service définissant de manière détaillée les fonctionnalités qu'elle offre aux autres couches; et de son protocole spécifiant les règles qui régissent les échanges à l'intérieur d'une couche entre entité distante. Nous voyons donc apparaître pour chacune des couches du système deux descriptions, le service et le protocole. Il faut établir l'adéquation de ces deux descriptions.

Notre méthode définit l'adéquation de service comme la preuve que le protocole (spécification du comportement interne) réalise bien le service (définition du comportement externe) qui caractérise chaque couche d'un système. Cette étape est le point clef du concept de couche, qui permet de faire abstraction des couches sous-jacentes, mais aussi du comportement interne à chaque couche.

L'analyse d'un système nécessite au préalable sa modélisation, qui associée à des outils adaptés, permet l'étude des propriétés du système. Néanmoins cette modélisation entraîne alors une possible divergence entre le système lui-même et sa modélisation.

Notre méthode définit la concordance de modèle comme la preuve que le modèle correspond bien aux spécifications du système étudié. Cette concordance est établie en comparant les propriétés issues du modèle et celles du système.

Parmi les protocoles de communication, notre choix s'est dirigé vers **la couche Transport**, car cette couche est celle normalisée dont les fonctions et le niveau de complexité sont des plus intéressants. Elle comporte un ensemble de fonctionnalités que possède d'autres couches (établissement et multiplexage de connexion, gestion de contrôle de flux, etc...) de gestion déjà assez complexe, mais aussi des fonctionnalités originales et jamais modélisées comme la réquisition de crédit.

De nombreuses modélisations ont déjà été effectuées sur des protocoles de télécommunication, cependant ils portaient soit sur des protocoles différents (HDLC, bit alterné, etc:...), soit aucune preuve n'était jointe au modèle, soit enfin, des

fonctionnalités importantes du protocole étaient écartées de la modélisation. Nous associerons, donc, dans notre étude des protocoles de télécommunication, modélisation et validation. Dans cette thèse, nous modélisons et nous validons le protocole de la couche Transport avec son mécanisme de contrôle de flux à crédit variable et réductible, et le service de la couche Réseau avec son mécanisme de resynchronisation.

Le protocole de la couche Transport gère le contrôle de flux par l'intermédiaire de crédit alloué par le récepteur et l'émetteur. La classe 3 de ce même protocole autorise la réquisition (réduction) d'un crédit déjà alloué. Cette réquisition place le protocole dans un état atypique : des messages envoyés légalement (avant réquisition) peuvent devenir brutalement incorrects (après réquisition). Nous allons démontrer que ce phénomène n'entraine pas d'état incohérent pour le protocole Transport.

Notre outil de modélisation et de preuve est les Réseaux de Petri (RdP), car ils associent une facilité de création de modèles, à des possibilités importantes de preuves, notamment pour tous les problèmes posés par les processus concourants. Les RdP ont été, dès l'origine, conçus pour mettre en évidence les relations de synchronisations, et grâce à leurs extensions (vers les réseaux colorés, à prédicats, à files, etc...) et leurs bases mathématiques associées, ils sont devenus un outil très puissant.

La thèse utilise **les Réseaux de Petri à prédicats**, dont nous donnons une définition formelle, qui nous permet ensuite de prouver la possibilité de modifier un modèle, décrit par RdPàP, tout en conservant un comportement équivalent. Cette équivalence nous est utile pour appliquer notre méthode, où un premier modèle, dont on connaît un ensemble de propriétés, est utilisé plus tard dans un modèle plus grand. Il faut que les propriétés du premier modèle soient conservées par le second. L'équivalence de comportement est appliquée au modèle du service Réseau pour prouver qu'il est indépendant des types de messages qu'il transporte pour le compte de la couche Transport. Le transport de donnée de la couche Réseau est transparent.

Méthodologie de validation des systèmes :

Nous apportons des solutions à plusieurs problèmes majeurs concernant la modélisation ou les preuves, des processus concourants. Pour la plupart, ces problèmes ont déjà trouvé en partie une résolution, cependant nous nous proposons d'intégrer et de compléter ces solutions , en vue de les appliquer à des cas complexes, comme peuvent l'être les protocoles de communication. Nous nous sommes intéressés, notamment, aux trois points suivants évoqués en allant du général au particulier :

Le problème de validation de processus concourants est un problème complexe et d'envergure. Nous y apportons une solution, après une réflexion et une classification nous permettant une approche méthodologique du problème. Nous développons une méthode liant les différentes phases de conception des systèmes informatiques aux deux points qui nous préoccupent particulièrement, la modélisation et la validation.

Qui veut modéliser un cas réel (ici le protocole Transport) rencontre le problème de l'étendue du modèle, et par là, celui de ses preuves. Il est résolu en partie par la puissance de l'outil de modélisation (ici, les Réseaux de Petri à prédicats), mais aussi par l'exploitation d'un principe de construction du modèle. Le modèle est, en fait ,constitué de sous-modèles, dont nous prouvons séparément les propriétés. Nos prouvons que le modèle global possède, aussi, les propriétés des sous-systèmes.

Nous avons modélisé et validé des mécanismes complexes et sophistiqués : le mécanisme de resynchronisation du service de la couche Réseau qui gère les pannes de sites et les pertes de paquets en transit sur les réseaux de transmission; les mécanismes d'acquittement, de retransmission et de contrôle de flux à crédit variable et réductible du protocole de la couche Transport.

Notre thèse est composée de trois parties, **présentation, réseau** et **transport.**

Dans la première partie, nous présentons un état de l'art, tant au niveau des protocoles de télé-communication, que des outils de modélisation et de preuves. Nous évoquons l'importance des problèmes de fiabilité de l'informatique d'aujourd'hui, et nous étudions les différents outils et méthodes proposés pour les résoudre. Puis, nous nous livrons à une présentation du protocole de télécommunication de la couche Transport. Nous enchaînons sur la présentation de notre méthode, qui introduit la concordance de modèle et l'adéquation de service. Enfin, nous rappelons la théorie sur les Réseaux de Petri et son extension vers les Réseaux de Petri à Prédicats (RdPàP), et démontrons la propriété d'équivalence de comportement, qui autorise la modification de la structure des marques circulant dans le modèle, sans changer les propriétés du modèle initial.

Dans la seconde partie, nous donnons une description informelle, et une modélisation en RdPàP du service de la couche Réseau, puis une preuve que les fonctions offertes par le modèle sont bien compatibles avec celles de la norme internationale de la couche Réseau, en accord avec la méthode proposée (concordance de modèle). Au cours de cette étude, nous approfondissons, notamment, la modélisation et la validation de la gestion des désynchronisations survenant pendant la phase de transfert de données du service Réseau. Une désynchronisation est la conséquence de pannes survenues sur les sites intermédiaires constituant le circuit virtuel qui relie les deux entités communiquantes.

Au cours de la troisième partie, s'appuyant sur le modèle de la partie précédente, nous définissons et modélisons le protocole de la couche Transport, puis nous prouvons sa bonne fonctionnalité par rapport aux spécifications de son service (adéquation de service). Nous étudions un ensemble important de fonctionnalités du protocole, notamment le mécanisme de contrôle de flux par crédit avec réquisition possible.

Méthodologie de validation des systèmes :

A - PREMIERE PARTIE

PRESENTATION

des Protocoles de Télécommunication

de notre Méthodologie de Validation

des Réseaux de petri à Prédicats

PLAN
de la Première Partie

1. INTRODUCTION

Dans cette première partie, nous expliquons et développons les trois sujets d'intérêt essentiel de notre thèse. Ce sont les Réseaux de petri à prédicats, le protocole de transmission téléinformatique de la couche appelée Transport, et les méthodes de modélisation et de validation. Cette partie de présentation est composée de quatre chapitres.

Le premier chapitre présente **l'état de l'art des méthodes de modélisations et de preuves**. Il permet de situer notre travail dans le prolongement des recherches passées et présentes sur la fiabilité des logiciels. L'étude sur les outils de modélisation et de preuves, nous amène à une réflexion sur les moyens et les objectifs de ces outils. Cette étude nous amène naturellement à évoquer les Réseaux de petri, outils de modélisations et de preuves que nous avons choisis ici. La description de cet outil est effectuée au dernier chapitre.

Le deuxième chapitre offre une **présentation d'ensemble des protocoles de télécommunication**. Notre choix s'explique aisément par le fait que les protocoles de communication sont de bons exemples de logiciel, où la complexité et le parallélisme nécessitent l'emploi de méthodes rigoureuses de conception. Dans un premier temps, nous situons l'ensemble des études sur les méthodes d'acheminement des données, puis nous présentons les **fonctionnalités du protocole de communication de couche 4 (dit Transport)**. Nous abordons successivement les concepts propres au modèle de référence des systèmes ouverts, puis plus précisément les fonctionnalités propres à la couche Transport (protocole de bout en bout, acquittement, contrôle de flux, fenêtre, crédit, contrôle d'erreur, retransmission, etc...). L'intérêt de ce protocole réside dans la complexité de ses fonctionnalités et particulièrement celle liée au mécanisme de crédit variable et réductible.

Le troisième chapitre est une **étude plus générale de la modélisation et de la validation des systèmes** informatiques réels et complexes. Nous étudions les concepts de couches et de service qui permettent une simplification et une abstraction du système. Nous développons une méthode permettant d'utiliser ces concepts aux phases de modélisation et de validation, en définissant deux procédés: la **concordance de modèle et l'adéquation de service**. La concordance permet d'établir que le modèle respecte bien les spécifications du système, l'adéquation prouve l'équivalence (nécessaire par définition de la notion de service) des propriétés issues du modèle du protocole et des propriétés issues du modèle du service de la couche étudiée. Ces deux notions seront appliquées par la suite pour la modélisation du protocole de Transport.

Le quatrième chapitre permet au lecteur d'appréhender l'outil que sont les réseaux de petri, notamment leur intérêt pour la modélisation et la preuve, intérêt dû en grande partie à l'émulation et la confrontation de l'ensemble des équipes travaillant sur le sujet depuis la création par petri de ses réseaux. Ainsi nous abordons succinctement la théorie des Réseaux de petri telle qu'elle a été définie par petri en 62 [Petri 62], puis par l'exemple d'un modèle, nous étudions l'intérêt et la concision offerts par les **Réseaux de petri à prédicats**. Au moyen d'une définition formelle des RdPàP, nous montrons un procédé original qui permet de modifier la structure des champs des uplets d'une classe d'un réseau tout en étant assurés de conserver toutes les propriétés du réseau initial. Ce procédé nous sera utile pour établir la conservation des propriétés du modèle du service Réseau (construit à la partie B de notre thèse) dans le modèle du protocole Transport (partie C).

Cette première partie, après avoir centré notre discours sur les protocoles de communication, définit donc l'ensemble des méthodes et outils qui nous seront utiles tout au cours de notre thèse.

2. la FIABILITE

2.1 les Méthodes

Tous les utilisateurs de systèmes informatiques exigent de plus en plus de fiabilité. Ceci est dû en grande partie à l'extension et à la banalisation de l'informatique. Le monde d'aujourd'hui paye la puissance acquise par l'outil informatique, d'une dépendance importante et d'une fragilité dues à la complexité des problèmes traités.

Les questions de fiabilité apparaissent à tous les niveaux d'utilisation de l'informatique, que ce soit pendant la phase de conception des programmes ou durant leur utilisation [Laprie 85].

La **fiabilité de fonctionnement est cruciale**: Ainsi avoir un produit livré qui fonctionne mal peut être très coûteux, voire catastrophique, pour l'utilisateur, comme pour le concepteur;

Le **réemploi et la standardisation**: L'accroissement de la demande de programmes provoque la nécessité de réutiliser les produits déjà existants, afin de les intégrer aux produits nouveaux, pour avoir un moindre coût de production. Ceci demande une bonne connaissance (spécification) et une bonne structuration (conception) des fonctionnalités de chaque produit;

La **validation de la conception et la correction de la programmation**: Enfin et surtout, de nombreuses études montrent que, plus les erreurs où mauvais choix sont détectés tardivement pendant la vie du produit, plus le coût de correction ou d'adaptation est important. De ce fait, les méthodologies qui permettent d'éviter ou de contrôler les erreurs doivent être performantes, mais aussi intervenir le plus tôt possible dans les phases de conception.

Les problèmes de fiabilité apparaissent donc être une préoccupation actuelle de plus en plus essentielle. Pour répondre à cette demande, deux approches peuvent être discernées:

- Une approche où le système est tolérant aux fautes, c'est-à-dire basé sur la redondance tant logicielle que matérielle. Ceci permet d'assurer un service continu (bien que parfois dégradé) en dépit de fautes résiduelles ou de défaillances imprévues. Ainsi, se sont développées des machines à organes redondants (double C.P.U, contrôleurs d'E/S autonomes, organes de mémoires supplémentaires, bus multiples, etc...), ou des logiciels s'autotestant et à récupération d'erreurs. L'un des plus fameux systèmes, employant cette méthode, se trouve être la navette spatiale américaine.

Nous laisserons de côté cette approche, car il n'existe actuellement aucune méthodologie affirmée pour résoudre les problèmes liés à la redondance. De plus, cette approche tente de corriger les erreurs en aval, c'est à dire après qu'elles soient apparues.

- L'approche préventive, elle, met en évidence les problèmes de conception et permet d'éviter les fautes d'implantations. Cette approche s'applique avant ou pendant la construction du système. Nous dégageons quatre principales classes:

- Les **outils de spécification**, tels que les langages F.D.T (Formal Description Technique) [ISO 8807], [ISO 9074], [Sparc], veulent rendre les procédures de spécification fiables, en évitant les sur-spécifications (spécifications trop détaillées ou multiples), les sous-spécifications (oubli d'un événement ou de son contrôle), les ambiguïtés (certaines spécifications se contredisent). La spécification est la base des méthodes de développement actuel.

- Les **méthodes de validation** reposent généralement sur un modèle construit à partir des spécifications. Associé au modèle, il doit exister une sémantique mathématique qui permette de l'exploiter, que ce soit de manière algébrique ou quantitative. Nous nous intéressons particulièrement aux outils de modélisation et de preuves, qui semblent offrir le plus de rigueur [Ayache 85].

- Les **méthodologies de programmation**, ont provoqué la venue des langages structurés tel que PASCAL ou ADA, [ADA 83], [ARSAC 77] et qui permettent une implantation plus aisée à partir d'une analyse préalable. Une tendance actuelle développe des outils d'implantation automatique [Beaudoin 84]. Nous n'évoquerons pas les problèmes posés par l'évolution, l' adaptation, ou la correction des produits déjà créés, qui relèvent du génie logiciel : gestion de modules, fichiers et versions [Estublier 84].

- Le **contrôle de la qualité et le passage de test** créé de manière plus ou moins automatique, permettent de tester le système produit par rapport à ses spécifications [Favreau 86]. Ces tests interviennent, soit pendant la phase de conception sur des maquettes ou une partie du système, soit après sur le système réel, pour évaluer leurs performances.

Toutes ces méthodes ne sont bien entendu nullement exclusives, mais au contraire complémentaires. La méthode parfaite présentant de manière homogène et automatique si possible, un enchaînement des méthodes liées aux quatre classes [Ansart 86].

Les outils regroupant un ensemble plus ou moins important des éléments précédents, sont en fort développement actuellement dans l'industrie, pour permettre de mettre en commun leurs avantages. Il en est ainsi des outils de génie logiciel ou de validation.

Nous voyons donc apparaître une double nécessité : d'une part, pour des raisons de coûts et d'efficacité, il faut intervenir le plus tôt possible dans les étapes de la création des systèmes (la phase de conception étant la plus cruciale); d'autre part, il faut disposer d'outils et de méthodes permettant d'obtenir une assurance de fiabilité maximale.

C'est pourquoi nous orientons notre discours à partir de maintenant sur les outils de validation amont, c'est-à-dire intervenant pendant la phase de conception des systèmes.

2.2 les Outils

De nombreux outils sont apparus durant ces dernières années pour prouver le bon fonctionnement des systèmes. Nous allons tenter de montrer leur évolution historique, sans assurer l'exhaustivité de cette liste, ni la rigueur du classement, car toutes les méthodes, ici décrites, ne sont pas figées et de nombreuses variantes ont été introduites.

Historiquement, les automaticiens ont eu très tôt le besoin de spécifier et de contrôler les actions de leurs automates. Leurs outils basés principalement sur l'algèbre de Boole, les automates à états, les tables évènement-action, ont été rapidement dépassées quant à la prise en compte de machines plus complexes, par l' accroissement du nombre d'états. La difficulté de mettre en évidence et de contrôler des phénomènes parallèles et simultanés a suscité l'arrivée du Grafcet [Grafcet 77] et sa normalisation en France .

Nous allons tout d'abord distinguer la **logique assertionnelle** introduite par Floyd et Hoare [Floyd 67],[Roucairol 74]. Cette méthode consiste à prouver que le système (décrit par un programme) vérifie un ensemble de formules (appelées assertions) composées de connecteurs logiques, de quantificateurs et de fonctions sur les variables du programme. Les formules ont une sémantique qui permet d'en déduire généralement le comportement du système. Des langages permettent l'insertion de ces formules dans le texte du programme, ce qui permet : le contrôle de la programmation pendant la phase de développement; l'auto-test des programmes lors de la phase de maintenance. Parfois, ce langage est étendu à la phase de spécification. De nombreuses recherches ont été effectuées sur ce domaine, surtout dans un but d'automatiser les systèmes de preuves de programmes.

La **logique temporelle** est un formalisme, qui permet de mieux exprimer et prouver les programmes concourants ou non déterminismes, que la logique assertionnelle dont elle est issue [Pnuelli 79]. Surtout, si on ne veut pas envisager

l'ensemble des états futurs des processus. Pour ce faire, la logique temporelle s'adjoint des opérateurs supplémentaires, dits opérateurs temporels. Il existe de nombreux modèles de cette logique, munis chacun d'opérateurs temporels différents. Par exemple dans [Lamport 80], sont introduits deux opérateurs:

- dorénavant A : qui signifie que l'assertion A est vraie maintenant et pour toujours.
- parfois A : signifiant que l'assertion A est vraie maintenant ou le sera dans le futur, mais peut redevenir fausse.

Le développement important des **outils mathématiques**, notamment algébriques, apporte sa contribution aux méthodes de preuve, en facilitant la création de la théorie des types de données abstraits. Cette méthode repose sur la création d'entité ,appelé objet abstrait, composé d'un identificateur de ce type d'objet, et de l'ensemble des opérations manipulant l'objet. Nous connaissons plusieurs développements:

- création de langage de programmation intégrant ce concept [ADA 83], [Hoare 74].
- production automatique à partir de spécifications formulées à l'aide des types abstraits, notamment de compilateur [Bouillier 84].
- utilisation des types abstraits pour la description de processus concourants dans un langage idoine [Berthomieu 81].
- construction de démonstrateur de théorème se servant des types de données abstraits [Sunshine 82].

L'analyse de système a bénéficié aussi des **théories en probabilité**, notamment la théorie des files d'attente. Cependant cette voie est beaucoup plus quantitative que précédemment, car on tente de connaître les performances du système. Cette méthode de modélisation est fortement teintée de temporalité. Elle mesure principalement les pourcentages d'arrivées de tel événement, ou le temps moyen de tel service [Véran 84],[Gelenbe 82]. Cet outil semble plus complémentaire que concurrent à ceux cités précédemment. Il permet, après avoir conçu et prouvé un modèle de façon qualitative, de valider la conception de manière temporelle. Cette validation peut se réaliser bien avant la programmation, contrairement aux autres méthodes qualitatives

(prototypes).

Les réseaux de petri offrent une alternative aux précédents outils de modélisation et de preuves. C'est un formalisme, basé sur une représentation graphique, modélisant sous forme d'un automate (action/événement) les processus étudiés. Les nombreux avantages de cette méthode sont:

- représentation graphique ;
- facilité de modélisation des processus parallèles;
- facilité de modélisation des mécanismes de synchronisation;
- existence de méthodes de preuve (invariants, méthodes structurelles, réduction, substitution, énumération d'état, etc...);
- associées à des outils automatiques: par exemple Rafaël [Behm 85], ARP [Haddad 87], ou OGIVE [Pradin 79] [Dufau 84].

De **nombreuses extensions**, vers des domaines très divers, prouvent que c'est un outil en pleine évolution:

- réseaux de petri à capacité ; (le nombre de marques dans une place est borné)
- réseaux de petri à priorité [Hack 75]; (le choix de franchissement des transitions déclenchables est fixé par priorité)
- réseaux de petri interprétés [Roucairol 74]; (on met en relation directe le modèle et le programme qu'il décrit, preuve de programmes)
- réseaux de petri à arcs inhibiteurs; (ce modèle possède le test de place vide, il étend la puissance des RdP)
- réseaux de petri multivalués; (ce modèle autorise le franchissement multiple des transitions, ce n'est qu'une facilité d'écriture des RdP normaux)
- réseaux de petri colorés [Jensen 81]; (les places et les transitions de ce modèle peuvent être colorés, ce qui permet une réduction de la taille du réseau)
- **réseaux de petri à prédicats** [Genrich 79]; (ce modèle possède des marques ayant une structure d'uplet)
- réseaux de petri évolués [Jensen 83];
- réseaux de petri relationnels [Reisig 83];
- réseaux de petri à files [Memmi 83]; (le modèle dont les places conservent l'ordre

d'arrivée des marques, possède un outil de modélisation associé [Behm 85])

- réseaux de petri temporisé [Chrétienne 83] [Ramchandani 73]; (modèle où l'on associe une contrainte temporelle aux places ou transitions)

- réseaux de petri stochastiques [Florin 85] [Zénié 85]; (ce modèle permet d'obtenir une analyse probabiliste sur la fréquence des événements)

- réseaux de petri numériques [Symons 80]; (ce modèle regroupe un nombre important d'extensions précédentes)

- etc...

On peut regretter ce nombre d'extensions, qui peut sembler excessif, il permet cependant d'obtenir un outil parfaitement adapté. Nous utiliserons les Réseaux de petri à prédicats, pour notre thèse. Nous présenterons une définition et la théorie de cet outil, ainsi que les raisons de notre choix ultérieurement.

2.3 Conclusion

Une mise en évidence des problèmes de fiabilité des logiciels et une rapide présentation d'un ensemble de méthodes et d'outils permettant d'y répondre, nous a permis de situer la place qu'occupe **les Réseaux de Petri**. Cette outil permet de **décrire (modéliser) et valider (prouver) les processus parallèles**, grâce à ses nombreuses extensions et au grand corpus de résultats déjà trouvés.

3. Les PROTOCOLES de Télécommunications

3.1 Introduction

Nous allons développer, au cours de cette présentation, les protocoles de télécommunication qui forment un bon exemple de processus concourants et asynchrones. Ils sont propres à mettre en évidence tous les écueils et problèmes provoqués par un parallélisme réel. Au début de notre étude, le protocole de Transport était encore en instance de discussion et avait été déposé en tant qu'avant-projet de norme; cette norme a été acceptée depuis. Il semble intéressant de modéliser son comportement, car le protocole comporte de nombreux mécanismes non encore validés dans leur totalité.

Nous allons tenter de situer l'objet de notre étude, la phase de transfert de la classe 3 de la couche Transport, définie dans la norme internationale pour l'interconnection des systèmes ouverts, parmi l'ensemble des protocoles de télécommunication. Nous en profitons pour donner l'ensemble des concepts et la terminologie définis par les organismes internationaux de normalisation des protocoles, notions et termes que nous serons amenés à employer au cours de notre discours.

Nous nous sommes ainsi intéressés dans les travaux préliminaires à notre thèse aux protocoles de télécommunications. Nous pouvons distinguer trois grands types de protocoles, que nous qualifions d'externes, par opposition aux protocoles internes applicables aux bus d'une même machine:

A- Les protocoles de communications pour réseaux nationaux ou internationaux. Ce sont ceux, dont les couches basses ont bénéficié les premières des efforts de normalisation, et ils servent actuellement de supports à un grand nombre de réseaux à travers le monde. Les recherches semblent se porter vers :

- L'étude des couches supérieures (couche Session [Estraillier 86], protocole d'appareil virtuel, cohérence [Architel 83], etc...) avec un ensemble de propositions

d'avant-projets pour les organisations de normalisation.

- La validation des protocoles de télécommunications existants ou à venir de manière rigoureuse [Diaz 82].

- Le développement d'une méthodologie globale permettant de spécifier, de prouver puis d'implanter aisément ces protocoles.

B- Les protocoles pour réseaux locaux, qui bénéficient de l'énorme développement de la bureautique [ECMA 14]. Les recherches en cours semblent se porter à la fois sur:

- Une recherche d'architecture et de protocole permettant une bonne mise à disposition de la puissance répartie et une vitesse et un débit adaptés aux nouvelles utilisations des réseaux [Hernandez 82].

- Une modélisation des protocoles déjà existants afin de les mieux spécifier et de les modéliser [Gressier 85].

C- La dernière voie de recherche s'intéresse aux transmissions de données propres aux satellites. Notamment les problèmes liés aux taux d'erreurs variables dus aux intempéries, aux délais importants dus à l'éloignement des partenaires, et à la technique particulière de transmission par diffusion [Valet 83][TELECOM 81].

3.2 La Norme O.S.I - les couches

La normalisation en matière d'interconnexion d'ordinateurs, fait suite à l'émergence d'un consensus au sein de l'"International Standard Organisation (I.S.O)" pour une architecture structurée en sept couches, grâce aux travaux de Zimmermann [Zimmermann 80]. Elle a permis de faire de rapides progrès quant à l'interconnexion des réseaux téléinformatiques.

Le principe de base est la définition pour chaque couche:

- Du protocole régissant le dialogue horizontal entre les entités appartenant à une même couche.
- Du service rendu à la couche supérieure au moyen des mécanismes du protocole.

En fait, on définit des services de plus en plus sophistiqués, pour les couches élevées, en s'appuyant sur les services rendus par les couches de niveaux inférieurs.

La définition normalisée des protocoles au niveau de chaque couche, permet, alors, une interconnexion de tous les systèmes respectant cette norme, et ce, en dehors de toute contrainte d'implantation.

Ce découpage a plusieurs avantages : la simplification de la conception, du fait de la spécificité et la modularité de la fonction de chaque couche; l'indépendance d'implantation de chaque couche. On doit uniquement :

- respecter le protocole de communication entre entités de même niveau (association horizontale).
- fournir les services définis par la norme à la couche supérieure (association verticale supérieure).
- utiliser les services définis par la norme de la couche inférieure (association verticale inférieure).

Nous allons énumérer succinctement les fonctions des 7 couches définies par l'"Open Systems Interconnection (O.S.I)" :

Ainsi **la couche 1** (Physique) définit les caractéristiques physiques (électriques et mécaniques) des supports et des signaux devant permettre la connexion, et permet d'échanger des bits significatifs.

Les fonctions internes remplies par la couche physique sont :

- établissement et libération de la connexion physique;
- transmission physique des bits.

Elle fournit le service suivant à la couche liaison:

- connexion assurant une transmission binaire, en mode duplex ou à l'alternat, entre les entités supérieures;
- conservation du séquencement des informations binaires;
- notification des défauts de fonctionnement;

La couche 2 (Liaison) permet d'échanger entre 2 sites de façon bilatérale en liaison point à point un ensemble de bits appelé trame, tout en s'assurant de la validité des données transmises [ISO HDLC].

Ses fonctions internes sont:

- établissement et libération de la connexion de liaison;
- établissement et synchronisation des trames ;
- détection et correction des erreurs de transmission;
- contrôle de flux point à point;

Les services fournis à la couche supérieure (Réseau) sont :

- conservation du séquencement;
- notification des erreurs irrécupérables;
- contrôle de flux;
- maintien de la qualité.

La **couche 3** (Réseau) autorise l'acheminement d'ensembles de bits appelés paquets, au travers d'un large et complexe réseau afin de permettre à 2 abonnés connectés à celui-ci de dialoguer ensemble [ISO 8348].

Les fonctions internes sont:

- multiplexage des connexions de la couche réseaux;
- routage;
- segmentation ou groupage des informations;
- conservation de la séquentialité;
- contrôle de flux.

Les services fournis regroupent:

- établissement et maintien des connexions;
- négociation des options (donnée express, réinitialisation);
- notification des erreurs;
- séquencement;
- contrôle de flux.

La **couche 4** (Transport), celle que nous modélisons, assure un service universel de transport des données contenues dans une structure, appelée message [ISO 8072] [ISO 8073]. Ce qui implique:

- un chemin bi-directionnel simultané, appelé connexion de transport, entre les deux partenaires du dialogue.
- un transfert transparent des données .
- une optimisation du transfert par le choix des meilleures connexions possibles.
- un contrôle du transfert de bout en bout pour un maintien optimum de la qualité.

Les fonctions internes sont donc:

- établissement et libération des connexion;
- traitement des adresses Réseau/Transport ;
- multiplexage des connexions Transport sur Réseau;
- segmentation ou groupage des informations;
- détection et correction des erreurs.

Ce qui permet d'obtenir le service voulu:
- établissement et maintien des connexions Transport;
- négociation des classes définissant un niveau de service;
- négociation des options (donnée express, crédit);
- contrôle de flux terminal.

La **couche 5** (Session) définit l'organisation des échanges et la structuration du dialogue entre applications [ISO 8326] [ISO 8327]. Cette couche prend en charge le contrôle des échanges d'informations entre entités distantes, et permet de synchroniser les opérations effectués sur les données.

La **couche 6** (Présentation) définit la représentation et la syntaxe des informations échangées [ISO 8822] [ISO 8823]. Elle comprend notamment la notion de représentation générique de données, qui permet la communication d'un ensemble d'applications hétérogènes.

La **couche 7** (Application) définit les mécanismes communs aux applications réparties, et la signification des informations échangées [ISO 8650]. Cette couche est la plus proche de l'utilisateur final.

Cette organisation se prête aisément à l'étude séparée de chaque couche. Nous intéressant au protocole de communication de la couche Transport, nous n'avons besoin, dans un premier temps, que de modéliser les services rendus par la couche Réseau, pour ensuite, modéliser le protocole de la couche Transport. La preuve à obtenir sera de montrer que le modèle respecte bien la norme du protocole et qu'il fournit bien à la couche supérieure (i.e. Session) les services attendus.

Prouver le protocole de la couche N, c'est démontrer que s'il respecte la norme ,en s'appuyant sur le service défini dans la norme de la couche N-1, il fournit le service attendu par la couche N+1.

3.3 Les Classes de la couche Transport

Il existe encore actuellement deux grandes propositions de norme pour la couche Transport.

La première, celle que nous étudions et qui a fait l'objet d'une norme définitive, est ce que l'on appelle "orientée connexion". Le protocole comporte trois phases, une phase d'établissement de connexion, une phase de transfert de données, une phase de rupture de connexion. La connexion établit une liaison logique et durable entre les partenaires de la transmission. Cette méthode est adaptée pour les liaisons à grandes distances (grand délai), et pour les applications de types transfert de fichiers (grand débit).

La deuxième proposition est appelée "sans connexion". Elle n'a qu'une seule phase. La connexion est établie à chaque échange d'informations. Cette méthode est adaptée aux réseaux locaux, et pour des applications de type service ou interrogation.

Pour décrire la couche Transport des protocoles de communication, nous nous appuyons sur la norme ECMA [ECMA 72]. Cette dernière prévoit pour la couche Transport cinq classes numérotées de 0 à 4 rendant des services de transport de messages de plus en plus complets.

Chaque classe doit normalement englober fonctionnellement tous les services des classes de niveaux inférieurs. Cette méthode permet au protocole de Transport de s'adapter aux différents types de service de la couche inférieure, pour offrir un service constant et adéquat pour l'exploitation de la couche supérieure.

Rien n'est plus divers, que les services assurés pendant le transfert de données:

- pour un réseau de type Télétex, le débit est faible et le délai de transfert non-crucial,
- dans un réseau de type Datagramme, les données qui y circulent, peuvent avoir

besoin d'un fort débit, d'une parfaite intégrité, mais la séquentialité des messages n'est pas assurée .

Nous allons énumérer l'ensemble des 5 classes définies dans la norme du protocole de Transport:

La classe 0, la plus simple définie dans cette couche, est réservée au service rendu pour le Télétex. Elle assure principalement la simple connexion des deux partenaires sans aucune sophistication du transfert de donnée, pas de multiplexage, ni concaténation, ni contrôle de flux, ni d'erreur, mais avec segmentation possible (un T-SDU éclaté dans plusieurs T-PDU).

La classe 1, autorise le multiplexage d'une connexion Réseau entre plusieurs connexions de Transport, la concaténation des T-PDU dans un N-SDU.

La classe 2 assure un contrôle de flux , qui régule le flot de T-PDU entre les deux entités situées aux extrémités de la connexion Transport. Un flot spécial, les données Express, permet de contourner ce contrôle de flux effectué sur les données normales.

La classe 3 permet de maintenir la qualité de la transmission en dépit de ses faiblesses. Elle gère le recouvrement des erreurs survenant au cours du transfert des données, et la reconfiguration nécessaire des connexions Réseaux si elles sont rompues.

La classe 4 permet le désordonnancement de ses messages pendant le transfert. Elle est recommandée dans le cas où le taux d'erreur est important. La détection des erreurs est basée sur le dépassement du temps nécessaire à la transmission des messages.

Nous avons choisi d'étudier la classe 3, car c'est celle qui est adaptée pour gérer un réseau de type TRANSPAC [STUR], du moins d'après les textes des normes. Elle comprend un grand nombre de fonctions caractéristiques de ce protocole, sans inclure un contrôle d'erreur rigoureux effectué à l'aide de temporisateur.

3.4 Les Phases de la couche Transport

Préalablement à toute poursuite de notre étude, nous allons mettre en évidence les différents états des connexions d'un protocole de communication. Nous établirons quatre états:

L'état déconnecté : Etat dans lequel il n'est pas attribué de correspondant à la connexion. Dans cet état, la partie de programme gérant le protocole est inactive.

L'état en cours de connexion : Etat dans lequel il existe un échange préliminaire entre les deux correspondants en vue d' établir une connexion. Ce premier échange permet aux deux partenaires de synchroniser et d'échanger les options de la connexion (état de transition qui permet de passer de l'état déconnecté à l'état connecté).

L'état connecté : La connexion ayant été établie entre les 2 correspondants, elle apparaît maintenant comme une simple liaison de bout en bout, qui permet d'échanger des données de manière transparente (cet état est aussi appelé phase de transfert).

L'état en cours de déconnexion : Un des deux partenaires (au moins) ayant décidé de rompre la connexion à cause de la fin de transmission ou le service à cause d'une perte irrécupérable de qualité sur la connexion inférieure, l'indique à son correspondant (cet état permet de passer de l'état connecté à celui déconnecté).

Pendant la phase de transfert, le protocole doit contrôler et assurer la communication et les services définis dans la norme de ce protocole.

- le contrôle de flux, qui permet de s'assurer que les partenaires ont les capacités de recevoir les messages qui leur sont envoyés.
- la récupération des erreurs ou perte de synchronisation, permet au protocole d'assurer une qualité de service constante malgré les aléas de la transmission.

- le transfert transparent des données sur la connexion.
- ..etc..

On s'aperçoit qu'en fait la phase de transfert est l'état prépondérant d'un protocole orienté connexion. C'est aussi l'état où la gestion du protocole est la plus critique. Et bien qu'en phase de transfert de données, il peut à tout moment survenir une déconnexion, on peut raisonnablement étudier séparément ces différentes phases.

3.5 Les Interfaces de la couche Transport

Nous avons vu que pour l'étude d'un protocole d'une couche N, il faut avoir défini les services fournis par les (ou à fournir aux) entités des couches inférieures "N-1" (supérieures "N+1"). Nous allons donc définir les primitives aux interfaces qui permettent d'assurer les services demandés.

Nous rappelons que la norme définit les informations qui transitent à travers l'interface qui sépare deux couches. La couche Transport échange avec le couche Session, ce que la norme appelle des T-SDU (Transport-service-data-unit). La couche Transport échange avec la couche Réseau des N-SDU (Network-service-data-unit).

Nous ferons abstraction dans tous les protocoles d'interface des problèmes de passage de l'information réelle. Notamment des T-IDUs (Transport-interface-data-unit) et des T-ICUs (Transport interface- command-unit) qui sont les messages échangés au niveau de l'interface Transport-Session, pour gérer le protocole vertical. Il en est de même pour les N-IDUs et les N-ICUs. On note que la norme ne spécifie en rien ces interfaces, car ils sont dépendants de l'implantation.

Notre définition des interfaces de la couche Transport avec les couches supérieures ou inférieures, est limitée strictement à la phase de transfert de ce protocole.

Interface Session : Dans la phase transfert du protocole de la couche Transport, les seuls messages échangés avec la couche Session sont les messages de données "data".

- T-SDU-data-req: la couche Session transmet (requête) à la couche Transport des données (data).
- T-SDU-data-ind: la couche Session reçoit (indication) de la couche Transport des données (data).

Interface Réseau : Dans la phase de transfert, le protocole de la couche Transport est susceptible d'échanger avec la couche Réseau, soit des données, soit des

signaux de réinitialisation "reset" indiquant des problèmes survenus sur la connexion Réseau.

- N-SDU-data-req: la couche Transport transmet (requête) à la couche Réseau des données (data).

- N-SDU-data-ind: la couche Transport reçoit (indication) de la couche Réseau des données (data).

- N-SDU-reset-ind: la couche Transport reçoit (indication) de la couche Réseau un reset, qui la prévient d'une perte de synchronisation intervenue dans le Réseau.

- N-SDU-reset-conf: la couche Transport prévient (confirmation) la couche Réseau de la bonne réception de son message de resynchronisation.

De manière similaire, la norme définit les messages échangés entre deux entités d'une même couche. Elle nomme cette structure les T-PDU (Transport-protocol-data-unit). Contrairement aux SDU, qui ne veulent être qu'une représentation abstraite des informations échangés entre les couches (primitive du Service), les PDU définissent la structure exacte (au bit près) des messages échangés à travers les connexions. Durant la phase de transfert de donnée, le protocole Transport transmet et reçoit :

- Les T-PDU-DT véhiculent les données sur la connexion Transport, ils contiennent tout ou partie d'un SDU de Transport.

- Les T-PDU-AK et T-PDU-REJ véhiculent les informations relatives aux acquittements et au crédit attribués à l'émetteur.

3.6 Les Fonctions de la couche Transport

Le protocole de transport de classe 3 comporte ces différentes fonctions :

Acquittement de bout en bout : Le message reçu est acquitté par le récepteur pour que l'émetteur s'assure de sa bonne arrivée au destinataire.

Envoi par anticipation des messages : L'émetteur, pour optimiser le débit de la voie de communication, est autorisé à envoyer plusieurs messages consécutivement, avant d'avoir reçu l'acquittement du premier message émis.

Contrôle de flux : Cependant le récepteur pilote l'émetteur à distance, par l'envoi de crédit qui autorise l'émetteur à envoyer un certain nombre de messages par anticipation, et ce, jusqu'à concurrence du crédit alloué .

Réquisition de crédit : C'est le point le plus original, mais aussi le plus problématique, de cette couche. Le récepteur pour des raisons internes, peut décider à tout moment de reprendre (réquisitionner) le crédit qu'il a déjà alloué à l'émetteur, ce qui permet d'avoir un contrôle de flux particulièrement souple, mais aussi peut engendrer des incohérences si l'on n'y prend pas garde.

Recouvrement d'erreur : Pour toute désynchronisation détectée (i.e. tout message inattendu ou perdu) l'entité détectrice prévient son partenaire, en lui envoyant un message spécifique, lui permettant de revenir dans un état cohérent.

Numérotation circulaire des messages : Toutes les fonctions précédentes appuient sur la numérotation de chaque message circulant sur la connexion Transport. Cette numérotation permet d'identifier chaque message afin de déterminer la perte ou la duplication éventuelles. Cependant la numérotation des messages est effectuée modulo N , N étant le nombre maximal de messages différents circulant au même moment sur la même connexion Transport.

D'autres fonctions sont réalisées par le service de la couche Transport, nous trouvons notamment :

A- Les traitements consistant principalement à regrouper et dégrouper des ensembles entre eux. Ils ne présentent que peu de difficultés :

- la segmentation des messages et leur désegmentation, éclate un SDU de Transport en plusieurs PDU de Transport qui seront rassemblés pour retrouver leur format d'origine à la destination.

- la concaténation et la dé-concaténation de messages, permet de concaténer plusieurs PDU de Transport dans un même SDU de Réseau. L'entité de Transport réceptrice doit alors séparer les différents PDU de Transport qui sont dans la même SDU de Réseau .

- le multiplexage et le démultiplexage de connexion, réalise le partage d'une connexion Réseau entre plusieurs connexions de Transport. Cette décision est prise au moment de l'établissement de la connexion Transport. On remarque, qu'alors, la fermeture de la connexion Transport n'entraine pas automatiquement la fermeture de la connexion Réseau.

B- Les traitements situés juste à l'interface entre les différentes couches. Ils sont dépendants de l'implantation, et sont toujours réalisés de manière séquentielle :

- la reconnaissance de la syntaxe des messages.

- la vérification de la cohérence des données par rapport à l'état courant de l'entité réceptrice.

- la gestion des interfaces avec les couches supérieures ou inférieures.

C- Les deux phases indépendantes de la phase de transfert de données:

- l'ouverture des connexions de Transport, et leur répercution au niveau des

connexion Réseau.

- La négociation de la classe de protocole et le choix des options possibles (Donnée Express, longueur optimal des T-PDUs, crédit initial, etc...).
- La mise en correspondance des adresses Transport avec les adresses Réseau.
- la fermeture des connexions de Transport.

D- Le traitement des messages Express n'appartient pas, à proprement parler, à la phase de transfert des messages de données. Il en est indépendant pour la gestion du contrôle de flux, du stockage, et des retransmissions :

- la réception ou l'envoi de messages Express et leur acquittement.
- la transmission de messages Express.

3.7 Conclusion

Le deuxième chapitre de cette première partie a mis en évidence les problèmes de fiabilité des systèmes informatiques, et nous a donné un aperçu rapide des outils permettant de les résoudre. Ce troisième chapitre aborde les systèmes qui composent les réseaux de transmission téléinformatiques, ces systèmes étant exemplaires quant à leur parallélismes et l'importance de leur sûreté de fonctionnement.

Nous venons de situer la couche Transport parmi les autres couches, d'après les normes internationales sur les télécommunications. Cette revue nous a permis de définir les termes et les notions employés dans ces normes, termes et notions que nous avons repris dans notre thèse. Nous en définissons d'autres au cours du chapitre suivant qui développe notre méthode de modélisation et de validation.

Nous avons vu les notions de couches, de mode de connexion, de phase d'un protocole, et de classe. Nous avons présenté succinctement les nombreuses fonctionnalités de la couche Transport. Nous reprenons plus en détail ces fonctionnalités dans la troisième partie de notre thèse, en les étudiant et en les modélisant.

Deux points issus de la norme d'interconnexion des systèmes ouverts sont à retenir plus particulièrement :

La notion de couche permet, pour l'étude d'une couche de faire abstraction des couches non contiguës;

La notion de service défini le comportement externe de la couche, permettant ainsi, de faire abstraction de son protocole (comportement interne) et de son implantation.

Ces deux points sont à l'origine de la méthodologie, développée au chapitre suivant.

4. METHODOLOGIE de VALIDATION des systèmes structurés en couches

4.1 Introduction

Dans ce chapitre, nous développons les différents points sur lesquels sont basés notre méthodologie pour modéliser et valider les systèmes structurés en couches. Nous sommes partis des méthodes bien connues de décomposition et d'abstraction d'un système. Nous utilisons ces principes pour la modélisation. Nous introduisons donc des décompositions et des abstractions de modèles.

Nous définissons la notion de concordance de modèle pour prouver qu'un modèle décrit correctement le système étudié. Nous nous assurons que le système n'est, ni sur-modélisé, ni sous-modélisé. La concordance est basée sur les deux notions : de description d'un système et de propriétés de cette description. Pour prouver la concordance, nous établissons une relation entre les propriétés de la description du système et les propriétés du modèle du système. La concordance de modèle oblige à définir, par l'établissement des propriétés de la description du système, ce que l'on veut modéliser.

Enfin, et c'est le principe fondamental de notre méthodologie, nous avons défini la notion d'adéquation de service. Celle-ci est basée sur les notions de couches, d'abstractions et de services. Conformément à la définition d'une couche, nous prouvons que la réalisation d'un système et le service de ce même système sont perçus par la couche supérieure comme ayant un comportement et des propriétés identiques, ce qui est rendu nécessaire par la définition même du service d'une couche. La preuve de l'adéquation consiste à comparer le modèle d'une réalisation du système et le modèle du service du même système.

D'une part, nous prouvons que la réalisation d'un système fournit bien le service demandé, d'autre part nous justifions le principe qui permet la conception et la modélisation d'une couche en faisant abstraction de toutes les couches non contiguës à celle étudiée.

4.2 Structure d'un système

4.2.1- Système et Couches

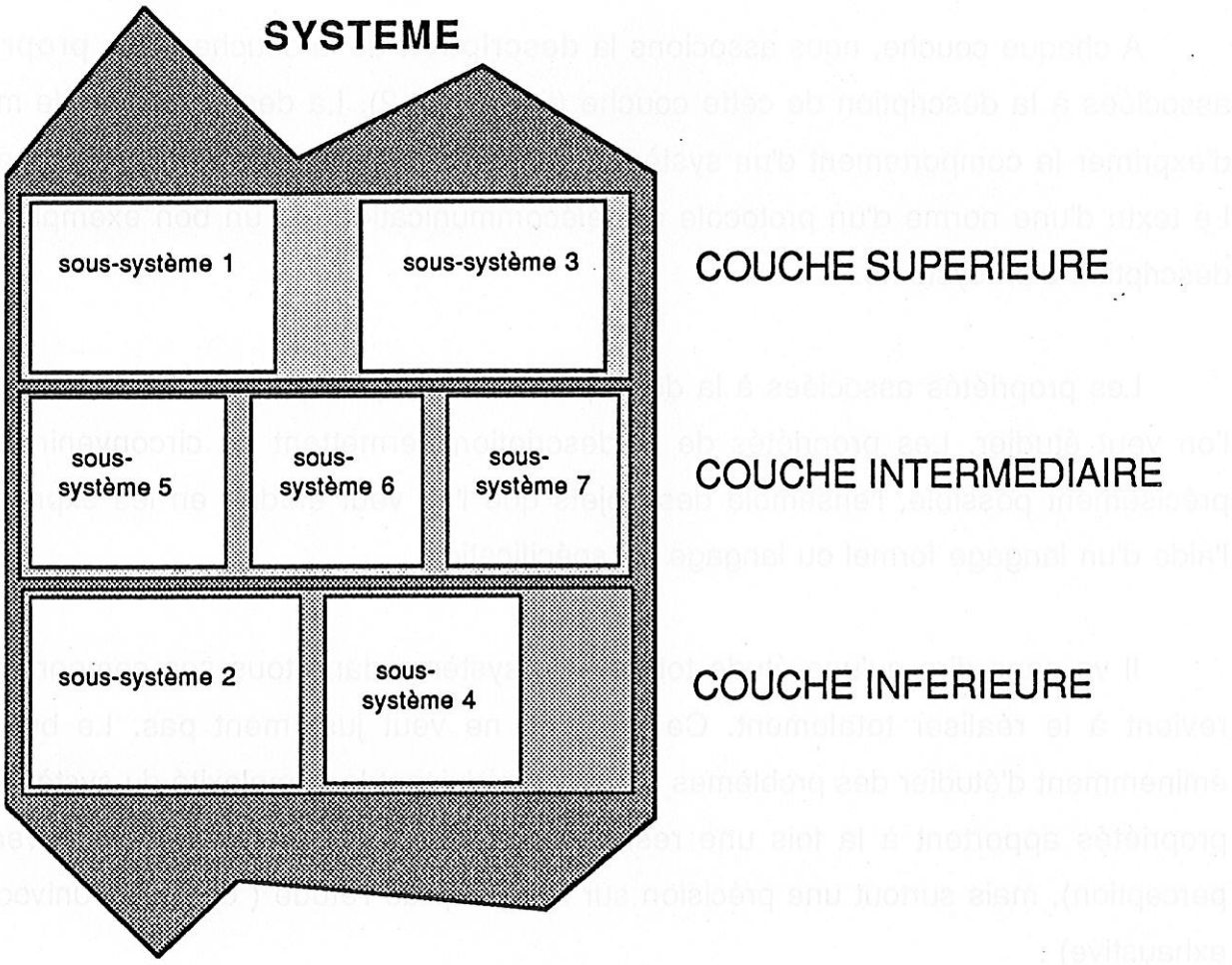
Depuis Descartes, nous savons que pour étudier un problème complexe, il suffit de le découper en plusieurs sous-problèmes que l'on sache résoudre. C'est pourquoi nous appliquons cette méthode à la modélisation et la validation des systèmes concourants tels que les protocoles de télécommunications.

Tout système général peut être décomposé en sous-systèmes. Chaque sous-système est responsable d'une ou plusieurs fonctionnalités permettant d'obtenir une partie du fonctionnement du système général.

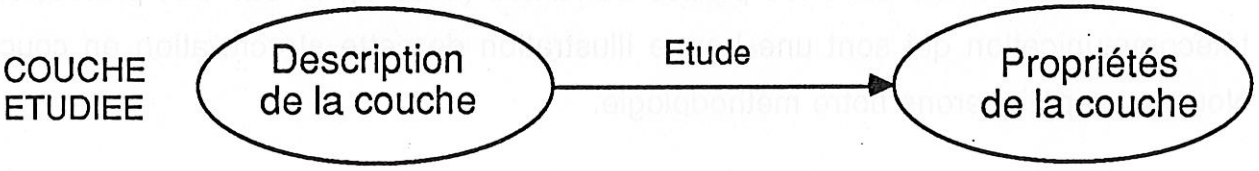
A des fins de structuration, on regroupe plusieurs sous-systèmes en un ensemble appelé couche. Chaque couche ne doit être en contact (échanger des informations) qu'avec deux couches qualifiées d'inférieure et de supérieure par rapport à la première (figure A-4.1). La couche est le plus petit ensemble de sous-systèmes qui ne communiquent qu'avec les sous-systèmes des couches directement inférieure ou supérieure.

Chaque couche utilise les fonctionnalités de sa couche inférieure pour produire les fonctionnalités destinées à sa couche supérieure. Donc les couches basses ont de faibles fonctionnalités, tandis que les couches hautes fournissent des fonctionnalités plus sophistiquées. De ce fait, la fonctionnalité de la dernière couche (la plus haute) correspond à la fonctionnalité du système général.

Cette structuration, permet pour comprendre ou concevoir une couche, d'ignorer toutes les autres couches hormis celles strictement inférieure et supérieure.



- Figure A-4.1 -
Structuration des Systèmes



- Figure A-4.2 -
Description et Propriérés

4.2.2 Description et Réalisation

A chaque couche, nous associons la **description** de la couche et les **propriétés** associées à la description de cette couche (figure A-4.2). La description est le moyen d'exprimer le comportement d'un système. Cette description utilise le langage naturel. Le texte d'une norme d'un protocole de télécommunication est un bon exemple d'une description d'un système.

Les propriétés associées à la description sont l'expression des mécanismes que l'on veut étudier. Les propriétés de la description permettent de circonvenir le plus précisément possible, l'ensemble des objets que l'on veut étudier en les exprimant à l'aide d'un langage formel ou langage de spécification.

Il va sans dire qu'une étude totale d'un système dans tous ses comportements revient à le réaliser totalement. Ce que l'on ne veut justement pas. Le but étant éminemment d'étudier des problèmes précis en réduisant la complexité du système. Les propriétés apportent à la fois une restriction (vue partielle d'un système, niveau de perception), mais surtout une précision sur le champ de l'étude (définition univoque et exhaustive) .

Puisqu'une couche dépend, et est construite, uniquement à partir de la couche inférieure, l'ensemble des propriétés d'une couche ne dépend que de l'ensemble des propriétés de la couche inférieure.

Nous étudierons dans les parties suivantes (B et C), le cas des protocoles de télécommunication qui sont une bonne illustration de cette structuration en couches. Nous leur appliquerons notre méthodologie.

4.3 Structure d'une couche

4.3.1 Service et Réalisation

Une bonne description d'un système, consiste à la fois à ne pas sous-modéliser en omettant la description d'un élément du système, et à ne pas sur-modéliser en introduisant dans la description un élément que le système ne nécessite pas.

Cependant, on peut vouloir décrire un même système à des niveaux de perception différents. Notamment, on peut facilement distinguer deux descriptions d'un même système (figure A-4.3):

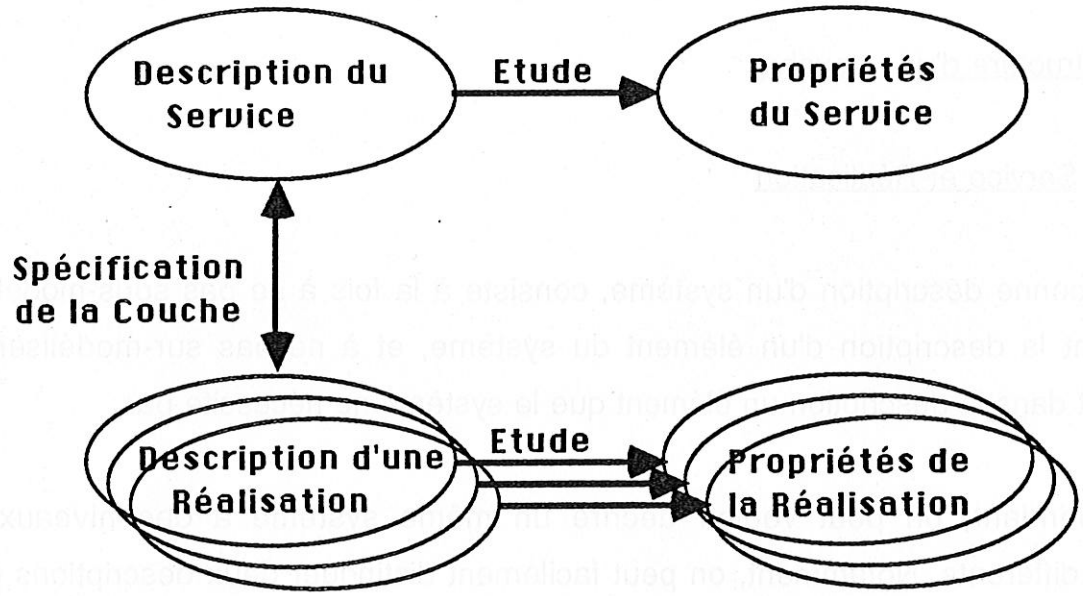
- La vue interne ou **réalisation**, qui est la description interne des mécanismes d'un système. Il peut y avoir plusieurs réalisations d'un même système dépendant, soit du niveau de description des mécanismes étudiés, soit des différents choix possibles pour les mécanismes internes d'un même système.

- La vue externe ou **service**, qui n'est que la description du comportement d'un système vue de l'extérieur de ce système (principe de la boite noire). Cette description doit être indépendante des réalisations et doit être unique pour son utilisation par la couche supérieure.

Le concept de service développé pour la normalisation des télécommunications informatiques apporte de nombreux avantages, qui ont permis de le faire adopter par les organismes de normalisation internationaux [Vissers 84].

La normalisation des protocoles téléinformatiques pour la communication des systèmes ouverts comporte actuellement pour chaque couche une spécification composée des deux descriptions suivantes:

- La description du service de la couche;
- La spécification du protocole de la couche.



- Figure A-43 -
Réalisation et Service

4.3.2 Adéquation de Service

Néanmoins, il convient de respecter certaines relations entre les différentes descriptions, notamment au sujet des propriétés associées à chacune de ces descriptions.

Si dans la description d'un système, on spécifie d'une part le service rendu et d'autre part le comportement interne du système, l'ensemble des propriétés de la première description doit se retrouver inclus dans l'ensemble des propriétés de la deuxième description. Ainsi, nous nous assurons que le système ayant le comportement interne défini par la deuxième description, offre bien le service nécessaire à la réalisation de la couche supérieure, tel que le définit la première description.

Les propriétés que la réalisation possède et que le service ne possède pas, sont issues des mécanismes internes.

Puisque tout système peut avoir plusieurs descriptions (internes, externe, etc...), on peut définir deux ensembles de propriétés que chaque système possède.

- Les propriétés d' une des réalisations du système sont propres à son comportement interne. Elles dépendent du niveau de description et des choix d'implantations.

- Les propriétés du service rendu par le système doivent être communes à l'ensemble des descriptions du système.

Nous avons vu au paragraphe précédent que toute couche dépend de la couche inférieure. Maintenant, si nous définissons le service d'une couche comme la description externe de la couche, toute couche ne dépend plus que du service de la couche inférieure.

Dans une structuration en couches, la réalisation d'une couche doit, sous peine de sur-définition, ne dépendre que du service rendu par sa couche inférieure. De même, les propriétés d'une couche ne doivent dépendre que des propriétés qui illustrent le service rendu par la couche inférieure. La réalisation d'une couche utilise indirectement la réalisation de la couche inférieure par l'intermédiaire du service, mais en aucun cas ne doit dépendre directement de cette réalisation.

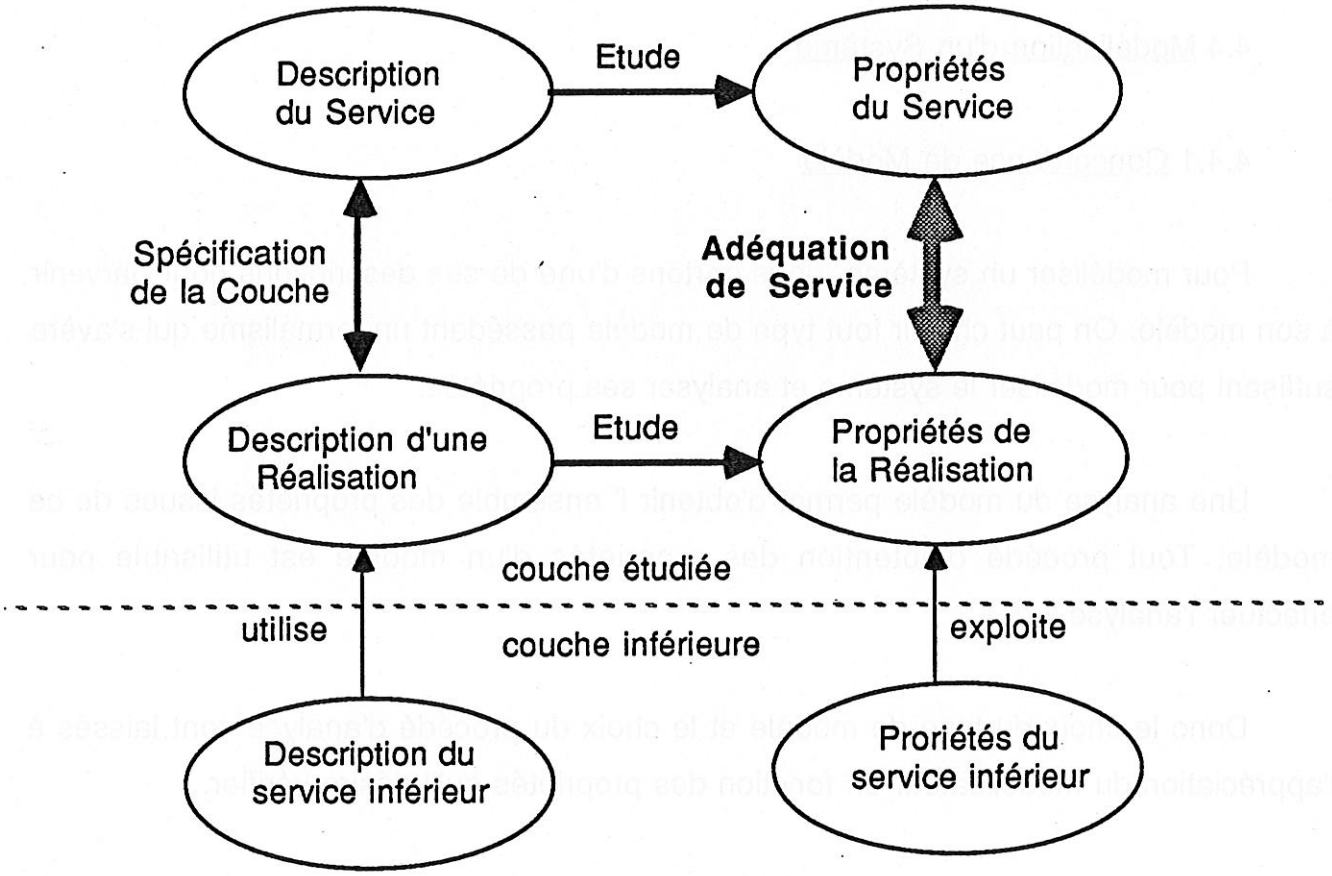
La notion de service crée une nouvelle abstraction. Maintenant une couche est non seulement indépendante de l'ensemble des couches non-contiguës (abstraction du premier niveau dû à la structuration en couches), mais elle est aussi indépendante des mécanismes internes (réalisation) des couches contiguës (abstraction de deuxième niveau dû à la notion de service).

Nous sommes donc amenés à définir ce que nous appelons **l'adéquation de service** pour établir la conformité entre le service et une des réalisations d'une couche.

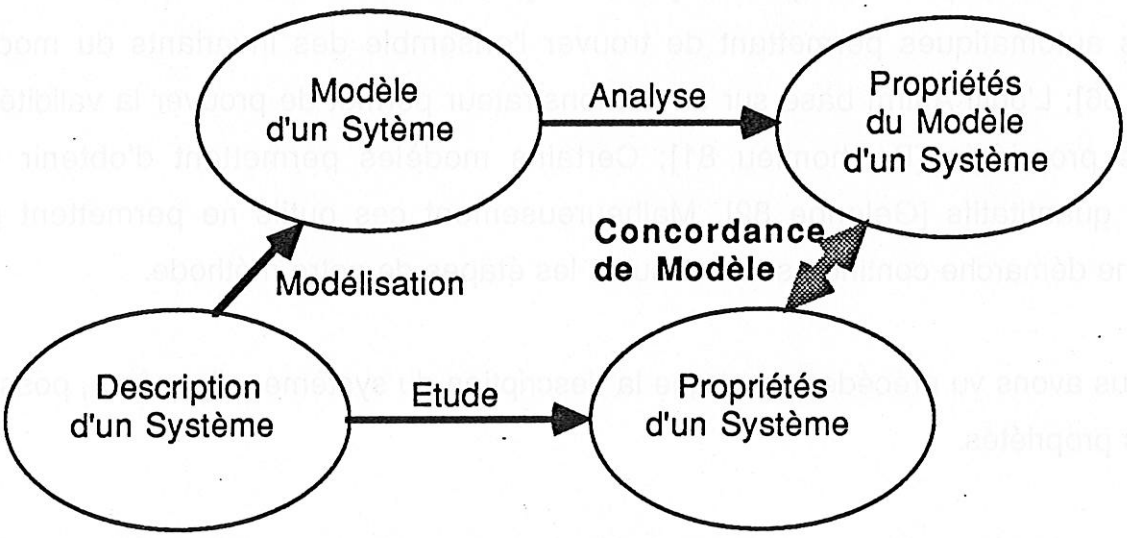
A partir de la spécification d'une couche fournissant une description du service et une **description d'une réalisation** possible utilisant la **description du service** de la couche inférieure, après une étude permettant d'**obtenir les propriétés** de chacune des deux descriptions, il convient d'établir l'adéquation entre les propriétés du service et celles de la réalisation exploitant les propriétés du service de la couche inférieure (figure A-4.4).

L'adéquation de service revient à prouver l'égalité sur les propriétés :

Service N = Protocole N (Service N-1)



- Figure A-4.4 -
Adéquation de Service



- Figure A-4.5 -
Concordance de Modèle

4.4 Modélisation d'un Système

4.4.1 Concordance de Modèle

Pour modéliser un système, nous partons d'une de ses descriptions pour parvenir à son modèle. On peut choisir tout type de modèle possédant un formalisme qui s'avère suffisant pour modéliser le système et analyser ses propriétés.

Une analyse du modèle permet d'obtenir l' ensemble des propriétés issues de ce modèle. Tout procédé d'obtention des propriétés d'un modèle est utilisable pour effectuer l'analyse.

Donc le choix du type de modèle et le choix du procédé d'analyse sont laissés à l'appréciation du modélisateur en fonction des propriétés qu'il désire vérifier.

Plusieurs types d'outils de modélisation peuvent être employés, en fonction de ces outils, les étapes de modélisation et d'analyse sont plus ou moins aisées. Par exemple, l'outil Rafaël permet d'obtenir automatiquement un modèle exprimé par un réseau à files à partir d'une description du système [Behm 85]; Les réseaux de Petri possèdent des procédés automatiques permettant de trouver l'ensemble des invariants du modèle [Haddad 86]; L'outil Affirm basé sur un démonstrateur permet de prouver la validité de certaines propriétés [Berthomieu 81]; Certains modèles permettent d'obtenir des résultats quantitatifs [Gelenbe 82]. Malheureusement ces outils ne permettent pas d'avoir une démarche continue suivant toutes les étapes de notre méthode.

Nous avons vu précédemment que la description du système, elle-même, possède certaines propriétés.

Nous obtenons alors un moyen simple de prouver la **concordance du modèle** avec la description du modèle, en établissant la concordance des propriétés issues du modèle et des propriétés issues de la description du système.

C'est ce que nous appelons la concordance du modèle (figure A-4.5).

En particulier, nous disposons de deux descriptions, celle du service du système et une d'une réalisation du système. Nous pouvons donc modéliser ces deux descriptions et établir la concordance de chacun des deux modèles avec leurs descriptions.

Les outils permettant de prouver la concordance de modèle sont difficiles à trouver. Cela tient au fait que les propriétés du système et les propriétés du modèle du système ne sont pas toujours exprimées dans le même langage. L'utilisation d'outils homogènes durant les phases de conception et de validation, ou une traduction automatique entre les deux langages permet de résoudre ce problème particulier.

4.4.2 Adéquation de service

La modélisation peut s'appliquer à toutes les descriptions du même modèle. Ainsi nous avons vu que si nous disposons de deux descriptions, l'une du service du système, l'autre d'une réalisation du système, nous retrouvons les propriétés de la première description dans la deuxième description.

De même, nous savons que les deux modèles issus des deux descriptions du même système possèdent chacun un ensemble de propriétés. Nous devons retrouver l'ensemble des propriétés issues du modèle du service du système dans l'ensemble des propriétés issues du modèle d'une réalisation du système.

Nous pouvons donc prolonger la notion d'adéquation de service, exprimé dans le précédent paragraphe sur les propriétés du système, et maintenant l'exprimer sur les propriétés du modèle du système (figure A-4.6).

Plusieurs équipes travaillent actuellement sur le sujet de l'adéquation de service.

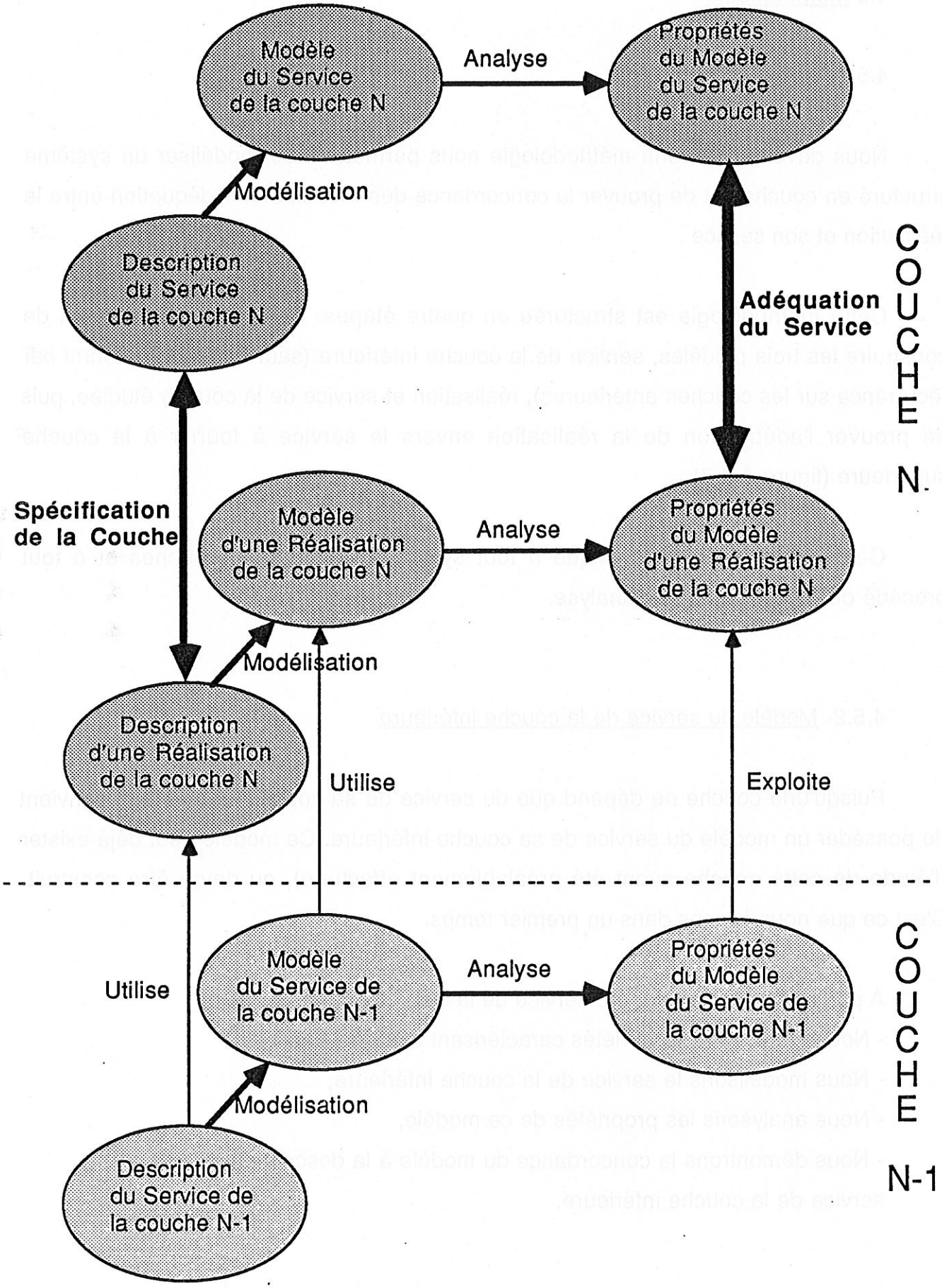
Nous pouvons répertorier trois méthodes appliquées aux Réseaux de Petri:

- Une première technique s'applique directement sur le modèle, par transformation d'un modèle jusqu'à ce qu'il soit identique à un autre. Cette technique utilise la notion de réduction développée par [Berthelot 83] et utilisée dans l'outil OVIDE.

- Une deuxième technique compare l'ensemble du comportement des deux modèles pour en déduire l'équivalence. Une technique compare les graphes des marquages accessibles des deux modèles [Bourguet 86].

- Nous avons été contraints de choisir une troisième voie. Les deux techniques précédentes, soit ne conservent pas toutes les propriétés par transformation, soit nécessitent un comportement fini, soit enfin ne peuvent être directement appliquables aux réseaux de Petri à prédicats qui nous sont rendus nécessaires par la complexité du système à modéliser. Nous avons donc développé "à la main" les propriétés et ainsi établi les concordances des modèles et l'adéquation de service. Ce procédé est laborieux, mais les développements dans l'avenir, soit des méthodes déjà appliquées aux RdP ordinaires, soit de nouveaux outils de preuves automatiques, pourront apporter une plus grande efficacité.

- Figure A-4.6 - Adéquation de service à partir des modèles



4.5 Méthodologie

4.5.1- Introduction

Nous développons une méthodologie nous permettant de modéliser un système structuré en couches et de prouver la concordance des modèles et l'adéquation entre la réalisation et son service .

Cette méthodologie est structurée en quatre étapes. Ces étapes permettent de construire les trois modèles, service de la couche inférieure (sauf s'il est déjà fourni par récurrence sur les couches antérieures), réalisation et service de la couche étudiée, puis de prouver l'adéquation de la réalisation envers le service à fournir à la couche supérieure (figure A-4.7).

Cette méthodologie s'applique à tout système structuré en couches et à tout procédé de modélisation et d'analyse.

4.5.2- Modèle du service de la couche inférieure

Puisqu'une couche ne dépend que du service de sa couche inférieure, il convient de posséder un modèle du service de sa couche inférieure. Ce modèle peut déjà exister (l'étude de cette couche ayant été préalablement effectuée), ou devra être construit. C'est ce que nous faisons dans un premier temps.

A partir de la description du service de la couche inférieure,
- Nous étudions les propriétés caractérisant cette description,
- Nous modélisons le service de la couche inférieure,
- Nous analysons les propriétés de ce modèle,
- Nous démontrons la concordance du modèle à la description du service de la couche inférieure.

A cette étape, nous venons d'établir un modèle du service de la couche inférieure en concordance avec sa description.

4.5.3- Modèle d'une réalisation de la couche étudiée

Nous pouvons, maintenant modéliser la couche qui nous intéresse. Nous construisons un premier modèle d'une réalisation de la couche étudiée.

A partir de la description de la réalisation de la couche étudiée,
- Nous étudions les propriétés caractérisant cette description,
- Nous modélisons la réalisation de la couche étudiée,
- Nous obtenons après analyse les propriétés de ce modèle,
- Nous démontrons la concordance du modèle à la description de la réalisation de la couche étudiée.

Cette deuxième étape nous permet d'établir un modèle d'une réalisation de la couche étudiée, modèle que nous prouvons conforme à la description.

4.5.4- Modèle du service de la couche étudiée

Il nous reste encore à prouver que cette réalisation de la couche étudiée entraine le service nécessaire à la couche supérieure.

C'est pourquoi nous allons étudier les propriétés du service de la couche étudiée en construisant un modèle du service.

A partir de la description du service de la couche étudiée,
- Nous étudions les propriétés caractérisant cette description,
- Nous modélisons le service de la couche étudiée,
- Nous obtenons après analyse les propriétés de ce modèle,
- Nous démontrons la concordance du modèle à la description du service de la couche étudiée.

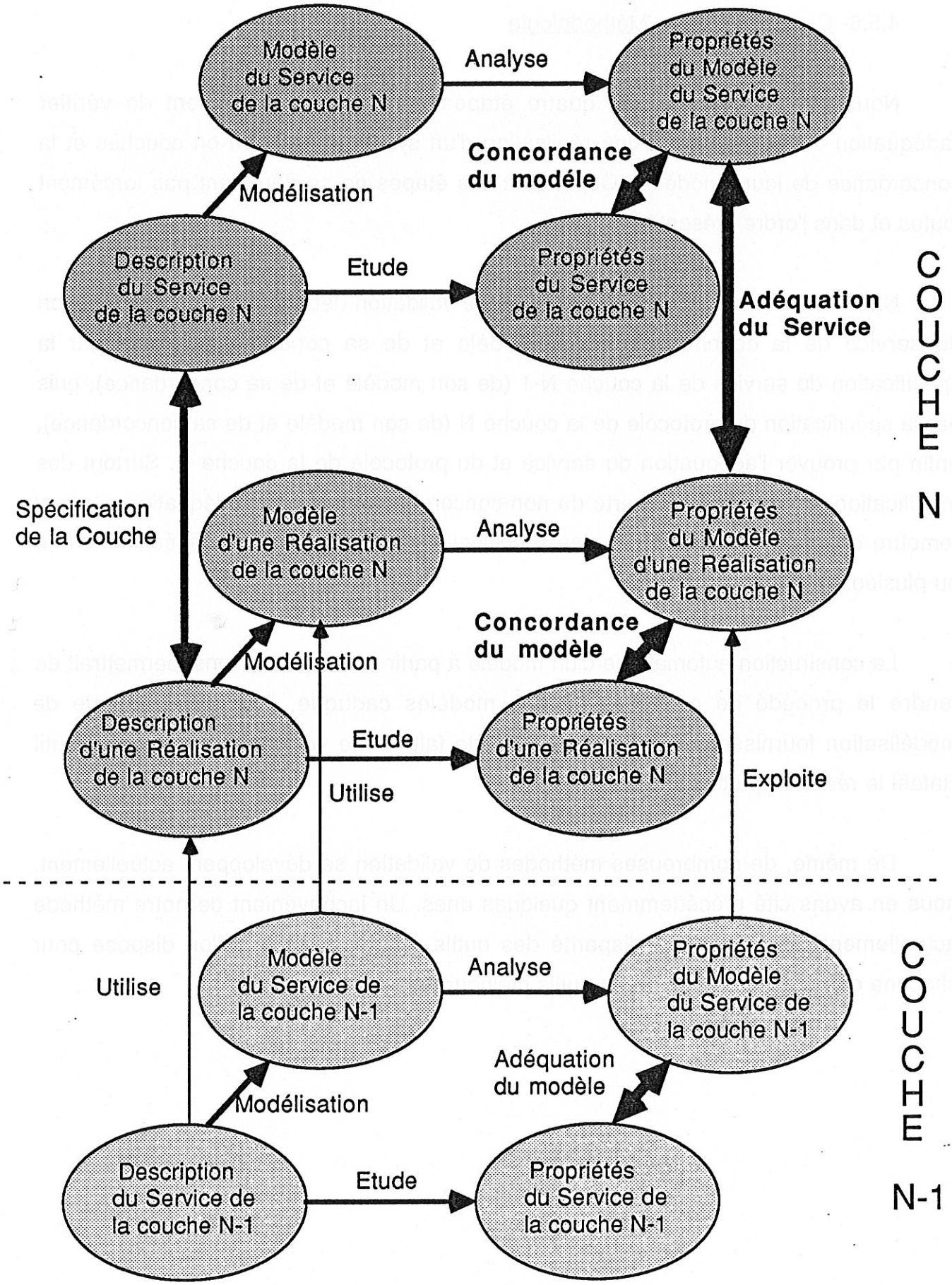
Cette troisième étape permet d'établir un modèle du service de la couche étudiée dont nous avons prouvé la concordance à la description.

4.5.5- Adéquation de service de la réalisation de la couche étudiée

A partir du modèle de la réalisation de la couche étudiée, des propriétés du modèle du service de la couche inférieure, et après analyse, nous obtenons un ensemble de propriétés qui caractérisent les fonctionnalités externes de la couche étudiée.

Ces propriétés doivent être celles de la description du service de la couche étudiée. Elles doivent donc se retrouver parmi les propriétés associées au modèle du service de la couche étudiée. L'adéquation de service établit cette correspondance et prouve ainsi que la réalisation de la couche étudiée a les fonctionnalités nécessaires à la couche supérieure.

- Figure A-4.7 - Méthodologie



4.5.6- Conclusion de la Méthodologie

Nous avons présenté, ici, quatre étapes permettant globalement de vérifier l'adéquation d'un service et d'une réalisation d'un système structuré en couches et la concordance de leurs modèles. Cependant ces étapes ne se déroulent pas forcément toutes et dans l'ordre présenté :

Une autre démarche de conception ou de validation débuterait par la spécification du service de la couche N (de son modèle et de sa concordance), puis par la spécification du service de la couche N-1 (de son modèle et de sa concordance), puis par la spécification du protocole de la couche N (de son modèle et de sa concordance), enfin par prouver l'adéquation du service et du protocole de la couche N. Surtout des modifications liées à la découverte de non-concordance ou de non-adéquation peuvent remettre en cause les modèles ou les spécifications, et nécessiter la réexécution d'une ou plusieurs étapes.

La construction automatique d'un modèle à partir des spécifications, permettrait de rendre le procédé de concordance des modèles caduque, l'outil automatique de modélisation fournissant cette concordance de fait. Cette voie semble d'avenir, l'outil Rafaël le réalisant partiellement.

De même, de nombreuses méthodes de validation se développent actuellement, nous en avons cité précédemment quelques unes. Un inconvénient de notre méthode actuellement réside dans la disparité des outils offerts, alors que l'on dispose pour chacune des étapes de nombreux outils malheureusement disparates.

4.6 Environnement de la méthodologie

4.6.1- Introduction

Il existe de manière éparse de nombreuses recherches sur les méthodes pour rendre fiable la conception et la réalisation de systèmes concourants. Cette recherche est en grande partie orientée vers et par les protocoles de télécommunication.

Les organismes internationaux se sont aperçus ces dernières années de la nécessité d'exprimer les normes non plus uniquement à l'aide d'un langage informel (la langue naturelle : l'anglais), mais avec un langage formel (langage de spécification ou Formal Description Technics). Des groupes de travail se sont formés pour une normalisation des langages présentés par différentes équipes ([ESTELLE 85], [LOTOS 85]...).

En fait, le langage formel n'est pas le but, mais un moyen. Le moyen d'avoir un langage univoque favorisant la communication. La conformité au langage, donc la levée des ambiguïtés, peut être confiée à un compilateur.

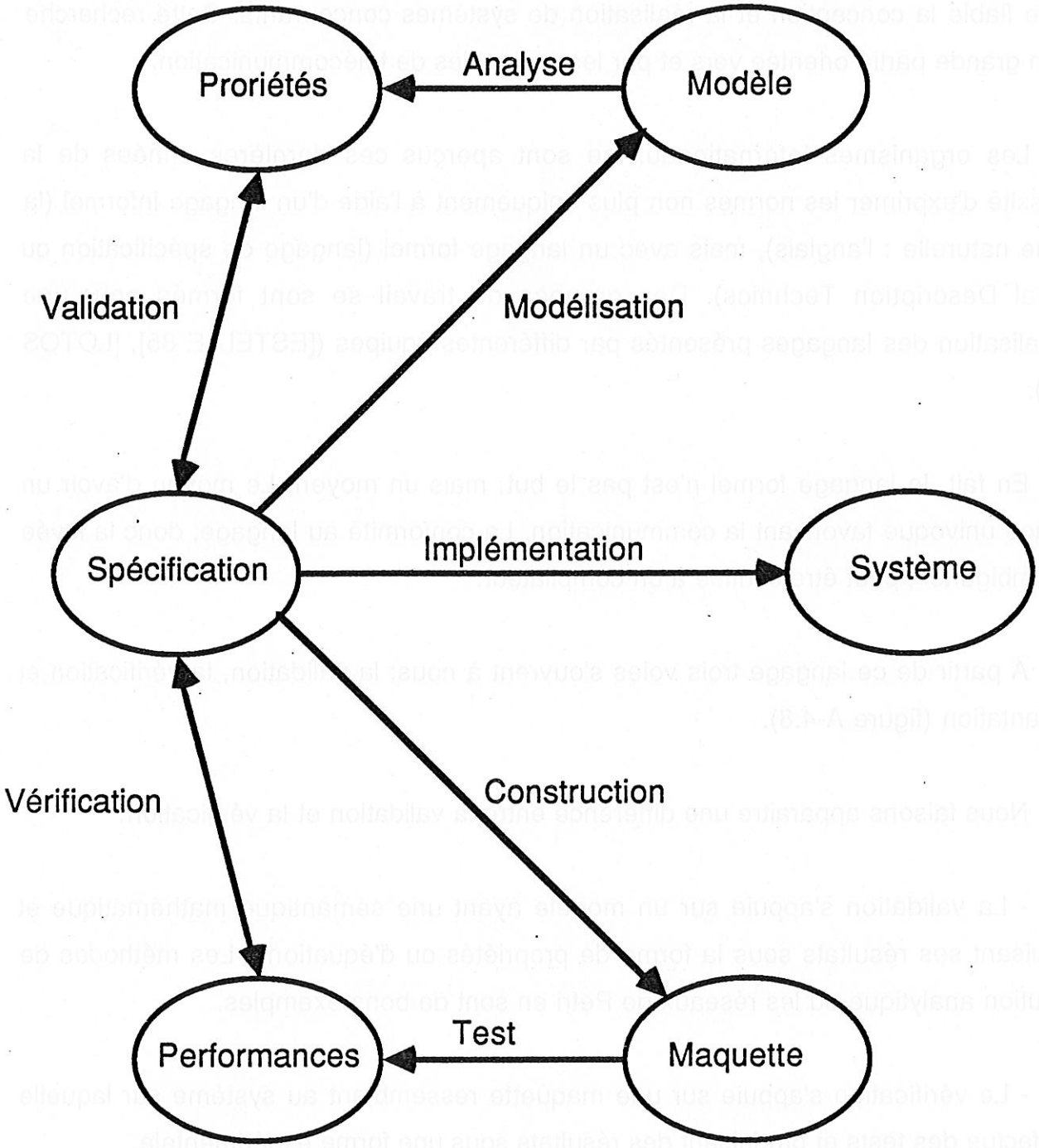
A partir de ce langage trois voies s'ouvrent à nous: la validation, la vérification et l'implantation (figure A-4.8).

Nous faisons apparaitre une différence entre la validation et la vérification:

- La validation s'appuie sur un modèle ayant une sémantique mathématique et produisant ses résultats sous la forme de propriétés ou d'équations. Les méthodes de résolution analytique ou les réseaux de Petri en sont de bons exemples.

- Le vérification s'appuie sur une maquette ressemblant au système sur laquelle on effectue des tests et produisant des résultats sous une forme expérimentale.

- Figure A-4.8 -
Environnement



4.6.2- Validation, Vérification et Implantation

Les deux techniques de validation et de vérification sont éminemment complémentaires, la première est orientée conception et est théorique, la deuxième est orientée implantation et de ce fait est plus pragmatique.

On note que ces deux techniques se réfèrent toutes les deux aux spécifications. Elles peuvent toutes les deux représenter tout ou partie du système, avec plus ou moins d'abstraction. Toutefois la vérification peut utiliser (non sans parfois le perturber) le système réel comme maquette, du moins en phase de développement et de recette.

La validation prouve formellement les spécifications du système. Il faut pour ce faire adjoindre au langage de spécification une sémantique mathématique, qui permettrait de déduire automatiquement le modèle des spécifications. La validation permet de vérifier, que le système n'a pas été mal spécifié (état inaccessible, non-connexité, état non-spécifié, ...), qu'il ne peut se bloquer (vivacité), ou enfin qu'il suit telle loi de service. Il existe actuellement deux grands modèles, les automates à états, et les réseaux de Petri. Ces deux modèles bénéficient de nombreuses extensions (files, types abstraits, etc...).

Pour la vérification (ou test), de même, on doit adjoindre au langage de spécification une machine abstraite (si elle n'est pas fournie par l'implantation). Pour les protocoles de télécommunications, différentes techniques de tests sont utilisées [ISO Test]. La vérification démontre le bon fonctionnement, vérifie la conformité quantitative de la maquette par rapport aux spécifications du système. Elle peut s'effectuer sur une maquette (ou un modèle) en phase de conception ; sur une partie du système en phase de développement et d'intégration; en fin de réalisation pour la recette du système; et même après, en phase de maintenance pour vérifier une évolution.

La vérification permet de détecter certains comportements erronés et d'évaluer les performances du système. Elle nécessite des outils de simulation permettant de générer

une partie du système ; un générateur de scénario de test parfois issu des spécifications du système ; d'outils de connexion, d'abstraction et d'arbitrage pour gérer les communications [Favreau 86] [Ansart 82] .

L'implantation automatique de protocole commence à devenir une voie de recherche, mais elle reste partielle. Elle nécessite d'adjoindre aux spécifications les contraintes et les choix d'implantations qui ont pu être éprouvés au cours des phases précédentes.

4.7 CONCLUSION

Nous venons de développer une méthodologie qui permet à partir d'un système structuré en couches de le modéliser et de prouver la concordance de ses modèles et l'adéquation de son service.

La concordance de modèle permet de valider le modèle par rapport à la description du système. L'adéquation de service permet de valider une réalisation d'un système par rapport au service demandé. Ces deux procédés permettent d'obtenir des modèles qui acceptent l'abstraction et la décomposition.

Ces deux procédés s'insèrent dans le champ plus général de la validation de système, qui, elle même, fait partie d'une méthode globale de conception et de réalisation des systèmes. Cette dernière méthode s'appuie sur quatre étapes: spécification, validation, vérification, et implantation.

Nous allons appliquer notre méthodologie à l'étude de la couche Transport des protocoles de télécommunication. L'outil de modélisation est les réseaux de Petri qui offre un modèle d'une grande puissance. Nous exploitons les nombreux résultats basés sur la théorie des réseaux de Petri pour effectuer l'analyse des modèles. Faute d'avoir un ensemble d'outils intégré à notre méthode, la modélisation, l'analyse, la concordance et l'adéquation seront effectuées "à la main". Néanmoins nous espérons que l'application que nous allons faire sera assez convaincante, pour que la poursuite des efforts de recherche nous offre dans le futur une méthode pourvue d'outils intégrés.

5. Les RESEAUX de PETRI à PREDICATS

5.1 Introduction

L'énumération des états d'un ensemble de processus parallèles et coopérants se heurte rapidement à un accroissement combinatoire de ces états et de leurs relations dû à l'asynchronisme des processus, ce qui interdit de les exploiter utilement.

Toutefois de nombreuses approches existent pour pallier cet inconvénient, une des plus répandues à ce jour sont les réseaux de Pétri, qui basés sur une représentation graphique permettent de modéliser synthétiquement les processus, puis de prouver le modèle obtenu au moyen de raisonnements formels, qui sont associés à la définition des Réseaux de Pétri.

Les Réseaux de Pétri présentent les avantages suivants:

Il sont bien adaptés à la modélisation de protocole de communication, car ils rendent aisément compte des synchronisations des différents événements et du parallélisme des processus concourants, comme le montre leurs précédentes utilisations [Diaz 82].

De nombreuses recherches durant ces dernières années ont permis de développer des algorithmes ou des méthodes permettant la validation théorique des Réseaux de Pétri.

Des extensions permettent d'appliquer les RdP à des problèmes complexes ou spécifiques avec facilité (réseaux à prédicats, réseaux à file), permettent d'étudier d'autres phénomènes (réseaux temporisé, réseaux stochastiques), ou permettent d'atteindre la puissance de la Machine de Turing (réseaux à arcs inhibiteurs ou à files).

La définition précise de la sémantique et des propriétés des réseaux de Pétri

permet leur traitement mathématique. Enfin le modèle formel est indépendant de l'implémentation.

Les réseaux de Pétri sont propices à la mise en oeuvre de notre méthodologie, car l'analyse des modèles permet d'obtenir les propriétés qui seront utiles pour prouver la concordance de modèle et l'adéquation de service.

La puissance de modélisation des réseaux de Pétri à prédicats sert pleinement pour la modélisation du service de la couche Réseau et du protocole de la couche Transport.

Nous nous servons d'un ensemble de résultats déjà existants permettant d'obtenir certaines propriétés des modèles. Nous démontrons un résultat supplémentaire : un modèle issu d'un autre modèle, après certaines modifications sur la structure interne des uplets, conserve un comportement équivalent. Ce résultat a été développé pour permettre l'application de notre méthode, en facilitant la conservation des propriétés d'un modèle inclus dans un autre.

Ce résultat nous est utile pour prouver que le modèle du service Réseau, qui a été conçu indépendamment du modèle du protocole Transport, est à même de véhiculer l'ensemble des messages Transport, dont il doit ignorer la structure interne, sans modifier son propre comportement. Ce résultat prouve que le service Réseau transporte de manière transparente les données de la couche supérieure, ce qui est nécessaire par définition de la norme du service, et qui doit être prouvé pour son modèle.

Ce résultat peut être rapproché de la notion de généricité des langages tel que le possède le langage ADA [ADA 83], ou encore de la notion de types abstraits [Berthomieu 81], [Vautherin 85].

Dans un premier paragraphe, nous allons décrire les Réseaux de Pétri sans nous étendre sur toute la théorie.

Le deuxième paragraphe se veut plus une sensibilisation aux RdP à prédicats, qu'une réelle formalisation. Il s'appuie principalement sur la modélisation d'une file réalisée en RdPàP, qui montre clairement les avantages de cette extension. Cet exemple est dû à Berthelot [Berthelot 81].

Dans le troisième paragraphe, nous proposons une définition formelle des Réseaux de Pétri à Prédicats. Nous définissons, ainsi, les notions d'uplets, de champs, de classes d'uplets et de prédicats. Ces définitions servent de supports pour prouver l'équivalence de réseau du paragraphe suivant.

Enfin dans un quatrième paragraphe, nous développons un résultat qui nous sert pour la preuve de notre modèle des protocoles de télé-communication. Ce résultat prouve l'équivalence de comportement, donc la conservation de toutes propriétés, entre un modèle et un autre construit par adjonction d'un champ à une des classes d'uplets.

En fait, toute cette partie ne consiste qu'en un rappel (pour les RdP), ou au mieux, qu'en une définition du contexte formel dans lequel nous travaillons (pour les RdPàP). Seul le quatrième paragraphe est un développement original et nécessaire pour notre travail, d'une importance fondamentale pour le bien fondé de la méthodologie employée dans notre thèse, qui nous permet de concevoir et prouver dans un premier temps le modèle du service Réseau, pour plus tard l'insérer dans le modèle du protocole Transport tout en conservant ses propriétés.

5.2 les Réseaux de Pétri

C'est en 1962 que l'automaticien C.A PETRI fit sa première publication sur les réseaux, qui dès lors prirent son nom. Le réseau est représenté graphiquement par un graphe composé, de barres ou rectangles (appelés transitions), de cercles (appelés places) et d'arcs liant transitions et places. Il a pour caractéristique première sa non-temporalité (obtenue en faisant abstraction des instants des événements et de la durée des actions), ce qui permet de mettre en valeur la logique de fonctionnement des processus modélisés par le réseau.

définition 1: Graphe de Pétri

Un graphe de Pétri est un quadruplet G=(P,T,E,S) ou:

- P est un ensemble fini d'éléments appelé places.

- T est un ensemble fini d'éléments appelé transitions. On note que les deux ensembles P et T sont disjoints.

- E est une application : PxT--->$\mathbb{N}$-{∞}, qui représente la fonction d'entrée des transitions. Soit une place $p \in P$ et une transition $t \in T$, si E(p,t)=n alors il existe un arc étiqueté 'n' de la place p vers la transition t.

- S est une application : PxT--->$\mathbb{N}$-{∞}, qui représente la fonction de sortie des transitions. Soit une place $p \in P$ et une transition $t \in T$, si S(p,t)=n alors il existe un arc étiqueté 'n' de la transition t vers la place p.

Le marquage d'un graphe de Pétri est une application M: P--->$\mathbb{N}$, graphiquement représenté pour une place p, si M(p)=n, par la présence dans cette place p (cercle) de 'n' marques.

définition 2: Réseau de Pétri

Un réseau de Pétri est un couple, RP=(G,Mo), associant un graphe de Pétri G et une fonction de marquage initiale Mo.

définition 3: Franchissabilité

Une transition t de T est franchissable pour un marquage M si et seulement si $\forall p \in P$ M(p)>=E(p,t). Cette relation est appelée condition de franchissement.

définition 4: Franchissement

Après le franchissement de la transition t, on obtient le nouveau marquage M' tel que $\forall p \in P$ M'(p) = M(p)-E(p,t)+S(p,t), aussi noté M(t> = M'.

Remarque : les particularités des Réseaux de Pétri (non-déterminisme, parallélisme et synchronisation) imposent :

- que le franchissement des transitions soit instantané;
- que si le réseau a la possibilité de déclencher plusieurs transitions simultanément, il effectue un choix unique parmi celles-ci;
- que la condition de franchissabilité étant vérifiée, il n'est absolument pas impératif de franchir une transition.

définition 5: Ensemble des marquages accessibles

On note A(RP,Mo), l'ensemble des fonctions de marquage accessibles par le réseau RP à partir du marquage initial Mo, c'est-à-dire l'ensemble des marquages M tel qu'il existe une suite quelconque de transition t1,t2,...,tn tel que Mo(t1>(t2>...(tn> = M.

définition 6: Quasi-vivacité

Une transition t d'un réseau RP est dite quasi-vivante, s'il existe un marquage accessible $M \in A(RP,Mo)$ à partir duquel la transition soit franchissable.

définition 7: Vivacité

Une transition t d'un réseau RP est dite vivante, si et seulement si pour tout marquage accessible M , t est quasi-vivante.

Par extension: Le réseau RP est dit vivant, si et seulement si toutes ses transitions sont vivantes.

définition 8: Etat d'accueil

Un réseau (RP,Mo) admet un état d'accueil Ma si et seulement si pour tout marquage accessible M, on peut atteindre cet état d'accueil , Ma∈ A(RP,M).

propriété 1: Si un réseau possède un état d'accueil Ma et si le réseau (RP,Ma) est quasi-vivant alors le réseau (RP,Mo) est vivant.

De nombreuses autres propriétés et définitions ont été établies, et peuvent être trouvées dans divers ouvrages [Brahms 83], nous n'avons retenu, ici, que celles qui nous servent dans notre discours.

5.3 Un exemple de Réseaux de Pétri à prédicats

Dans le paragraphe précédent, nous avons présenté les Réseaux de Pétri ordinaires (RdP). Cependant de nombreuses extensions sont apparues. Celle qui nous intéresse, le Réseau de Pétri à Prédicats (RdPàP), fut créée pour répondre aux structures répétitives et régulières qui rendent difficiles l'exploitation du dessin alors qu'il peut être conceptuellement simple.

Nous allons illustrer notre exposé sur les RdPàP par l'exemple que présente la file dite 'fifo'. Elle se prête aisément à notre démonstration, par sa structure répétitive qui s'adapte parfaitement à la technique du repliage. De plus, la file fifo est le support de base des mécanismes de communication, que nous allons étudier plus tard (modélisation de câble de transport, de connexion de télécommunication, etc...). La définition formelle de ce réseaux peut être trouvée au paragraphe suivant.

La figure A-5.1a représente une file ordonnée de trois emplacements. La transition "Entrée" représente l'entrée d'un message dans la file, alors que la transition "Sortie" modélise la sortie de la file. Les places pleines "Pi" (pour i=1,2,3) représentent la présence à l'emplacement 'i' d'un message. Les places vides "Vi" (i=1,2,3) représentent que l'emplacement 'i' est vide. La transition T12(resp. T23) fait passer un message de l'emplacement 1 à l'emplacement 2 (resp. 2 à 3).

Le marquage initial du modèle de la file FIFO a :

- les 3 places Vi sont mono-marquées,
- les 3 places Pi sont vides, car la file, au début, ne contient aucun message.

Il est aisé de constater l'accroissement que provoque l'adjonction à cette file des emplacements supplémentaires. Il faut ajouter pour chaque emplacement nouveau, deux places et une transition, ce qui devient prohibitif pour une file un peu longue, et ce, sans provoquer un intérêt quelconque pour le modèle ou la preuve.

Les réseaux de Pétri à prédicats, nous offrent la possibilité de construire toutes structures régulières simplement. La base de cette abréviation est le paramétrage de chaque structure, qui après repliage permet de les distinguer.

C'est ainsi qu'on confond les places Vi en une seule place V, et on attribue aux marques se trouvant dans les places V1,V2,V3 un paramètre i spécifiant le rang de la place. On procède de même avec les places P1,P2,P3 et P, en précisant pour chaque arc de chaque transition la valeur de la marque la déclenchant. On obtient alors le graphe de la figure A-5.1b.

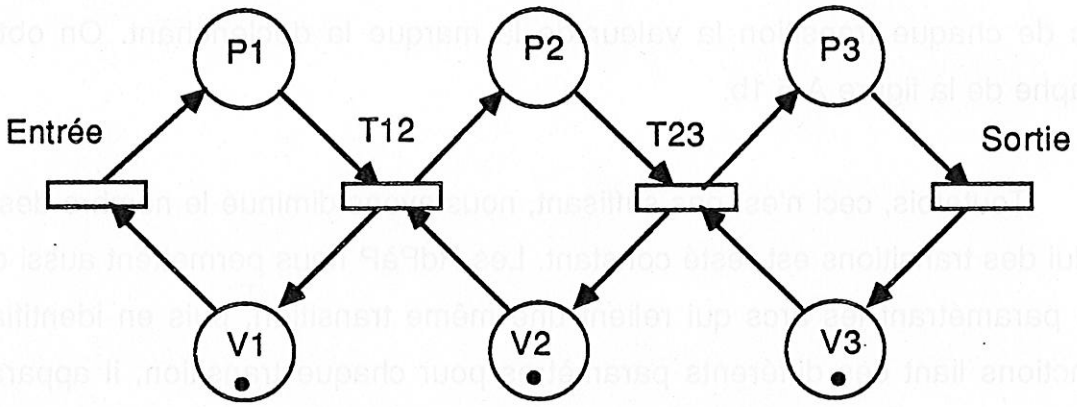
Toutefois, ceci n'est pas suffisant, nous avons diminué le nombre des places mais celui des transitions est resté constant. Les RdPàP nous permettent aussi de le réduire. En paramétrant les arcs qui relient une même transition, puis en identifiant la ou les fonctions liant ces différents paramètres pour chaque transition, il apparaît alors une similitude, nous permettant de replier les arcs et les transitions correspondantes.

Ainsi la transition T utilise une marque de rang i de la place P et une marque de rang j de la place V, pour produire une marque de rang j dans P et i dans V, et ce, si la condition j==i+1 est vérifiée (cette condition est appelée "prédicat"). Nous associons, donc, à chaque transition un prédicat qui écrit sur le graphe à côté du trait représentant cette transition. Nous obtenons la figure A-5.1c. On remarque que l'on peut encore replier, notamment en définissant une classe d'uplet modélisant un emplacement. Cet uplet ayant deux champs, un champ modélisant le numéro de l'emplacement, l'autre champ modélisant l'état de l'emplacement (vide, plein). On obtient la figure A-5.1d.

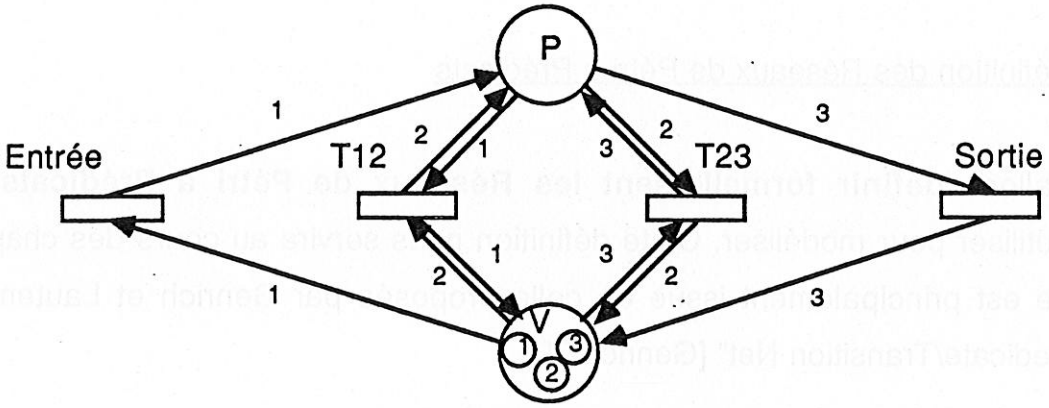
Nous tenons à rappeler que les RdPàP ne sont qu'une manière concise de décrire les RdP, pour peu que tous les prédicats et les uplets portent sur des variables à valeurs dans des ensembles finis. En fait, les modèles sont directement conçus en Réseaux de Pétri à prédicats. On construit des uplets paramètrés, ayant une sémantique particulière (par exemple: un message comportant les champs suivants, type de message, numéro

de message, donnée du message, etc...). Chaque transition manipule ces uplets, aux moyens des opérateurs et prédicats associés.

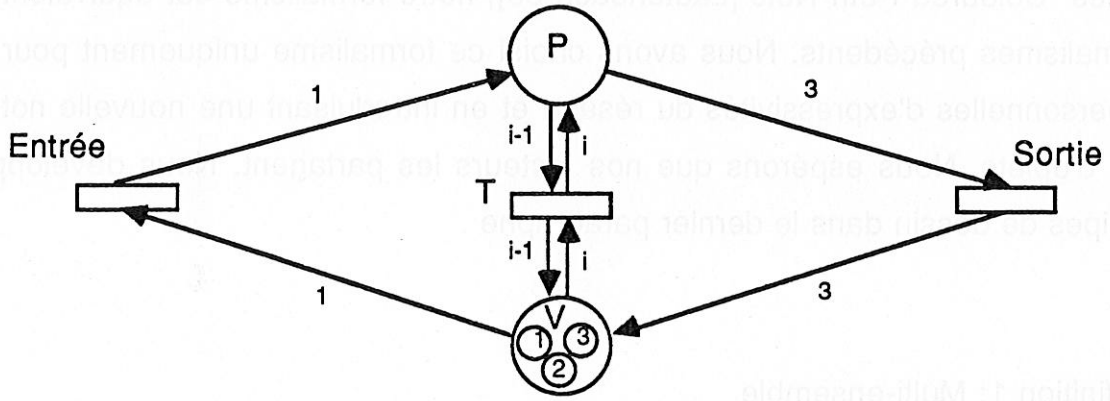
L'ensemble des définitions ou propriétés définies au paragraphe précédent 'les Réseaux de Pétri', doivent être étendues aux réseaux de Pétri à prédicats.



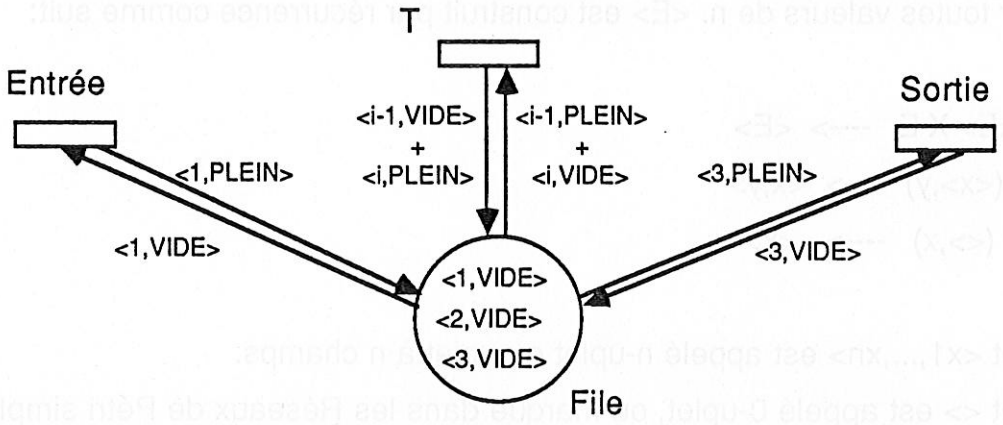
- Figure A-5.1a - File RdP simple



- Figure A-5.1b - File RdPàP



- Figure A-5.1c - File RdPàP



- Figure A-5.1d - File RdPàP replié

5.4 - Définition des Réseaux de Pétri à Prédicats

Nous allons **définir formellement les Réseaux de Pétri à Prédicats** que nous allons utiliser pour modéliser. Cette définition nous servira au cours des chapitres suivants, elle est principalement issue de celle proposée par Genrich et Lautenbach pour leur "Predicate/Transition Net" [Genrich 79].

Notre définition des réseaux de Pétri à prédicats peut ne pas sembler très orthodoxe, cependant comme Lautenbach l'a montré entre les "Predicates/transitions Nets" et les "Coloured Petri Nets"[Lautenbach 85], notre formalisme est équivalent aux deux formalismes précédents. Nous avons choisi ce formalisme uniquement pour des raisons personnelles d'expressivités du réseau et en introduisant une nouvelle notion : la classe d'uplets. Nous espérons que nos lecteurs les partagent. Nous développons nos principes de dessin dans le dernier paragraphe .

Définition 1: Multi-ensemble

Nous appelons le multi-ensemble E (noté <E>), l'ensemble construit à partir de l'ensemble E, avec l'opérateur <>. C'est l'ensemble de tous les n-uplets à composantes dans E pour toutes valeurs de n. <E> est construit par récurrence comme suit:

<> : <E> X E ----> <E>
(<x>,y) ----> <x,y>
(<>,x) ----> <x>

L'uplet <x1,...,xn> est appelé n-uplet ou uplet à n champs.

L'uplet <> est appelé 0-uplet, ou marque dans les Réseaux de Pétri simple.

L'uplet <x> est appelé 1-uplet ou uplet simple, ou couleur dans les Réseaux colorés.

On note :

A*,la généralisation de la notation A^n , $n \in \mathbb{N}$.

[A --> B] , l'ensemble des fonctions de A dans B .

Définition 2 : Réseaux de Pétri à prédicats

Nous définissons le graphe des Réseaux de Pétri à prédicats que nous allons utiliser comme un sextuplet.

R = < P , T , U , Pré , Post , Pred > , où

- P est un ensemble fini d'objets, appelés places.

- T est un ensemble fini d'objets, appelés transitions.

- U est une partie d'un multi-ensemble construit à partir d'un ensemble fini E. Cet ensemble U caractérise:

Un ensemble F d'uplets de fonctions de U vers $\mathbb{N}$:

$F \subset [U \text{ --> } \mathbb{N}]$;

Un ensemble OP d'opérateurs de fonctions de F* vers F :

$OP \subset [F^* \text{ --> } F]$;

Un ensemble PR de prédicats de fonctions de F* vers {vrai, faux} :

$PR \subset [F^* \text{ --> } \{vrai,faux\}]$.

- Pré est la fonction d'incidence avant de PxT vers F, qui correspond aux libellés des arcs sortant des places.

- Post est la fonction d'incidence arrière de PxT vers OP, qui correspond aux libellés des arcs entrant dans les places.

- Pred est la fonction de condition de T vers PR , qui correspond aux libellés des transitions.

Définition 3 : Fonction de marquage

Nous définissons l'ensemble M des fonctions de marquage d'un réseau comme un ensemble de fonctions de l'ensemble des places P vers l'ensemble F des uplets .

$M \subset [P \rightarrow F]$;

Nous définissons un modèle de Réseaux de Pétri à prédicats comme un graphe muni d'une fonction de marquage initiale M° de P vers F.

On note :

Pré(.,t) : l'ensemble des uplets élément de F*, issu des places associées à la transition t élément de T.

Définition 4 : Condition de franchissabilité

Pour pouvoir franchir la transition t élément de T, à partir de la fonction de marquage M, il faut que quelque soit la place p élément de P, quelque soit f élément de F , on ait :

$Pré(p,t)(f) \leq M(p)(f)$, et que

Pred(t) (Pré(.,t)) soit vrai .

Définition 5 : Action de franchissement

Le franchissement de la transition t élément de T, provoque la modification de la fonction de marquage M vers M'. Quelque soit p élément de P , quelque soit f élément de F :

$$M'(p)(f) = M(p)(f) - Pré(p,t)(f) + Post(t,p)(Pré(.,t))(f)$$

Définition 6 : Classe d'uplets

Nous appelons **Cp** la fonction qui associe à chaque place de l'ensemble P, l'ensemble des uplets de U susceptibles d'y résider. Cet ensemble peut être déterminé par le domaine d'arrivée des fonctions valuant les arcs d'entrées de chaque place. Cette notion de classe d'uplet trouve son explication dans la signification associée à la place. C'est souvent un ensemble d'objets ayant une structure identique, et qui ne diffèrent que par leur nom et leur état. Elle peut aussi se rattacher à la notion de couleur de place des réseaux colorés.

Nous appelons **Ct** la fonction qui associe à chaque transition de l'ensemble T, l'ensemble des uplets de U susceptibles de la traverser. Cet ensemble peut être déterminé par le domaine d'arrivée des fonctions valuant les arcs de sorties de chaque transition. Cette notion de classe d'uplet trouve son explication dans la signification associée au prédicat de chaque transition. C'est souvent un ensemble d'objets ayant une structure identique, et qui ne diffèrent que par leur nom et leur état. Elle peut aussi se rattacher à la notion de couleur de transition des réseaux colorés.

Chaque classe Cp et Ct forment un ensemble de partie de l'ensemble des uplets U. Nous pouvons donc restreindre les ensembles de fonctions F, OP, PR, M et les fonctions Pré, Post, et Pred à ces classes.

Notation matricielle :

Comme pour les réseaux colorés, tout réseau exprimé à l'aide de notre formalisme, peut s'écrire sous une forme matricielle. Cependant pour pouvoir calculer de manière automatique les invariants linéaires du modèle, il faut restreindre l'ensemble des fonctions utilisables.

Chaque uplet s'écrit comme un vecteur sur l'ensemble U des uplets possibles. Les fonctions Pré et Post s'expriment, alors, sous forme d'une matrice (CpxCt).

L'ensemble des définitions ou propriétés définies au paragraphe précédent "les Réseaux de Pétri", doivent être étendues aux Réseaux de Pétri à prédicats.

Définition 7: Quasi-vivacité d'un marquage

Un réseau R est quasi-vivant à partir du marquage M si

$\forall t \in T$, $\exists M' \in A(M,R)$ tel que $M'(t>$.

On voit, ici, qu'il suffit que n'importe quel uplet (de n'importe quel classe) franchisse les transitions pour que le réseau soit quasi-vivant. Le réseau de Pétri ordinaire obtenu par dépliage systématique pourrait de ce faite ne pas être quasi-vivant (par exemple : un type d'uplet ne franchissant jamais telle transition). Toutefois, cette particularité n'est pas problématique, car la sémantique associé au franchissement des transitions des Réseaux à prédicats est adéquat à notre définition de la quasi-vivacité.

Définition 8 : Vivacité d'un réseau

Un réseau R est vivant si

$\forall t \in T$, $\forall M \in A(Mo,R)$, $\exists M' \in A(M,R)$ tel que $M'(t>$.

Définition 9 : Ensemble d'état d'accueil

L'ensemble d'état d'accueil est une extension de l'état d'accueil des réseaux de Pétri ordinaires adapté aux réseaux de Pétri à prédicats, c'est une collection d'états d'accueils. Un réseau (R,Mo) admet un ensemble d'accueil Ea si et seulement si pour tout marquage accessible $M \in A(R,Mo)$, on peut atteindre un des marquages appartenant à l'ensemble d'accueil Ea.

$\forall M \in A(R,Mo)$, $\exists M' \in A(R,M)$ tel que $M' \in Ea$.

Propriété 2 :

Si un réseau (R,Mo) possède un état d'accueil Ea et si le réseau est quasi-vivant pour tout marquage de l'ensemble d'accueil alors le réseau est vivant.

5.5 Equivalence de modèles

5.5.1 Introduction

Bien que les Réseaux de Pétri à Prédicats soient des outils très puissants, la modélisation de cas réels, tels que les protocoles de télécommunications, peut engendrer de très grands modèles.

La méthodologie que nous avons développée permet d'avoir une structure générale pour la conception et la preuve de processus concourants. Elle est basée sur trois principes :

- Diviser le système en couches indépendantes, ce qui permet de simplifier et clarifier les problèmes à résoudre. Définir une vue externe (le service) et une vue interne (la réalisation) de chaque couche ce qui augmente encore son indépendance (notamment par rapport aux choix d'implémentation).

- Utiliser un langage formel pour spécifier sans ambiguïté les systèmes. Cette description est la base de référence de la méthode.

- Valider les choix effectués pendant les phases de conception ou d'implémentation en associant au langage de spécification un modèle, dont on puisse extraire les propriétés qui nous intéressent.

En suivant cette méthode, il suffit pour valider le système de valider chaque couche indépendamment. Valider une couche, nécessite la construction du modèle du service de la couche N-1, pour l'insérer dans le modèle de la réalisation de la couche N. Si comme nous l'étudions, on s'intéresse aux protocoles de télécommunication, et si l'on ne veut pas faire trop d'approximation, ces modèles peuvent encore s'avérer d'une grande complexité.

Nous sommes donc contraints de trouver, à nouveau, les moyens de réduire cette complexité pour étudier plus confortablement les comportements du système.

La technique de division a fait ses preuves, cependant ici on suppose que l'on a atteint la plus petite partie licitement indépendante des autres. Nous allons donc devoir diviser le modèle en sous-modèles non indépendants conceptuellement des autres. Une question est soulevée alors, peut-on conserver les propriétés de sous-modèles qui ne sont pas indépendants?

Nous avons à notre disposition plusieurs techniques:

- Les réductions de réseau de Pétri permettent de passer d'un réseau à un réseau équivalent, plus petit dont on connaît déjà les propriétés [Berthelot 83]. Cette technique fortement développée par Berthelot a déjà été appliquée avec succès sur des protocoles de télécommunications.

- Les abstractions d'André [André 81], qui dit : si deux réseaux ont même abstraction sur un ensemble de transitions distinguées, alors il est possible de remplacer l'un par l'autre à l'intérieur d'un troisième sans modifier le fonctionnement de ce dernier.

- La méthode des blocs bien formés est une technique qui permet de construire des sous-modèles, puis de les réunir, tout en s'assurant la conservation de bonnes propriétés [Valet 83], [Kotov 78].

- Certaines équipes ont développé une technique permettant de comparer le graphe des marquages accessibles de deux modèles de réseaux de Pétri [Dufau 84][Bourguet 86]. En rapprochant, des transitions deux à deux, on peut vérifier si les deux modèles ont un comportement similaire.

Malheureusement toutes ces techniques ne s'appliquent qu'à des Réseaux de Pétri simples, et ne conservent que partiellement les propriétés des sous-réseaux

(souvent la vivacité). Nous nous servons de Réseaux de Pétri à prédicats, nous avons donc développé une technique permettant d'insérer un sous-modèle dans un modèle en conservant l'ensemble de ses propriétés. Cette technique, bien que d'application restreinte, est parfaitement adaptée à l'utilisation de notre méthode pour la validation des protocoles de télé-communication. Elle est l'équivalent, pour les RdPàP, des places et des transitions non-contraignantes de la technique de réduction pour les RdP (notion de places implicites, etc...), ou encore de la technique d'abstraction d'André. Cependant contrairement aux réductions nous sommes assurés avoir le même comportement, donc la conservation de toutes les propriétés exprimables à la fois dans les deux modèles; et par rapport au principe d'abstraction qui essaye de prouver à posteriori l'équivalence de comportement de deux réseaux, notre technique propose un moyen d'obtenir à priori des réseaux équivalents.

5.5.2 Equivalence de deux réseaux - Preuve

Nous allons construire à partir d'un réseau R un nouveau réseau R' dont nous allons prouver qu'il conserve les propriétés. Le nouveau réseau R' est issu du réseau R en ajoutant un champ à une classe d'uplet. L'ensemble du reste du réseau reste inchangé.

Condition de construction pour avoir équivalence

A partir d'un réseau R : <P,T,U,Pré,Post,Pred>, on construit un nouveau réseau R' : <P, T, U', Pré', Post', Pred'> conservant la structure du graphe bi-parti. Nous modifions uniquement U en U' ,en substituant un uplet u par un autre u', construit à partir de u, mais en lui rajoutant un champ e à valeur dans E.

Le réseau R' respecte les règles de construction suivante, il existe une projection ¶ tel que :

¶ : U' ⊂ <U,E> ---> U

u' = <u,e> ---> u

Qui a tout élément de U' associe un et un seul élément de U.

∀ u' ∈ U' , ∃ u ∈ U tel que u = ¶(u').

De même nous pouvons étendre cette projection sur l'ensemble des fonctions d'uplets de F' vers l'ensemble des fonctions d'uplets F, qui a toute fonction f' de F' associe la fonction f de F tel que : ∀ f' ∈ F' , ∃ f ∈ F tel que

∀ u' ∈ U' , f'(u') = f(¶(u')). On note f=¶f(f').

Puis nous prolongeons cette projection sur les ensembles de fonctions Pred, Pré et Post vers Pred', Pré' et Post' qui doivent être construits comme suit :

Pré' : PxT ---> F'

p,t ---> f'

Qui à tout couple place/transition associe la fonction f' construit tel que :

$\forall f' \in Pré'(p,t)$, $\exists f \in Pré(p,t)$ tel que $f = ¶f(f')$.

Post' : PxT ---> OP'

p,t ---> op'

Qui à tout couple place/transition associe l'opération op' construit à partir des fonctions f_i' tel que :

$\forall op' \in Post'(p,t)$, $\exists op \in Post(p,t)$ tel que $op = ¶p(op')$,

avec ¶p définit comme suit :

$¶p : OP' \subset [F'^* \rightarrow F'] \quad \rightarrow \quad OP \subset [F^* \rightarrow F]$

$op' = (f_1'x...xf_n' \rightarrow f') \rightarrow op = (f_1 x...xf_n \rightarrow f)$ avec $\forall i \; f_i = ¶f(f_i')$.

Pred' : T ---> PR' $\subset$ [F'* ---> {VRAI,FAUX}]

t ---> pr' = $(f_1'x...xf_n' \rightarrow vf)$

Qui à toute transition associe le prédicat pr' construit à partir des fonctions f_i' tel que :

$\forall f_i \in Pred'(t)$, $\exists f_i \in Pred(p,t)$ tel que $f_i' = ¶f(f_i)$.

Enfin, le marquage initial Mo' du réseau R' doit avoir pour image par la projection ¶ le marquage initial Mo du réseau R :

$\forall p \in P$,$\forall f' \in Mo'(p)$, $\exists f \in Mo(p)$ tel que $¶f(f')=f$.

<u>Preuve d'équivalence de comportement si les deux réseaux respectent les conditions de construction.</u>

Nous voulons prouver, maintenant, que les deux réseaux R et R' ont même comportement, c'est à dire qu'à tout marquage accessible du réseau R', nous pouvons trouver un marquage accessible sur R équivalent.

Nous définissons l'équivalence de deux marquages en disant que l'un est déduit de l'autre par la projection ¶. Cette extension de la projection est licite, car l'ensemble des fonctions de marquage est défini de l'ensemble des places P vers F. $\forall p \in P, \forall f' \in M'(p), \exists f \in M(p)$ tel que $¶f(f')=f$. On le note $¶m(M')=M$.

Donc les deux réseaux sont équivalent si :

$\forall M' \in A(R',Mo')$, $\exists M \in A(R,Mo)$ tel que $M=¶m(M')$.

Nous prouvons cette équivalence de comportement, par récurrence sur la longueur des suites de franchissements des transitions.

Les deux réseaux partent d'un état équivalent, par hypothèse de construction de leur marquage initial respectif : $¶m(Mo')=Mo$.

Maintenant, il faut vérifier qu'à partir de n'importe quel marquage accessible $M' \in A(R',Mo')$, connaissant le marquage associe $M = ¶m(M') \in A(R,Mo)$, si l'on franchit une transition $t \in T$ sur le réseau R' en obtenant le marquage $M_1'=M'(t>$, alors on peut aussi la franchir sur le réseau R en obtenant un marquage $M_1=M(t>$, et l'on obtient alors un marquage équivalent $¶m(M_1')=M_1$.

Nous allons démontrer dans un premier temps que les conditions de franchissements sont réalisées, puis que les actions de franchissements produisent un marquage résultant équivalent.

Evaluation de la première condition de franchissement :

Par hypothèse de construction de la fonction Pré' , nous savons que :

$\forall p \in P , \forall f' \in F'$, Pré'(p,t) (f') = Pré(p,t) (¶f(f')).

Par hypothèse de récurrence, la condition de franchissement d'une transition t dans le réseau R muni d'une fonction de marquage M implique :

$\forall p \in P , \forall f \in F$, Pré(p,t) ≤ M(p) (f).

Et par hypothèse de construction de l'ensemble F' :

$\forall f' \in F', \exists f \in F$ tel que f = ¶f(f').

Donc $\forall p \in P , \forall f' \in F'$, Pré'(p,t) (f') ≤ M(p) (¶(f')).

Par hypothèse de récurrence sur l'état des deux réseaux , nous savons que :

$\forall p \in P , \forall f' \in F'$, M'(p) (f') = M(p) (¶(f')).

Donc nous obtenons la vérification de la première condition de franchissement sur le réseau R' :

$\forall p \in P , \forall f' \in F$, Pré'(p,t) (f') ≤ M'(p) (f').

Evaluation de la deuxième condition de franchissement :

D'après l'hypothèse de construction de l'ensemble de fonctions Pred', nous savons que :

$\forall f' \in F'$, Pred'(t) (f') = Pred(t) (¶(f')).

Par hypothèse de construction le l'ensemble Pré' , nous savons que :

$\forall p \in P , \forall f' \in F'$, Pré'(p,t) (f') = Pré(p,t) (¶(f')).

Et par conservation du graphe bi-parti, nous déduisons que:

$\forall f' \in F'$, Pré'(.,t) (f') = Pré(.,t) (¶(f')).

Par hypothèse sur la deuxième condition de franchissement, nous avons :
Pred(t) (Pré(.,t)) vrai.

Nous pouvons donc affirmer que la deuxième condition de franchissement de la transition t est réalisée sur le réseau R', c'est à dire:

Pred'(t) (Pré'(.,t)) est vrai.

Preuve que nous obtenons un marquage équivalent.

Par hypothèse de récurrence sur l'action de franchissement,

si M(t>M1 : $\forall p \in P, \forall f \in F$, nous avons

M1(p) (f) = M(p) (f) - Pré(p,t) (f) + Post(t,p) (Pré(.,t)) (f).

Par hypothèse sur la construction de l'ensemble F' :

$\forall f' \in F', \exists f \in F$ tel que f' = ¶f(f')).

Donc $\forall p \in P, \forall f' \in F'$, M1(p) (¶f(f')) = M(p) (¶f(f')) - Pré(p,t) (¶f(f')) + Post(t,p) (Pré(.,t)) (¶f(f')).

Par hypothèse de construction de Pré':

$\forall f' \in F'$, Pré'(p,t)(f') = Pré(p,t)(¶f(f')).

Par hypothèse de construction de Post' :

$\forall f_i' \in$ Post'(p,t) , $\exists f_i \in$ Post(p,t) tel que f_i = ¶f(f_i').

Par hypothèse de récurrence sur le marquage :

$\forall p \in P, \forall f' \in F'$, M'(p(f') = M(p)(¶f(f')).

Donc $\forall p \in P, \forall f' \in F'$, M1(p) (¶f(f')) = M'(p) (f')- Pré'(p,t) (f')+ Post'(t,p) (Pré'(.,t)) (f').

Par définition de l'action de franchissement :

$\forall p \in P, \forall f' \in F'$, M1'(p) (f')= M'(p) (f')- Pré'(p,t) (f')+ Post'(t,p) (Pré'(.,t)) (f').

Donc les deux marquages résultants du franchissement d'une transition t quelconque sont bien équivalents :

$\forall p \in P, \forall f' \in F', M1(p)(\P f(f')) = M1'(p)(f')$.

Extension de l'équivalence à une classe d'uplets

Nous venons de prouver qu'un réseau R' construit à partir d'un réseau R, en substituant un uplet par un autre construit à partir du premier en lui rajoutant un champ, ont un comportement équivalent. Nous pouvons de même construire un nouveau réseau R" à partir de R' en modifiant un second uplet, ces réseaux ayant toujours un comportement équivalent. Si nous généralisons ce processus à l'ensemble des uplets formant une classe, nous obtenons un nouveau réseau R° ayant un comportement équivalent au premier réseau R.

Exemple

Un exemple de construction d'un réseau équivalent peut être illustré par le modèle d'une file. Le transfert de donnée à travers la file devant être transparent aux types de données, il va de soi que le comportement de la file ne dépend pas de la structure interne des messages en transit. C'est ce qu'exprime notre équivalence de réseau.

5.6 Règles de modélisation pour les RdPàP

Nous avons établi plusieurs règles de dessin pour les Réseaux de Pétri à Prédicats, afin de systématiser la conception, et de faciliter la lecture. Ces règles ne limitent en rien la puissance de modélisation des RdPàP, mais canalisent la divergence pouvant être engendrée par les différentes possibilités de modélisation.

Les places n'ont le droit de contenir que des uplets d'une seule classe. De ce fait, les arcs entrant (sortant) enlèvent (mettent) des uplets appartenant à la même classe que celle de la place.

Les arcs sortants ne portent que des noms d'uplets.

Les arcs entrants peuvent supporter des fonctions d'uplets.

Les prédicats ne peuvent en rien modifier les objets du réseau. Ils servent uniquement à autoriser ou refuser un franchissement, à faire une sélection parmi les uplets des places d'entrées.

Nous soutenons que, tout modèle écrit en RdPàP sans ces règles de dessin, peut se réécrire en suivant ces règles de dessin, en conservant tous ses comportements.

Pour exemple, nous donnons la figure A-5.2, qui est toujours un modèle d'une file fifo, mais cette fois de longueur N.

L'ensemble des transitions: T = { Entrée , Sortie , Transit }
L'ensemble des places : P = { File }
L'ensemble des marques :

U = { u / u=<noempl,état>, noempl∈[1,N] et état∈{VIDE,PLEIN} }

L'ensemble des opérateurs : OP = { remplir , vider, transfert }
L'ensemble des prédicats :

Pred = { u.noempl==1 , u.noempl==N , u.état==PLEIN , u.état==VIDE }

PLACES, TRANSITIONS, MARQUES et PREDICATS du MODELE

P={file} et T={Sortie, Entrée, Transit}

U = { u / u=<n,e> , n∈ [1,N] et e ∈{VIDE,PLEIN} }

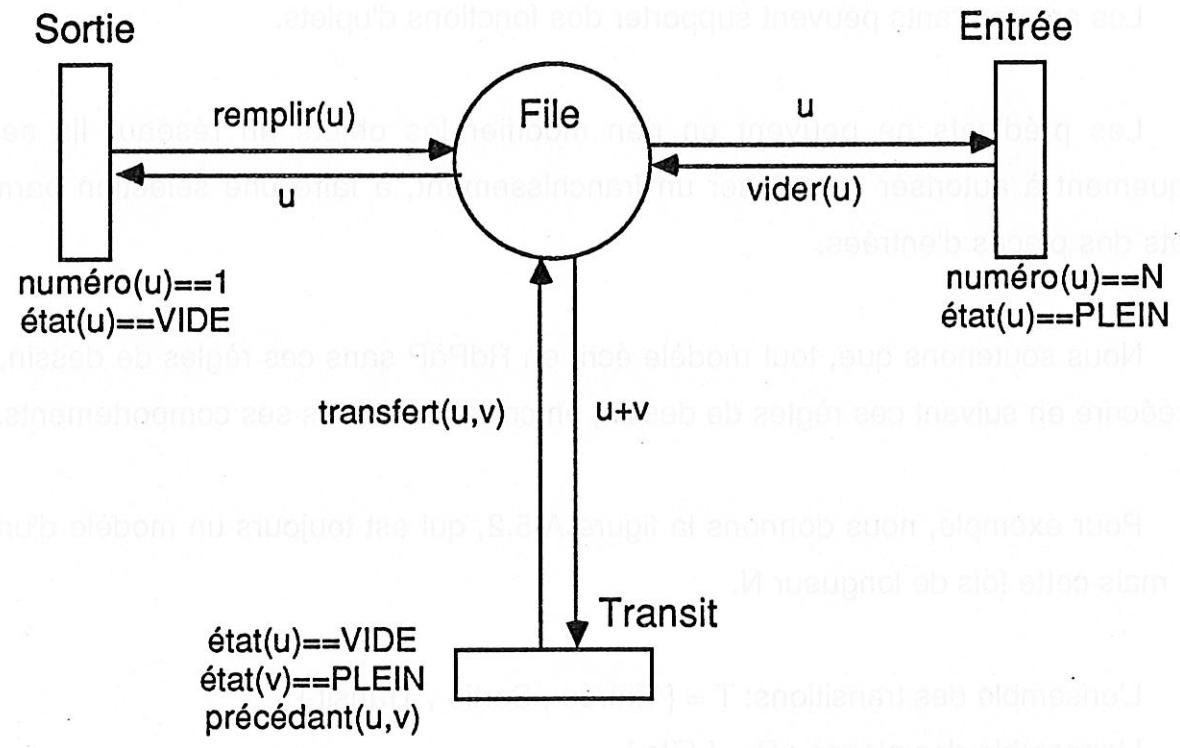
remplir(u):v={état(v)=PLEIN,numéro(v)=numéro(u)}

vider(u):v= {état(v)=VIDE,numéro(v)=numéro(u)}

precedant(u1,u2)={ (numéro(u1)==numéro(u2)+1) VRAI sinon FAUX}

transfert(u1,u2):v1,v2={ numéro(v1)=numéro(u1), numéro(v2)=numéro(u2)
état(v1)= état(u2) , état(v2)=état(u2) }

état(<n,e>)={e} , numéro(<n,e>)={n}



- Figure A-5.2 - File RdPàP

5.7 Conclusion

Nous venons de décrire formellement les Réseaux de Pétri à prédicats, en introduisant un ensemble de notations et de propriétés qui nous faciliteront l'établissement des preuves pendant l'étape de validation du modèle, notre description étant directement issue de celle des "Predicates/Transitions Nets".

Nous avons défini une nouvelle notion : la classe d'uplet, qui représente un regroupement d'uplets ayant une même interprétation. Il semble naturel de s'attendre au même comportement pour des uplets appartenant à la même classe.

Notre description des Réseaux de Pétri à prédicats nous a permis de prouver que, les modifications de structure (l'ajout d'un champ) d'une classe d'uplets ne modifient pas le comportement du modèle initial Ce résultat sera utilisé pour la conception et la preuve du modèle du protocole Transport, qui doit inclure un sous-modèle de service Réseau.

Nous avons, enfin, défini un ensemble de règles de modélisation nous permettant de systématiser la conception des modèles et de faciliter la lecture des modèles que nous allons devoir construire dans les parties suivantes de notre thèse.

6. CONCLUSION

Dans cette première partie de notre thèse, nous avons étudié successivement quatre sujets:

La mise en évidence des problèmes de fiabilité des systèmes informatiques et des divers outils et méthodes développés actuellement pour les résoudre. Ce qui nous a permis de situer les Réseaux de Pétri, comme un outil de modélisation et de validation, permettant d'intervenir pendant les phases primordiales de conception des systèmes.

Nous avons situé le protocole de télécommunication de la couche 4, dit Transport, de la norme O.S.I, parmi l'ensemble des moyens téléinformatiques actuels. Nous nous sommes aperçus que l'étude de la phase de transfert de données de la classe 3 était particulièrement intéressante par la complexité de ses fonctionnalités.

Puis, nous développons après une étude générale, une méthodologie qui intègre les concepts de couches et de service dans les phases de modélisation et de validation des systèmes complexes. Nous définissons, ainsi, deux techniques pour établir la validation d'un modèle d'un système: la concordance de modèle et l'adéquation de service . La première technique établissant une équivalence entre le système et son modèle, la deuxième établissant une équivalence entre le service et la réalisation d'un même modèle.

Enfin, nous avons défini formellement l'outil de modélisation et de validation que nous employons : les Réseaux de Pétri à prédicats. Nous avons porté notre attention sur les modifications de réseaux, qui conservent les propriétés initiales. Nous avons prouvé qu'un principe de modification simple des champs des uplets appartenant à une même classe conserve un comportement équivalent. Cette technique de construction nous sera utile, par la suite, pour assurer le transport transparent des messages de Transport sur le modèle du service Réseau.

B - DEUXIEME PARTIE

Le SERVICE RESEAU

Son Modèle

Sa Validation

PLAN
de la Deuxième Partie

1.INTRODUCTION

Pour modéliser la couche Transport, il nous faut préalablement définir et modéliser les services rendus par la couche inférieure. Pour ce faire, nous allons décrire, modéliser **le service de la couche Réseau** et prouver ses propriétés, afin d'établir **la concordance du modèle** en accord avec la méthode proposée en première partie.

Nous ne nous intéressons, ici, qu'au service de la couche Réseau et non au protocole, ce qui nous permet d'ignorer des comportements complexes (routage, numérotation, retransmission, etc...). Toutefois, nous devons tenir compte de nombreux phénomènes, dont la modélisation est loin d'être évidente: la perte d'informations, la conservation de la séquentialité, la non duplication, et surtout le traitement de la phase de resynchronisation.

Cette **phase de resynchronisation** est particulièrement critique. Elle est lancée quand le réseau détecte des risques d'incohérences provoqués par une désynchronisation interne d'un site assurant la transmission des données. Elle a pour but de nettoyer la connexion Réseau de tous les paquets résiduels, et d'émettre vers les deux entités communiquantes un signal de réinitialisation. Ce signal assure et prévient les deux entités de la possibilité d'incohérence sur les informations envoyées ou reçues antérieurement à la réception du signal, et le retour à la normale pour les informations envoyées et délivrées après la réception du signal.

Il faut bien constater que cette resynchronisation est effectuée de manière décentralisée, et que notamment la réception des signaux de réinitialisation peut se faire à une extrémité, bien avant sa réception à l'autre extrémité. Autrement dit, une extrémité s'est aperçu de la désynchronisation intervenue dans le réseau, alors que l'autre l'ignore encore.

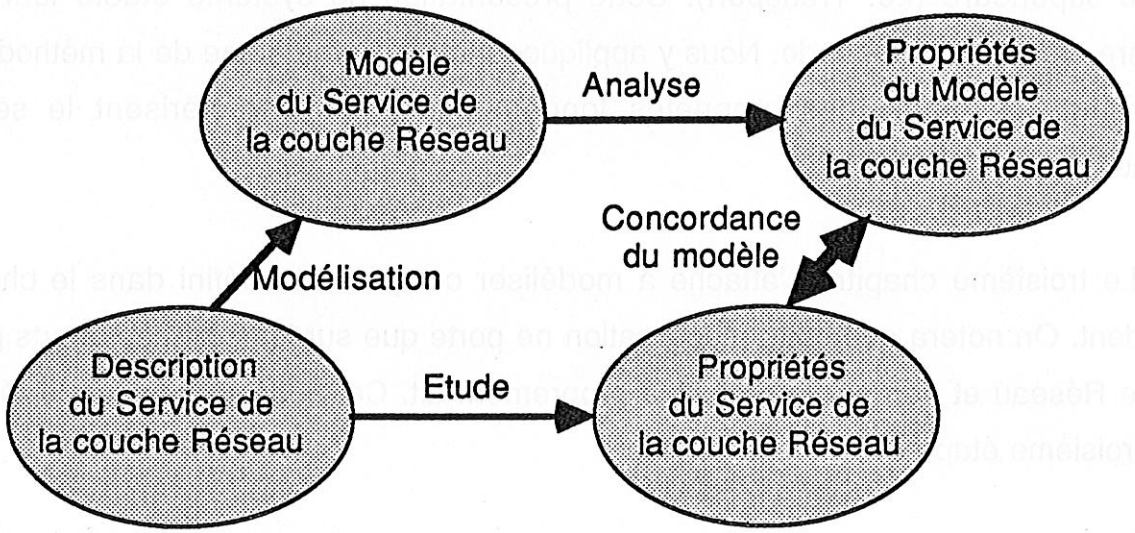
Cette deuxième partie de notre thèse, après cette brève introduction qui constitue le premier chapitre, nous permettra d'illustrer notre méthodologie en l'appliquant au service Réseau (figure B-1):

Le deuxième chapitre présente les services de la couche Réseau, services qu'elle doit assurer (d'après la Norme d'Interconnexion des Systèmes Ouverts) auprès de la couche supérieure (i.e. Transport). Cette présentation du système étudié fournit la première étape de la méthode. Nous y appliquons la deuxième étape de la méthode, en établissant l'ensemble des propriétés fondamentales qui caractérisent le service Réseau.

Le troisième chapitre s'attache a modéliser ce qui a été défini dans le chapitre précédent. On notera que cette modélisation ne porte que sur les services offerts par la couche Réseau et non sur le protocole proprement dit. Ce modèle constitue le résultat de la troisième étape de la méthode.

Le quatrième chapitre démontre formellement que le modèle construit respecte bien les services demandés au deuxième paragraphe (c'est-à-dire correspond à la Norme), c'est ce que nous avons appelé la concordance de modèle dans notre méthodologie. Nous démontrons principalement que le modèle et la description du service Réseau possèdent en commun les quatre propriétés suivantes:

- le système n'arrive jamais dans une situation de blocage,
- la conservation de la séquentialité des paquets,
- leur non duplication en cours de transfert,
- la détection de toute perte ou corruption de paquets.



- Figure B-1. - Méthodologie appliquée à la couche Réseau

2. LE SERVICE RESEAU

2.1 Introduction

Ce premier paragraphe permet d'appréhender le fonctionnement du protocole et du service de la couche Réseau, puis d'extraire du service un ensemble de propriétés qui permettront dans le dernier paragraphe d'exprimer la concordance du modèle du service Réseau avec sa description.

Notre compréhension de la couche de niveau 3 ou Réseau est basée sur les documents des normes internationales [ISO 8208] [ISO 8348], ou françaises [STUR 80]. Dans ces documents nous trouvons d'un part la description du service, d'autre part la spécification du protocole. Nous nous intéressons ici au fonctionnement "en mode connexion" de la couche Réseau.

La couche de niveau 3, appelée Réseau, autorise l'acheminement des données structurées en paquets au travers d'un réseau composé de nombreux sites intermédiaires, afin de permettre à deux entités distantes de communiquer. Le réseau, support de la communication, est essentiellement public en France du fait du monopole d'état. Ce peut être :

A- Un réseau à liaison spécialisée, qui établit une liaison permanente entre les entités communiquantes. Ce sont :

- les liaisons spécialisées téléphoniques en transmission analogique sur 2 ou 4 fils,
- les canaux numériques à débit spécifique (Transmic) qui sont réservés à la transmission de données synchrones.

Un réseau utilisant comme support un réseau non conçu pour la transmission de donnée, comme :

- le réseau téléphonique,
- le réseau télex.

B- Un réseau commuté, conçu spécifiquement pour la transmission et la commutation de données, qui peut être aujourd'hui :

- le réseau à commutation de circuit analogique Caducée,
- le réseau à commutation de paquets Transpac,
- le réseau de commutation de circuits numériques, constitué de la partie numérique du réseau téléphonique RTC 64 ou du réseau satellite Télécom1.

C- Enfin, ce pourrait être le futur réseau numérique à intégration de services RNIS.

Cependant à côté des technologies et des services de réseau offerts par l'administration pour le transfert de données à grande distance, il faut mentionner les technologies permettant de constituer des réseaux privés :

- les réseaux locaux (par exemple Ethernet),
- les auto-commutateurs multiservices (ou PABX).

Le réseau Transpac est notre support de référence , car ce réseau est très largement diffusé en France, et il offre à l'heure actuelle un service de transmission de données directement lié au service Réseau défini par les normes internationales OSI en mode connexion. Ce réseau est à commutation de paquets. Un paquet étant l'unité de transmission des informations sur la connexion, c'est le nom des N-PDU de la couche Réseau.

Les caractéristiques générales du service qu'assure la couche Réseau auprès de la couche Transport en s'appuyant sur la couche Liaison sont :

- transfert transparent des paquets,
- routage des paquets en fonction de leur adresse réseau,
- maintien de la qualité du service de transfert des données.

Ces caractéristiques amènent le service Réseau à fournir:

- le moyen d'établir une connexion entre deux entités distantes, la connexion étant appelée circuit virtuel,

- un accord tri-partie (les deux entités communiquantes + le fournisseur) sur la qualité du service,
- le moyen de transférer des données de manière transparente et performante,
- un contrôle de flux, grâce auquel le récepteur peut réguler la cadence d'émission des données,
- la détection des erreurs de transmission et leur correction par retransmission,
- la resynchronisation des deux entités distantes en cas d'incident majeur intervenant sur la connexion.
- la libération de la connexion réseau,

Par définition la couche Réseau fait abstraction du type de lignes et des procédures de liaison entre les sites formant le réseau de transmission. La couche Liaison est là pour assurer le transfert inter-sites. La couche Réseau n'aperçoit jamais un paquet en transit sur une ligne, un paquet est soit dans un site, soit dans le suivant (ou un autre). Le protocole de Liaison assure la délivrance du paquet au site.

2.2 Le protocole de la couche Réseau

Le protocole de la couche Réseau gère les différents circuits virtuels de chaque entité communiquante, en assurant les fonctions suivantes :

Adressage des différents correspondants : un numéro d'abonné, qui identifie de manière unique, est attribué à tout point de raccordement dans le réseau. Cependant afin d'éviter le transport de l'adresse complète du correspondant dans chacun des paquets, une fois que le circuit virtuel est établi, on utilise un mécanisme d'adressage local abrégé: le numéro de voie logique;

Etablissement et libération des circuits virtuels : l'entité qui prend l'initiative d'établir un circuit virtuel émet un paquet d'appel (N-PDU-demande de connexion) en direction de l'entité distante. Cette entité distante sur réception de ce paquet répondra favorablement par un paquet d'acceptation de connexion (N-PDU-acceptation de connexion), ou refusera par une paquet de libération

(N-PDU-libération). La libération de la connexion est provoquée par l'émission d'un paquet de libération ,et est confirmée par un paquet de confirmation (N-PDU-confirmation de libération);

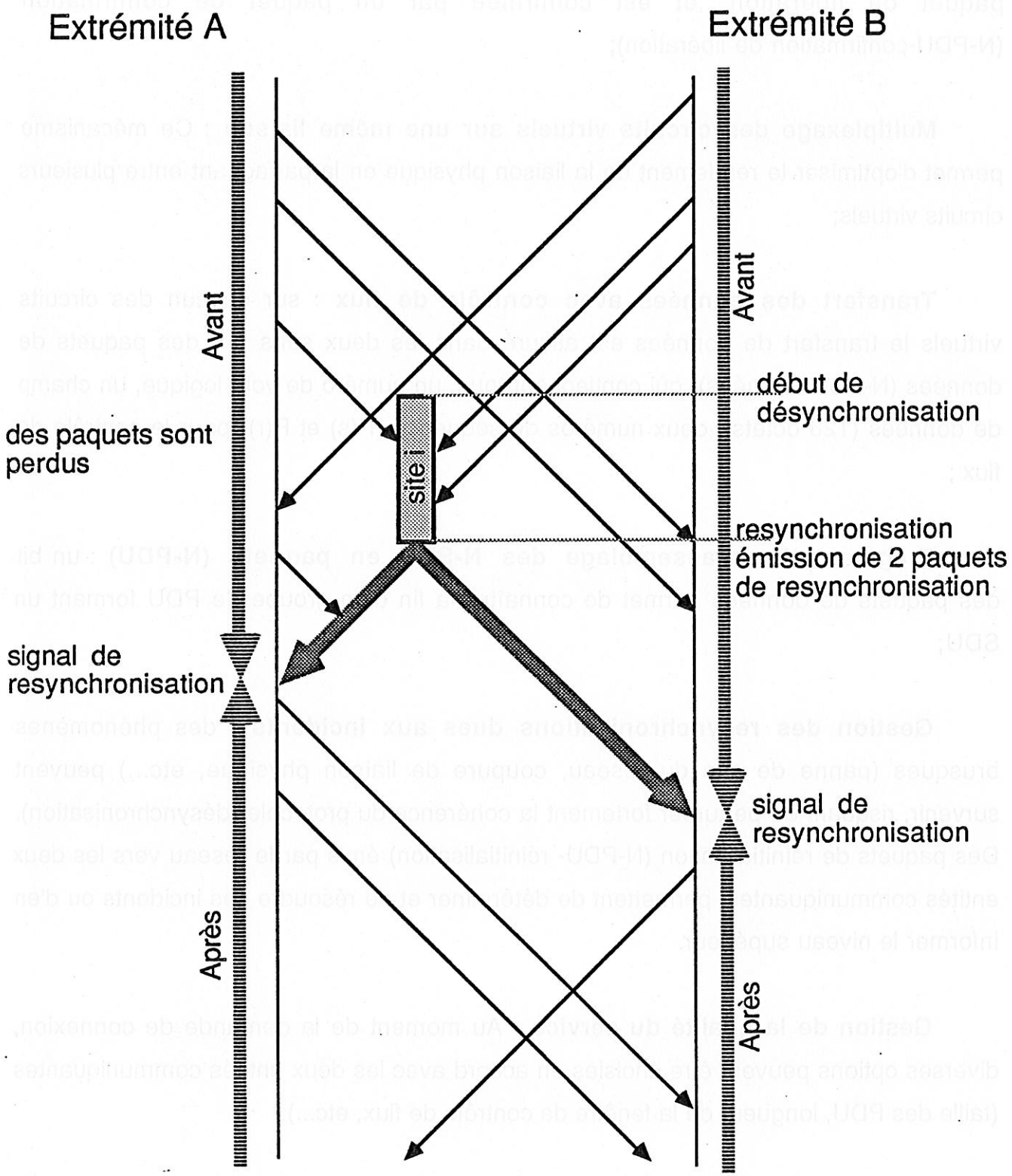
Multiplexage des circuits virtuels sur une même liaison : Ce mécanisme permet d'optimiser le rendement de la liaison physique en la partageant entre plusieurs circuits virtuels;

Transfert des données avec contrôle de flux : sur chacun des circuits virtuels le transfert de données est assuré dans les deux sens par des paquets de données (N-PDU-données), qui contiennent tous, un numéro de voie logique, un champ de données (128 octets), deux numéros de séquence (P(s) et P(r)) pour le contrôle de flux ;

Fragmentation/réassemblage des N-SDU en paquets (N-PDU) : un bit des paquets de données permet de connaître la fin d'un groupe de PDU formant un SDU;

Gestion des resynchronisations dues aux incidents : des phénomènes brusques (panne de site du réseau, coupure de liaison physique, etc...) peuvent survenir, risquant de perturber fortement la cohérence du protocole (désynchronisation). Des paquets de réinitialisation (N-PDU- réinitialisation) émis par le réseau vers les deux entités communiquantes, permettent de déterminer et de résoudre ces incidents ou d'en informer le niveau supérieur.

Gestion de la qualité du service : Au moment de la demande de connexion, diverses options peuvent être choisies en accord avec les deux entités communiquantes (taille des PDU, longueur de la fenêtre de contrôle de flux, etc...).



- B-2- Désynchronisation d'un site -

2.3 Le Service de la couche Réseau

Le service Réseau n'est qu'une vue externe des fonctionnalités induites par le protocole de la couche Réseau. De ce fait il n'est pas utile de modéliser les mécanismes internes à cette couche (routage, transmission, gestion de flux, etc...).

Notre propos est de définir les propriétés essentielles, provoquant des phénomènes perceptibles durant la phase de transfert de données du protocole de la couche Transport, et qui sont garanties par le service de la couche Réseau. C'est pourquoi nous étudions le service de la couche Réseau en phase de transfert de données. Les primitives qui s'échangent entre le protocole Transport et le service Réseau durant la phase de transfert de données sont les N-SDU-données.requête, et les N-SDU-données.indication.

Une fois la connexion réseau établie au travers des couches inférieures, on peut la considérer comme une liaison directe, et néanmoins virtuelle, entre les deux entités distantes. Cette liaison a quelques propriétés, notamment elle conserve la séquentialité des paquets et ne les duplique pas, mais elle peut perdre des paquets.

Au cours de la phase de transfert, divers événements peuvent se produire:

- Une demande de déconnexion du circuit virtuel par l'un des partenaires (N-SDU-libération.requête).
- Une déconnexion du circuit virtuel provenant d'une impossibilité à maintenir la qualité de la connexion (N-SDU-libération.indication).
- Une réinitialisation de la connexion Réseau produite par la détection d'une perte de synchronisation (N-SDU-réinitialisation. indication). Nous n'étudions pas les deux premiers cas, ici, car ils participent à la phase de déconnexion de la couche réseau, et leurs traitements séquentiels sont simples.

Nous nous intéressons tout particulièrement à la gestion des désynchronisations du réseau de transmission (Figure B-2). Ces désynchronisations sont provoquées par

des pannes de sites intermédiaires intervenant pendant la transmission. Ces pannes provoquent la perte des informations en transit (paquets) à travers ce site, mais aussi la perte des connaissances nécessaires à une bonne gestion du réseau de transmission. Passé un certain délai, le réseau se reconfigure. Ce traitement des pannes génère des paquets de réinitialisation pour chacun des circuits virtuels qui traversaient le site au moment de la panne. Ces paquets sont destinés à la couche supérieure (Transport), qui sera ainsi prévenue de l'incident (N-SDU-réinitialisation).

Cependant un problème peut survenir, les paquets de réinitialisation transitant dans le réseau au même titre que les autres paquets peuvent aussi être perdus. On doit donc s'assurer que les paquets de réinitialisation parviennent bien aux deux extrémités communiquantes, et qu'ils permettent une resynchronisation efficace de tout le réseau.

2.4 Les propriétés du service de la couche Réseau

Notre étude basée sur les normes du service de la couche Réseau porte sur la phase de transfert de donnée en mode connexion. Le service doit assurer la transmission bidirectionnelle des données émises et veiller au respect des directives qui définissent le comportement spécifique du réseau face aux désynchronisations.

D'un point de vue fonctionnel, le transfert de paquets pour les réseaux de type Transpac (à contrario des réseaux de type Datagramme), peut être assimilé à la gestion d'une file dont chaque emplacement correspond à la zone de mémorisation d'un paquet traité par un site du réseau de télécommunication. Les paquets d'informations transitent sur des sites intermédiaires avant d'être délivrés à leur destinataire. La relation d'ordre, induite par la file, est l'interprétation de la chronologie des arrivées des paquets sur la connexion réseau.

Le service de transmission est bidirectionnel. La description d'un emplacement doit donc être complétée par la caractérisation de son sens de transmission (voie1, voie2). Chaque site intermédiaire du réseau de télécommunication est donc modélisé

par un double emplacement qui représente l'organe de stockage de chacun des deux sens de transmission.

Pendant la phase de transmission de données, les deux voies fonctionnent de manière indépendante. Seule la phase de resynchronisation gère de manière similaire les deux voies.

Au cours de la phase de transfert, un incident (panne de station, état incohérent du protocole, etc...) peut provoquer une désynchronisation qui perturbe éventuellement le transfert des paquets. Les zones de mémorisation des paquets peuvent alors contenir des informations incohérentes. Les emplacements qui leur correspondent sont, soit dans l'état synchronisé (c'est le fonctionnement normal de la station), soit dans l'état désynchronisé en cas d'erreur. Chaque emplacement est donc caractérisé, en plus des éléments précédemment décrits, par son état (synchronisé, désynchronisé).

Le service Réseau intègre la perte de paquets de tous types (donnée, réinitialisation, etc...), mais il assure qu'après une phase de désynchronisation (pendant laquelle des paquets ont été perdus) survient une phase de resynchronisation. Cette phase permet de prévenir les deux extrémités réceptrices par l'intermédiaire de paquets de réinitialisation.

Le service Réseau va être illustré par quatre propriétés, les deux premières sont relatives au fonctionnement du service dans un état cohérent : l'avancement des paquets de données à travers le réseau se fait de manière séquentielle et sans duplication ; La troisième propriété est relative à la gestion des incidents par le service : les paquets de réinitialisation permettent la signalisation de cet événement; A ces trois propriétés nous rajoutons une dernière, qui ne figure pas explicitement dans la norme, mais qui est toujours sous-entendue : le service est toujours assuré.

Nous définissons par :

- Te(p) la date d'émission du paquet p, c'est-à-dire le franchissement de l'interface du service Réseau par la N-SDU-donnée.requête p.

- Tr(p) la date de réception du paquet p, c'est-à-dire le franchissement de l'interface du service Réseau par la N-SDU-donnée.indication p. Ti(p) la date de réception d'un signal de réinitialisation, c'est-à-dire le franchissement de l'interface du service Réseau par la N-SDU- réinitialisation.indication p.

- opp(r) le signal de resynchronisation délivré à l'extrémité du circuit virtuel opposée à celle où le signal r à lui-même été délivré.

Les quatre propriétés s'expriment alors de la manière suivante:

Propriété R-1 : le service Réseau ne désordonnance pas les paquets durant le transfert;

Quelque soient pi et pj deux paquets, si Te(pi)>Te(pj) alors Tr(pi)>Tr(pj).

Propriété R-2 : le service Réseau ne duplique pas les paquets durant leur transfert;

Quelque soient deux paquets p1 et p2,

si Te(p1) ≠Te(p2) alors Tr(p1) ≠Tr(p2).

Propriété R-3 : le service Réseau gère les désynchronisations.

- Un paquet de réinitialisation est transmis aux deux extrémités de la connexion.

- Les paquets soumis au réseau avant la réception d'un paquet de réinitialisation sont, soit délivrés avant la réception d'un paquet de réinitialisation , soit détruits.

- Les paquets soumis au réseau après la réception d'un paquet de réinitialisation

sont délivrés à l'autre extrémité, après que celle-ci ait reçu un paquet de réinitialisation.

Quelque soit p un paquet, et r un signal de resynchronisation:
si Te(p)<Ti(r) alors
- soit Tr(p)<Ti(opp(r)),
- soit il n'existe pas Tr(p);

sinon si Te(p)>Ti(r) alors
- Tr(p)>Ti(opp(r)).

Propriété R-4 : le service Réseau est toujours susceptible de transférer des paquets (il n'est jamais en situation de blocage irrémédiable).

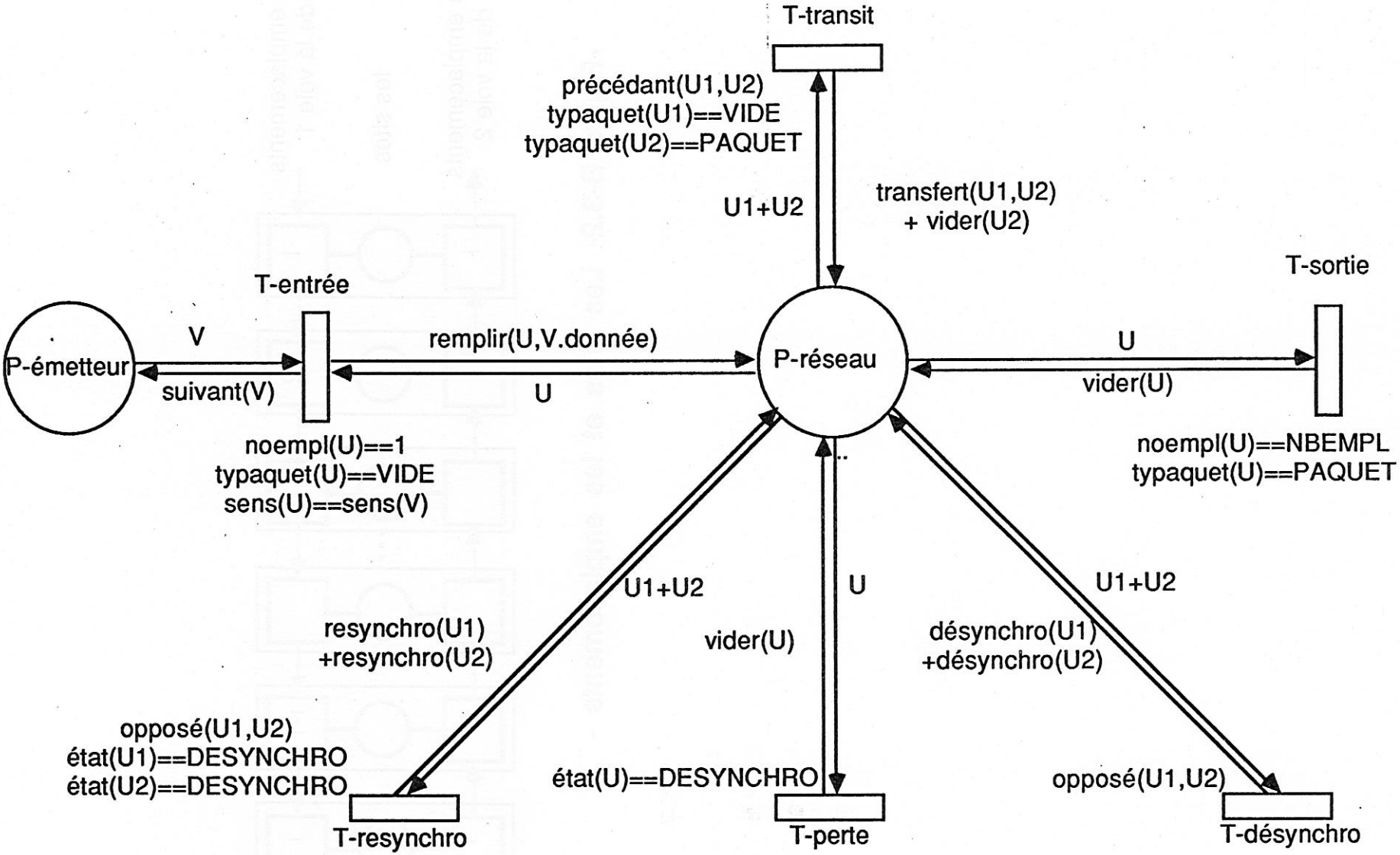
3. Le MODELE

3.1 Introduction

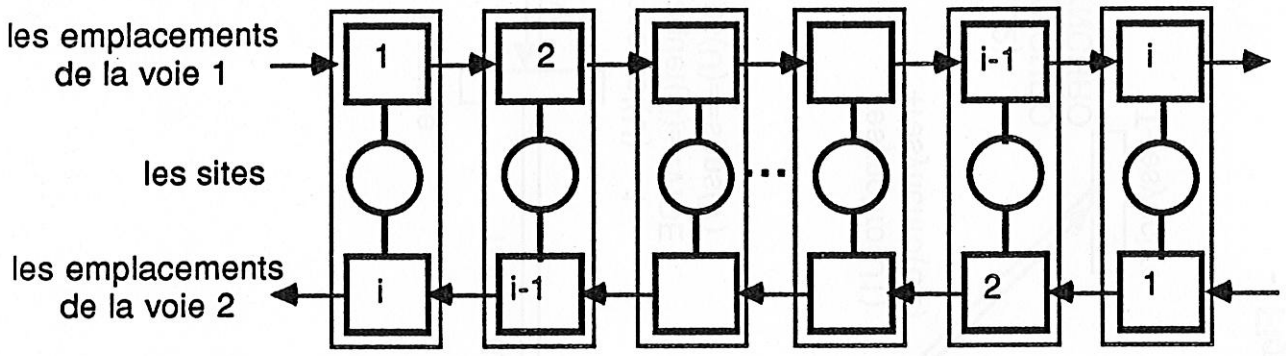
Nous allons maintenant modéliser le service offert pendant la phase de transfert de données de le couche Réseau : la transmission bidirectionnelle des paquets de données et la gestion des désynchronisations du réseau. Ce paragraphe constitue la deuxième étape de notre méthodologie (la modélisation).

Nous allons entreprendre ici, la description du modèle du service de la couche Réseau (Figure B-3.1), en énumérant et expliquant les places, les transitions et les marques formant le modèle.

Le modèle du service que nous proposons est basé sur le mécanisme d'une file FIFO qui représente la transmission des données au travers du réseau de télécommunication. Le modèle de la file a été déjà largement utilisé à cette fin [Berthelot 81]. Nous proposons une modélisation équivalente sous une forme très condensée (1 place, 6 transitions) propice à notre démarche. Chaque emplacement est caractérisé par son numéro d'ordre (noempl) et par l'éventuel paquet qu'il contient (info). Chaque emplacement peut donc contenir, soit un paquet de données (DONNEE), soit un paquet de réinitialisation (REINIT), soit être vide (VIDE). La file contient un nombre fini d'emplacements (NBEMPL). Cette limite correspond au nombre maximum de paquets pouvant circuler simultanément sur chaque connexion réseau.



- Figure B-3.1 - Le Modèle du Service Réseau



-Figure B-3.2- Les sites et les emplacements -

Le service de transmission est bidirectionnel. La description d'un emplacement doit donc être complétée par la caractérisation de sa voie de transfert (VOIE1, VOIE2). Chaque site du réseau de télécommunication est donc modélisé par un double emplacement qui représente l'organe de stockage de chacun des deux sens de transmission.

Du fait des incidents pouvant survenir sur le réseau chaque site est, soit dans l'état synchronisé (SYNCHRO)(c'est le fonctionnement normal de la station), soit dans l'état désynchronisé en cas d'erreur (DESYNCHRO). Chaque site est donc caractérisé, en plus des éléments précédemment décrits, par son état.

3.2 Les Places

Un paquet entre dans le réseau à l'initiative de la couche Transport. La donnée qu'il contient permet l'identification des paquets qui circulent sur le réseau. Cette identification est réalisée à l'aide d'un compteur géré par la couche Transport (P-émetteur). Ce compteur est incrémenté après chaque nouvelle émission. Un paquet de données transitant sur le réseau est ainsi identifié par un numéro unique (nodonnée). Le plus grand numéro correspond au dernier message émis par la couche Transport. Ce compteur est uniquement ajouté au modèle afin d'exprimer les propriétés du service, sans toutefois en perturber le fonctionnement.

La place P-réseau modélise l'état d'un circuit virtuel du service Réseau. Elle représente à la fois l'ensemble des sites intermédiaires qui constituent le réseau, le circuit virtuel bi-directionnel qui relie les deux extrémités communiquantes et l'ensemble des paquets circulant à travers le réseau.

Le réseau de transmission est donc modélisé comme une suite de sites. Le nombre de sites formant le circuit virtuel est fixé à NBEMPL. Chaque site possède deux structures de stockage appelées emplacements, une pour chacune de deux voies de transmission du circuit virtuel.

Tous les emplacements d'une même voie sont numérotés de 1 à NBEMPL en fonction de leur place dans la file formant le circuit virtuel (Figure B-3.2). L'emplacement numéro 1 est du côté de l'émetteur, celui numéro NBEMPL étant du côté récepteur. De ce fait, le numéro de l'emplacement appartenant au même site mais à la voie opposée à l'emplacement de numéro i, est (NBEMPL-i+1).

Nous avons fixé la capacité de stockage de chaque emplacement a un seul paquet. Cette modélisation présente l'avantage d'offrir le plus de souplesse d'interprétation possible. Elle recouvre l'ensemble des autres cas de figure, où les sites ont la possibilité de stocker de nombreux paquets. Pour modéliser ce comportement, il suffit de franchir de manière successive l'ensemble des transitions modifiant l'état d'un même site.

3.3 Les Transitions

Le paquet peut être déposé (T-entrée) en début de la file des paquets (P-réseau) dès que l'emplacement numéro 1 est libre. Le paquet progresse ensuite (T-transit) dans la file en fonction des emplacements qui se libèrent. Il peut enfin être délivré (T-sortie) à son destinataire lorsqu'il atteint l'emplacement numéro NBEMPL (le dernier de la file). Le paquet est alors transmis au récepteur et l'emplacement qui le contenait est désormais vide.

La normalisation du service Réseau fait apparaître à l'interfaces de la couche des primitives, appelé N-SDU. La transition T-entrée (resp. T-sortie) modélise la primitive N-SDU-Data-requête (resp. N-SDU-Data-indication et N-SDU-Reset-indication). Ces deux transitions n'appartiennent pas à proprement parler au service Réseau, car elles symbolisent l'interface Réseau/Transport. Elles servent comme la place P-émetteur à la dynamique du modèle (sans la génération de paquets le modèle serait immobile), et à l'établissement des preuves (détection du franchissement de l'interface).

Toute perturbation du comportement d'un site provoque éventuellement la désynchronisation (T-désynchro) au niveau des deux emplacements correspondants qui

passent d'un état synchronisé à l'état désynchronisé. A partir de cet instant, chaque paquet transitant par un de ces emplacements peut éventuellement être perdu (T-perte). L'emplacement redevient alors vide. Dès qu'il détecte la désynchronisation, le service réseau, après un traitement de l'incident, émet (T-resynchro) un paquet de réinitialisation sur chacune des deux voies. Les deux emplacements du même site retournent alors à l'état synchronisé

On note qu'une fois émis, le paquet de réinitialisation subit le traitement de tout autre paquet en progressant dans la file de manière séquentielle. La perte d'un paquet de réinitialisation peut survenir pendant son transfert sur le Réseau. Toutefois, du fait de la récursivité du phénomène de resynchronisation, cette perte régénère inévitablement (mais ultérieurement) un paquet de réinitialisation sur la voie de télécommunication.

Dans leur phase de transmission de données, les deux sens de communication sont asynchrones. Sur ces voies, les traitements d'entrée (T-entrée), de transit (T-transit) et de sortie (T-sortie) des paquets sont indépendants. Le mauvais fonctionnement d'un site est répercuté (T-désynchro) au niveau des deux voies par la désynchronisation des deux emplacements qui correspondent au site. Les emplacements d'émission et de réception passent alors simultanément dans l'état désynchronisé. Sur chacune des voies peut alors se produire une perte (T-Perte) des paquets qui transitent dans le site défaillant. Le contenu de ces emplacements est détruit. De telles pertes se produisent indépendamment sur les deux voies du même site. A la détection de la désynchronisation, après correction, le service réseau émet (T-resynchro) un paquet de resynchronisation simultanément sur chacun des deux sens de communication.

3.4 Les Marques

Nous allons récapituler ici la structure des marques qui circulent à travers le modèle. Ces marques forment en fait des uplets, pour lesquels on associe à chaque champ en ensemble de valeurs possibles et une signification.

On note en majuscules les constantes, et en minuscules les variables qui prennent valeur dans un ensemble.

Structure des uplets:

Les deux marques de la place P-émetteur sont définies par le 2-uplet <donnée,voie> :

- donnée : représente la valeur du compteur d'émission, donnée ∈ ℕ ;
- voie : représente l'appartenance du compteur à l'un des sens de transmission, voie ∈ {VOIE1,VOIE2} .

Chaque marque de la place P-réseau est un emplacement défini par un 4- uplet <noempl,info,état,voie> :

- noempl : représente le numéro d'emplacement,

  noempl ∈ [1..NBEMPL] ;
- info : représente l'information contenue par l'emplacement, on a :

  info ∈ {VIDE,paquet},

  paquet ∈ {REINIT,donnée},

  donnée ∈ ℕ ;
- état : représente l'état de l'emplacement,

  état ∈ {SYNCHRO,DESYNCHRO};
- voie : représente l'appartenance de l'uplet à l'un des deux sens de transmission,

  voie ∈ {VOIE1,VOIE2}.

Notation : Pour toute marque ayant une structure d'uplet, il est utile de pouvoir accéder à chacun de ses champs séparément. Pour ce faire nous proposons comme notation de postfixer la marque par le nom du champ à atteindre. Par exemple : L'état d'une marque M de la place P-réseau sera exprimé par M.état.

Nous allons établir le marquage initial Mo du modèle du service de la couche Réseau :

- $\forall$ u $\in$ Mo(P-réseau), u.info = VIDE, u.état = SYNCHRO . Il n'y a aucun paquet circulant sur le réseau, et chaque site est dans un état sans panne.

- $\forall$ u $\in$ Mo(P-émetteur), u.donnée = 0 . La couche Transport n'a encore émis aucune donnée.

3.5 Les Prédicats

Le modèle comporte aussi un certain nombre de prédicats sous la forme de fonctions que nous allons expliquer et définir formellement.

La fonction indique si les deux emplacements appartiennent au même site :
opposé(u1,u2) =

{ (u1.voie≠u2.voie) && (u1.noempl==NBEMPL+1-u2.noempl) }

La fonction indique si l'emplacement u1 précède l'emplacement u2:

précédant(u1,u2) = { (u1.voie==u2.voie) && (u1.noempl==u2.noempl+1) }

La fonction remplit l'emplacement u avec la donnée d :

remplir(u,d) = { u.voie = u.voie , u.noempl = u.noempl , u.état = u.état ,
u.donnée = d }

La fonction vide le contenu de l'emplacement u :

vider(u) = { remplir(u,VIDE) }

La fonction place l'emplacement dans un état synchronisé :

synchro(u) = { u.voie = u.voie , u.noempl = u.noempl , u.état = SYNCHRO , u.donnée = u.donnée }

La fonction place l'emplacement dans un état désynchronisé :

désynchro(u) = { u.voie = u.voie , u.noempl = u.noempl ,
u.état = DESYNCHRO , u.donnée = u.donnée }

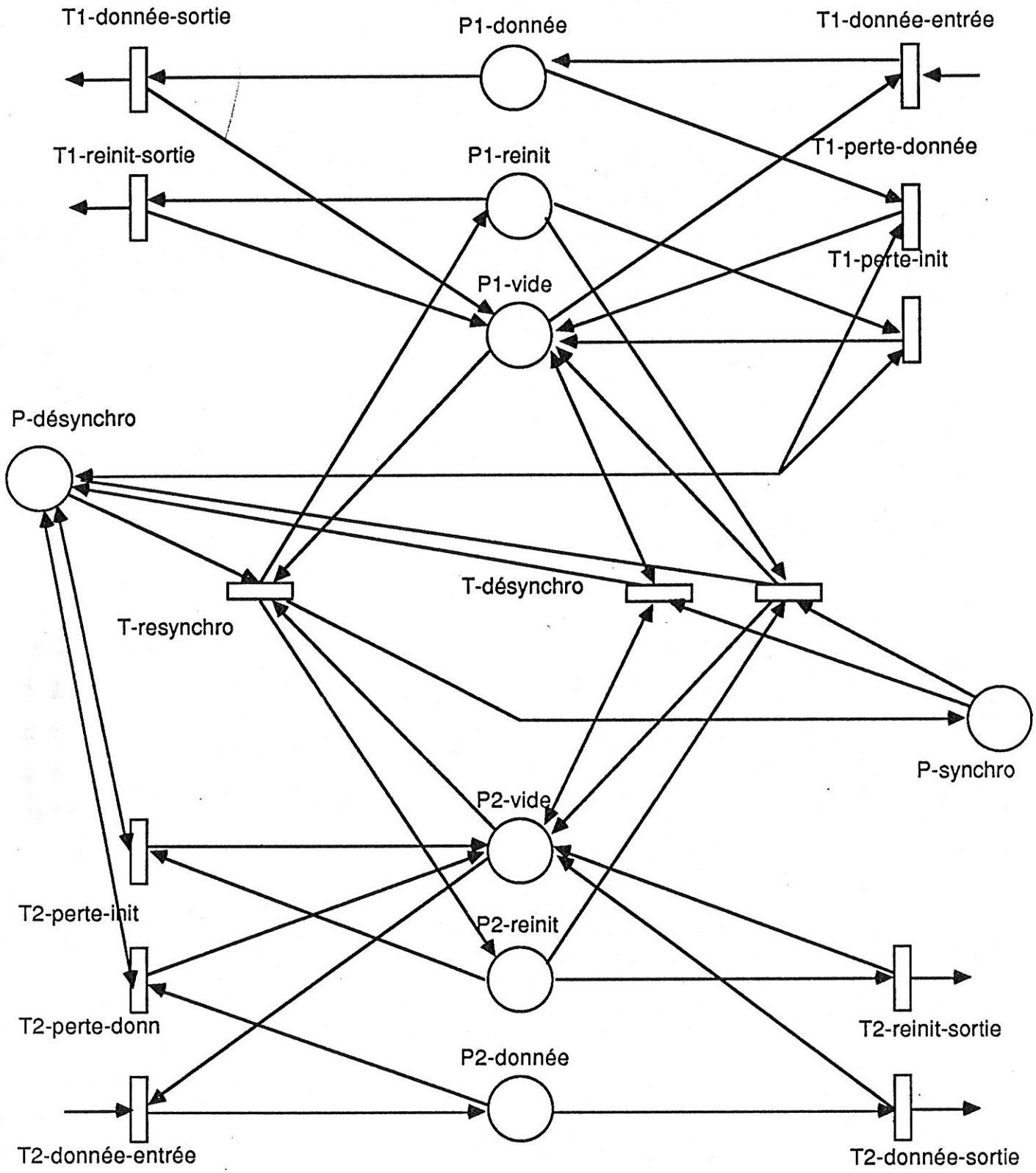
La fonction échange le contenu de 2 emplacements :
transfert(u,v) = { donnée = u.donnée, remplir(u,v.donnée),
remplir(v,donnée) }

La fonction permet de générer le paquet suivant :
suivant(v) = { v. voie = v.voie, v.donnée = v.donnée + 1 }

3.6 Conclusion

Nous venons de construire un modèle du service Réseau. Ce modèle peut sembler simpliste, mais s'il ne possède que deux places et six transitions, cette apparence est due à la puissance d'expression des Réseaux de Petri à Prédicats, qui permettent un pliage important des places et des transitions. Le modèle déplié d'un seul site intermédiaire du réseau de transmission devrait persuader aisément quiconque, de la complexité réelle du système modélisé, et de l'intérêt d'avoir utilisé les RdPàP (Figure B-3.3). Ce modèle possède huit places et vingt-deux transitions.

Ce modèle compact va maintenant être validé dans le chapitre suivant. Cette compacité va être mise à profit pour faciliter l'établissement des preuves.



- Figure B-3.3 - Le modèle déplié d'un site -

4. VALIDATION du MODELE

4.1 Introduction

Le modèle du service de la couche Réseau, doit être validé. Dans ce chapitre, nous montrons que le modèle respecte les propriétés essentielles définies dans la norme, qui caractérisent ce service.

Nous introduisons d'abord l'ensemble des notations nécessaires à notre preuve, puis nous proposons un ensemble de propriétés, exprimées à l'aide de lemmes et de théorèmes, qui concrétisent, au niveau du modèle décrit dans le paragraphe précédent, les propriétés du service Réseau.

Après avoir expliqué la démarche suivie pour obtenir les preuves des théorèmes, nous prouvons formellement que le modèle possède bien l'ensemble des lemmes et théorèmes précédemment décrits. Cette obtention des propriétés du modèle constitue la troisième étape de notre méthodologie.

Enfin nous exprimons les quatre propriétés définissant le service à partir des théorèmes et des lemmes issus du modèle du service. Cette ultime et quatrième étape de notre méthodologie permet de prouver la concordance du modèle. Nous prouvons ainsi que le modèle du service est conforme à la norme, et qu'il est équivalent, dans ce sens, à tout modèle du protocole associé.

4.2 Les Propriétés du modèle

4.21 Les Notations

Fonctions

Afin de manipuler aisément les uplets du modèle, nous définissons les fonctions suivantes:

- Suivant(ui,P): la fonction rend l'uplet contenant un paquet suivant l'uplet ui parmi l'ensemble P. Dans le modèle :

Soit S l'ensemble de tout paquet émis après le paquet ui:

S ={ uk/uk ∈ P, typaquet(uk)≠VIDE et noempl(uk)<noempl(ui) }

Suivant(ui,P)=uj si et seulement si l'uplet uj est le plus proche de l'uplet ui: ∀ uk ∈ S noempl(uj)>=noempl(uk).

- Dernier(P): la fonction rend l'uplet contenant le paquet le plus proche de l'émetteur (il comporte le numéro d'emplacement le plus petit parmi l'ensemble P). Dans le modèle :

Dernier(P)=ui ssi ui ∈ P, typaquet(ui)=DONNEE et ∀ uj ∈ P tel que typaquet(uj)=DONNEE, noempl(ui)<=noempl(uj).

- Premier(P): la fonction rend l'uplet contenant le paquet le plus proche du récepteur (il comporte le numéro d'emplacement le plus grand de l'ensemble P). Dans le modèle :

Premier(P)=ui ssi ui ∈ P, typaquet(ui)=DONNEE et ∀ uj ∈ P tel que typaquet(uj)=DONNEE, noempl(ui)>=noempl(uj).

- Voie1(P): la fonction rend l'ensemble des uplets appartenant à la voie de transmission 'VOIE1' de l'ensemble P. Dans le modèle :

Voie1(P)= { ui ∈ P tel que voie(ui)=VOIE1}.

- Voie2(P): la fonction rend l'ensemble des uplets appartenant à la voie de transmission 'VOIE2' de l'ensemble P. Dans le modèle :

Voie2(P)= { ui ∈ P tel que voie(ui)=VOIE2}.

Principes

Pour faciliter la démonstration des lemmes et des théorèmes, nous avons suivi la démarche suivante pour chaque assertion:

- la démonstration qu'à l'état initial, l'assertion est vérifiée.
- puis en explorant, l'ensemble des transitions du modèle (dans l'ordre T-entrée, T-transit, T-sortie, T-désynchro, T-perte, T-resynchro), nous prouvons que l'assertion est conservée par le franchissement de la dite transition. La preuve de la conservation d'une assertion, par le franchissement d'une transition donnée est généralement apportée en considérant l'ensemble U des uplets devant vérifier l'assertion. L'ensemble des uplets appartenant à U est exploré en traitant:
- d'une part les uplets n'étant pas intervenus dans le franchissement de la transition,
- d'autre part les uplets issus du franchissement de la transition.

Nous obtenons pour chaque assertion la structure suivante: Soit U l'ensemble des uplets devant vérifier l'assertion.

-1. Preuve de l'état initial.

-2. Preuve de la conservation pour l'ensemble des transitions du modèle. Pour chaque transition du modèle:

- Preuve de la conservation de l'assertion d'une transition. Soit u l'ensemble des uplets issus du franchissement de la transition.

-2.1 Preuve des uplets éléments de U-u.

-2.2 Preuve des uplets éléments de u.

Notations

Afin de faciliter les notations, nous attribuons des noms génériques aux différents uplets, nous permettant de prouver nos théorèmes.

Pour chaque assertion : On appelle ui,uj,uk les uplets vérifiant les propriétés correspondant aux libellés de ces assertions.

Pour chaque transition : On appelle u1,u2,u3,u4 les uplets impliqués dans le franchissement de la transition.

On note M la fonction de marquage du modèle avant le franchissement, et par M', la fonction de marquage après le franchissement de la transition. On note le franchissement de la transition t à partir du marquage M par : M(T>M'.

On note par le signe +(-), l'adjonction (la suppression) d'un uplet au marquage d' une place.

Nous appelons, A l'ensemble des fonctions de marquage accessible à partir du marquage initial Mo du modèle, et T l'ensemble des transitions du modèle.

4.22 Les Théorèmes

Nous allons exprimer les quatre propriétés caractérisant les services Réseau, sous forme de théorèmes s'appuyant sur le modèle de la figure B-3.1. Nous rappelons les 4 propriétés du service :

- la couche Réseau conserve la séquentialité des paquets.
- la couche Réseau ne duplique pas les paquets.
- toute désynchronisation est correctement détectée.
- le modèle de la couche Réseau est vivant.

L'ensemble des quatre théorèmes T1,T2,T3,T4 caractérise l'expression du service à travers le modèle. Nous allons dans un premier temps nous contenter de les exprimer en leur associant une petite explication quant à leur interprétation. La preuve des théorèmes se trouve après la preuve des lemmes qu'ils nécessitent.

Théorème 1: le modèle conserve la séquentialité des paquets.

T1: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} , $\forall$ ui $\in$ Voie(M(P-réseau)) , $\forall$ uj $\in$ Voie(M(P-réseau)) avec typaquet(ui)=DONNEE et

typaquet(uj)=DONNEE

si noempl(ui)>=noempl(uj) alors nodonnée(ui)<=nodonnée(uj).

Interprétation: Soient 2 emplacements contenant des paquets de données,le paquet le plus près du récepteur (de numéro d'emplacement le plus grand) est le paquet ayant été émis le plus tôt (de numéro de paquet le plus petit).

Théorème 2: le modèle ne duplique pas les paquets.

T2: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} , $\forall$ ui $\in$ Voie(M(P-réseau)) , $\forall$ uj $\in$ Voie(M(P-réseau)) avec typaquet(ui)=DONNEE et typaquet(uj)=DONNEE ,

si noempl(ui)$\neq$noempl(uj) alors nodonnée(ui)$\neq$nodonnée(uj).

Interprétation: Soient deux emplacements contenant des paquets de données, s'ils sont différents alors ils contiennent des données différentes.

Théorème 3: toute désynchronisation est correctement détectée.

T3: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} , $\forall$ ui $\in$ Voie(M(P-réseau)) avec typaquet(ui) = DONNEE, si $\exists$ uj $\in$ Voie(M(P-réseau)) tel que Suivant(ui,Voie(M(P-réseau)))=uj , typaquet(uj)=DONNEE et nodonnée(ui)≠nodonnée(uj)-1 alors :

- T3.1: soit $\exists$ uk $\in$ Voie(M(P-réseau)) tel que typaquet(uk)=REINIT et noempl(uk)<noempl(ui);
- T3.2: soit $\exists$ uk $\in$ Voie(M(P-réseau)) tel que état(uk)=DESYNCHRO et noempl(uk)<noempl(ui);

Interprétation: soient deux emplacements consécutifs contenant deux paquets de données émis non-successivement (il y a eu une perte entre ces deux paquets):
- soit il existe un paquet de réinitialisation dans un des emplacements suivants;
- soit un des emplacements suivants est encore désynchronisé.

Théorème 4: le modèle est vivant.

T4: $\forall$ M $\in$ A , $\forall$ t $\in$ T , $\exists$ s $\in$ T* tel que M(St> .

Interprétation: toutes les transitions du modèle peuvent toujours être franchissables.

4.23 Les Lemmes

Pour la démonstration des théorèmes précédents, les lemmes suivants doivent être prouvés.

Lemme 1:

Aucun emplacement ne contient un paquet de données ayant un numéro de paquet supérieur au compteur d'émission.

L1: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} ,

$\forall$ ui $\in$ Voie(M(P-réseau)) ,

si typaquet(ui)=DONNEE alors nodonnée(ui) < Voie(M(P-émetteur)).

Démonstration :

A **l'état initial,** la place P-réseau ne contient que des emplacements vides, donc L1 est vérifié par défaut.

Pour les franchissements des différentes transitions du modèle :

- **T-entrée**: le franchissement provoque d'après les arcs et les prédicats:
M(P-émetteur)'=M(P-émetteur)+1 et
M(P-réseau)'=M(P-réseau)-u1+u2 avec
nodonnée(u2)=M(P-émetteur) et noempl(u1)=noempl(u2).

Comme le marquage de M(P-émetteur) s'accroît,

- pour ui$\neq$u2 $\in$ M(P-réseau)', c'est-à-dire $\forall$ ui $\in$ M(P-réseau) avec typaquet(ui)=donnée, le Lemme L1 s'appliquait avant le franchissement de la transition T-entrée sur l'uplet ui donc, nodonnée(ui)<M(P-émetteur)<M(P-émetteur)';

- pour u2 , on a
  nodonnée(u2)=M(P-émetteur)<M(P-émetteur)'.

Le Lemme 1 est alors verifié pour la transition T-entrée.

- Aucune des **autres transitions** du modèle ne manipule ,ni le numéro de paquet (nodonnée), ni le marquage de la place P-émetteur.
  Ainsi le Lemme 1 est trivialement conservé.

Le **lemme L1** étant conservé par chacune des transitions, et vérifié à l'état initial, c'est donc un **invariant** du modèle.

Lemme 2:

Toute phase de désynchronisation (perte de paquet) est déterminée par la présence d'un emplacement désynchronisé dans la place modélisant le Réseau.

L2: $\forall$ M $\in$ A , si M(T-perte> alors
$\exists$ ui $\in$ M(P-réseau) tel que état(ui)=DESYNCHRO .

Démonstration :

Supposons que l'on franchisse la transition T-perte, d'après les règles de franchissement des transitions des réseaux de Petri à prédicats, la valuation des arcs et le prédicat associé à la transition T-perte impliquent la présence d'un uplet d'état désynchronisé issu de la place P-réseau . Ce qui vérifie l'affirmation.

Donc si l'on franchit la transition T-perte, il existe au moins un uplet dans la place P-réseau en état de désynchronisation, **le lemme 2 est donc vérifié.**

Lemme 3:

Toute phase de resynchronisation se concrétise par l'émission sur chacune des deux voies de transmission d'un paquet de resynchronisation.

L3: $\forall$ M $\in$ A , si M(T-resynchro> = M' alors

$\exists$ u1 $\in$ Voie1(M'(P-réseau)) tel que typaquet(u1)=REINIT ,

$\exists$ u2 $\in$ Voie2(M'(P-réseau)) tel que typaquet(u2)=REINIT .

Démonstration :

Le déclenchement de la transition T-resynchro provoque le passage de l'état désynchronisé à l'état synchronisé d'un uplet de la place P-réseau. Cette transition génère un paquet de resynchronisation à la place d'un emplacement vide pour chacune des deux voies de transmission. **Le lemme L3 est vérifié** par définition des règles de franchissement des transitions des réseaux de Petri.

Lemme 4:

Pour chacune des deux voies, chaque marque de la place P-réseau comporte un numéro unique élément de l'intervalle [1..NBEMPL] qui définit le rang de l'emplacement correspondant dans la file modélisée par la place (lemme d'unicité des emplacements).

L4: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} ,

{noempl(uk) tel que uk $\in$ Voie(M(P-réseau))} = [1..NBEMPL].

Démonstration :

A **l'état initial**, on vérifie que pour Voie $\in$ {Voie1,Voie2}:

Mo(P-réseau)={ uk /noempl(uk) ∈ [1..NBEMPL] et état(uk)=SYNCHRO et typaquet(uk)=VIDE } donc le Lemme 4 est vérifié à l'état initial.

D'après le marquage des arcs de toutes les transitions du modèle, et d'après leur interprétation au moment du franchissement selon la théorie des réseaux de Petri, on s'aperçoit que:

- pour tout arc qui enlève une marque de P-réseau avec un numéro d'emplacement donné,
- il existe un arc inverse qui remet une marque de même numéro d'emplacement dans la place P-réseau.

En voici la preuve pour chaque transition du modèle.

- **T-entrée** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-transit** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

- **T-sortie** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-resynchro** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

- **T-perte** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-désynchro** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

Donc, le **Lemme 4 est vérifié** pour le modèle du service de la couche Réseau.

Lemme 5:

Toute désynchronisation affectant le dernier paquet émis est détectée par le service de la couche Réseau.

L5: ∀ M ∈ A , ∀ Voie ∈ {Voie1,Voie2} , ∀ ui ∈ Voie(M(P-réseau))
si Dernier(Voie(M(P-réseau)))=ui
et nodonnée(ui)≠Voie(M(P-émetteur))-1 alors :

- L5.1: soit ∃ uk ∈ Voie(M(P-réseau)) tel que
  typaquet(uk)=REINIT et noempl(uk)<noempl(ui);
- L5.2: soit ∃ uk ∈ Voie(M(P-réseau)) tel que
  état(uk)=DESYNCHRO et noempl(uk)<noempl(ui);

Interprétation: Soit le paquet le plus près de l'émetteur, s'il comporte un numéro de paquet non inférieur de un à la valeur du compteur d'émission (il ne vient pas d'être émis):

- soit il existe un paquet de réinitialisation entre lui et l'émetteur;
- soit il existe un emplacement en état de désynchronisation entre lui et l'émetteur.

Démonstration :

A **l'état initial**, comme la place P-réseau ne contient que des marques de type vide, le lemme L5 est vérifié par défaut.

Vérification de la conservation du lemme, après le franchissement de toutes les transitions du modèle:

- **T-entrée**, son franchissement provoque:
  M(P-émetteur)'=M(P-émetteur)+1 et
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  nodonnée(u2)=M(P-émetteur) et noempl(u1)=noempl(u2)=1.

- Soit ui≠u2,
  l'uplet ui est inchangé par le franchissement de la transition,
  donc s'il vérifiait le lemme avant le franchissement, il existait
  un uplet uk:
  - soit uk≠u2,
    l'uplet uk vérifiait le lemme avant le franchissement, comme
    le franchissement de la transition le laisse inchangé, il
    vérifie le lemme après le franchissement.
  - soit uk=u2,
    il peut donc respecter une des deux propositions:
    - L5.1, impossible pour l'uplet u2 et non modification de la
      proposition par l'uplet u1 car
      typaquet(u1)=VIDE, typaquet(u2)=DONNEE;
    - L5.2,
      l'uplet u1 avant le franchissement de la transition vérifiait
      le lemme, d'après les conditions de déclenchement des transitions
      et du lemme d'unicité des emplacements: état(u1)=état(u2).
      Il découle que l'uplet u2 se substitue à l'uplet u1 pour
      vérifier le lemme L5.
- Soit ui=u2,
  d'après les conditions de franchissement, noempl(u2)=1 et
  typaquet(u2)=donnée , par construction de la fonction Dernier, on a
  Dernier(M(P-réseau)')=u2.
  D'après les prédicats et les arcs associés à la transition, on a

nodonnée(u2)=M(P-émetteur)=M(P-émetteur)'-1,
ce qui vérifie le lemme par défaut, car l'uplet ui ne peut être égal à l'uplet u2.

Nous venons de démontrer le lemme L5 pour la transition T-entrée.

- **T-transit**, qui provoque l'échange suivant :
M(P-réseau)'=M(P-réseau)-u1-u2+u3+u4 , avec
noempl(u1)=noempl(u3) et noempl(u2)=noempl(u4)=noempl(u1)+1,
typaquet(u2)=typaquet(u3)=VIDE et nodonnée(u4)=nodonnée(u1).

- Soit Dernier(M(P-réseau)')=u4, et nodonnée(u4)≠M(P-émetteur)'-1.
D'après les conditions de franchissement de la transition,
noempl(u4)=noempl(u1)+1 et nodonnée(u4)=nodonnée(u1) ,
noempl(u1)=noempl(u3) et typaquet(u3)=VIDE,
en appliquant le lemme L4, on en déduit Dernier(M(P-réseau))=u1.
Comme M(P-émetteur)'=M(P-émetteur)+1, on en déduit
nodonnée(u1)≠M(P-émetteur).
On peut appliquer le lemme L5 à l'uplet u1 qui vérifiait avant le franchissement :
- soit la proposition L5.1 du lemme. Alors il existait un uplet uk tel que: typaquet(uk)=REINIT et noempl(uk)<noempl(u1).
Cet uplet uk ne peut être égal à u2, car typaquet(u2)=VIDE.
L'uplet uk, existe toujours et est inchangé, après le franchissement de la transition T-transit.
L'uplet u4 prend la place de l'uplet u1 pour vérifier la proposition L5.1, d'après les règles de franchissement de la transition T-transit.

- soit la proposition L5.2. Alors il existait un uplet uk tel que:
état(uk)=DESYNCHRO et noempl(uk)<noempl(u2).
L'uplet uk, existe toujours et demeure inchangé après le franchissement de la transition T-transit, car la transition ne

modifie l'état d'aucun uplet du modèle.
L'uplet u4 prend la place de l'uplet u1 pour vérifier la proposition L5.2, d'après les règles de franchissement de la transition T-transit.

- soit Dernier(M(P-réseau)')=ui et ui≠u4, nodonnée(ui)≠M(P-émetteur)'-1.
Les propositions L5.1 ou L5.2 étaient vérifiées avant le franchissement:

∃ uk ∈ M(P-réseau) tel que
soit typaquet(uk)=REINIT et noempl(uk)<noempl(ui) (L5.1),
soit état(uk)=DESYNCHRO et noempl(uk)<noempl(ui) (L5.2).
- si uk≠(u1 ou u2) alors les propositions sont conservées, car l'uplet uk est inchangé par le franchissement de la transition.
- si uk=u1, on vérifiait avant le franchissement,
- soit la proposition L5.1, et
d'après les conditions de déclenchement :
nodonnée(u1)=nodonnée(u4) et noempl(u1)=noempl(u4)-1 ,
noempl(u1)=noempl(u3) et typaquet(u3)=VIDE ;
d'après le lemme 4, on en déduit qu'après le franchissement de la transition u4 vérifie la proposition L5.1.
- soit la proposition L5.2, d'après les conditions de déclenchement, état(u1)=état(u3) et noempl(u1)=noempl(u3), et d'après le lemme L4, on en déduit que l'uplet u3 vérifie la proposition L5.2.
- si uk=u2,
- la proposition L5.1 est impossible car typaquet(u1)=VIDE.
- la proposition L5.2 étant vérifiée avant le franchissement, d'après les arcs et prédicats, état(u2)=état(u4) et noempl(u2)=noempl(u4);
d'après le lemme L4 , on en déduit que l'uplet u4 vérifie la proposition, après le franchissement de la transition.

Le lemme L5 vient d'être vérifié pour le franchissement de la transition T-transit.

- **T-sortie**, qui provoque l'événement suivant:
M(P-réseau)'=M(P-réseau)-u1+u2 avec
noempl(u1)=noempl(u2)=NBEMPL et typaquet(u2)=VIDE.

- Soit ui=Dernier(M(P-réseau)'), ui≠u2, et
nodonnée(ui)≠M(P-émetteur)'-1.

Comme ui ∈ M(P-réseau), il vérifiait le Lemme L5, donc on avait avant le franchissement de la transition T-sortie:

- soit les propositions L5.1 ou L5.2, pour lesquelles existe un uplet uk, qui ne peut être l'uplet u1, car noempl(u1)=noempl(u2)=NBEMPL.
Donc le même uplet uk vérifie les propositions L5.1 ou L5.2 après le franchissement de la transition, car inchangé.
- Soit Dernier(M(P-réseau)')=u2. Or cela est impossible car, typaquet(u2)=VIDE.

Le franchissement de la transition T-sortie conserve donc bien le lemme L5.

- **T-désynchro,** la transition déclenche :
M(P-réseau)'=M(P-réseau)-u1+u2 avec
noempl(u1)=noempl(u2) et état(u2)=DESYNCHRO.

- Soit Dernier(Voie(M(P-réseau)'))=u2, et
nodonnée(u2)≠M(P-émetteur)'-1.
Comme noempl(u1)=noempl(u2) et typaquet(u1)=typaquet(u2), d'après le lemme 4, l'uplet u1 vérifiait le lemme 5 avant le franchissement de la transition:

- Une des propositions L5.1 ou L5.2 était vérifiée. Alors, il existait un uplet uk de la place P-réseau les vérifiant avant le franchissement. Comme uk≠(u1 ou u2), d'après le lemme L4, l'uplet uk vérifie la même proposition après le

franchissement.

- $\forall$ ui $\in$ M(P-réseau)' et noempl(ui)>noempl(u2), alors la proposition L5.2 est au moins vérifiée car par les conditions de franchissement de la transition, état(u2)=DESYNCHRO.

- $\forall$ ui $\in$ M(P-réseau)' et noempl(ui)<noempl(u2), par hypothèse l'uplet ui devait vérifier avant le franchissement le lemme L5.
  Donc l'uplet ui vérifie une des deux propositions L5.1 ou L5.2.
  Alors, avant le franchissement il existait un uplet uk tel que noempl(uk)<noempl(ui).
  D'après les conditions de franchissement, on a noempl(u2)=noempl(u1)>noempl(ui) et avec le lemme L4, il devient évident que l'uplet uk existe et vérifie toujours la proposition L5.2 ou L5.1 après le franchissement de la transition.

Le lemme L5 est conservé par la transition T-désynchro.

- **T-perte**, cette transition provoque:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  noempl(u1)=noempl(u2) et typaquet(u2)=VIDE.

  - soit Dernier(M(P-réseau)')=ui avec ui≠u2, alors on a une démonstration similaire à celle utilisée pour la transition T-désynchro. Car d'après les arcs et prédicats associés à la transition, on a état(u2)=DESYNCHRO et noempl(u1)=noempl(u2), qui laissent inchangé le lemme.

  - soit u2≠Dernier(M(P-réseau)') car typaquet(u2)=VIDE.

La transition T-perte vérifie le lemme L5.

- **T-resynchro,** qui déclenche:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  noempl(u1)=noempl(u2) et typaquet(u2)=REINIT.

La démonstration est identique à celle utilisée pour la transition T-perte, car typaquet(u2)=REINIT qui vérifie la proposition L5.2, noempl(u2)=noempl(u1) qui laisse le lemme inchangé, et Dernier(M(P-réseau)')≠u2 car typaquet(u2)=REINIT.

La transition T-resynchro conserve le Lemme L5.

Le **lemme L5** étant prouvé à l'état initial, et conservé par le franchissement de toutes les transitions du modèle, c'est un bien **invariant.**

4.23 Les Lemmes

Pour la démonstration des théorèmes précédents, les lemmes suivants doivent être prouvés.

Lemme 1:

Aucun emplacement ne contient un paquet de données ayant un numéro de paquet supérieur au compteur d'émission.

L1: ∀ M ∈ A , ∀ Voie ∈ {Voie1,Voie2} ,

∀ ui ∈ Voie(M(P-réseau)) ,

si typaquet(ui)=DONNEE alors nodonnée(ui) < Voie(M(P-émetteur)).

Démonstration :

A **l'état initial,** la place P-réseau ne contient que des emplacements vides, donc L1 est vérifié par défaut.

Pour les franchissements des différentes transitions du modèle :

- **T-entrée**: le franchissement provoque d'après les arcs et les prédicats:
  M(P-émetteur)'=M(P-émetteur)+1 et
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  nodonnée(u2)=M(P-émetteur) et noempl(u1)=noempl(u2).

  Comme le marquage de M(P-émetteur) s'accroît,

  - pour ui≠u2 ∈ M(P-réseau)', c'est-à-dire ∀ ui ∈ M(P-réseau) avec typaquet(ui)=donnée, le Lemme L1 s'appliquait avant le franchissement de la transition T-entrée sur l'uplet ui donc, nodonnée(ui)<M(P-émetteur)<M(P-émetteur)';
  - pour u2 , on a nodonnée(u2)=M(P-émetteur)<M(P-émetteur)'.

  Le Lemme 1 est alors vérifié pour la transition T-entrée.

- Aucune des **autres transitions** du modèle ne manipule ,ni le numéro de paquet (nodonnée), ni le marquage de la place P-émetteur.
  Ainsi le Lemme 1 est trivialement conservé.

Le **lemme L1** étant conservé par chacune des transitions, et vérifié à l'état initial, c'est donc un **invariant** du modèle.

Lemme 2:

Toute phase de désynchronisation (perte de paquet) est déterminée par la présence d'un emplacement désynchronisé dans la place modélisant le Réseau.

L2: ∀ M ∈ A , si M(T-perte> alors

∃ ui ∈ M(P-réseau) tel que état(ui)=DESYNCHRO .

Démonstration :

Supposons que l'on franchisse la transition T-perte, d'après les règles de franchissement des transitions des réseaux de Petri à prédicats, la valuation des arcs et le prédicat associé à la transition T-perte impliquent la présence d'un uplet d'état désynchronisé issu de la place P-réseau . Ce qui vérifie l'affirmation.

Donc si l'on franchit la transition T-perte, il existe au moins un uplet dans la place P-réseau en état de désynchronisation, **le lemme 2 est donc vérifié.**

Lemme 3:

Toute phase de resynchronisation se concrétise par l'émission sur chacune des deux voies de transmission d'un paquet de resynchronisation.

L3: $\forall$ M $\in$ A , si M(T-resynchro> = M' alors

$\exists$ u1 $\in$ Voie1(M'(P-réseau)) tel que typaquet(u1)=REINIT ,

$\exists$ u2 $\in$ Voie2(M'(P-réseau)) tel que typaquet(u2)=REINIT .

Démonstration :

Le déclenchement de la transition T-resynchro provoque le passage de l'état désynchronisé à l'état synchronisé d'un uplet de la place P-réseau. Cette transition génère un paquet de resynchronisation à la place d'un emplacement vide pour chacune des deux voies de transmission. **Le lemme L3 est vérifié** par définition des règles de franchissement des transitions des réseaux de Petri.

Lemme 4:

Pour chacune des deux voies, chaque marque de la place P-réseau comporte un numéro unique élément de l'intervalle [1..NBEMPL] qui définit le rang de l'emplacement correspondant dans la file modélisée par la place (lemme d'unicité des emplacements).

L4: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} ,

{noempl(uk) tel que uk $\in$ Voie(M(P-réseau))} = [1..NBEMPL].

Démonstration :

A **l'état initial**, on vérifie que pour Voie $\in$ {Voie1,Voie2}:

Mo(P-réseau)={ uk /noempl(uk) $\in$ [1..NBEMPL] et état(uk)=SYNCHRO et typaquet(uk)=VIDE } donc le Lemme 4 est vérifié à l'état initial.

D'après le marquage des arcs de toutes les transitions du modèle, et d'après leur interprétation au moment du franchissement selon la théorie des réseaux de Petri, on s'aperçoit que:

- pour tout arc qui enlève une marque de P-réseau avec un numéro d'emplacement donné,
- il existe un arc inverse qui remet une marque de même numéro d'emplacement dans la place P-réseau.

En voici la preuve pour chaque transition du modèle.

- **T-entrée** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-transit** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

- **T-sortie** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-resynchro** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

- **T-perte** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)+u1-u2 avec
  noempl(u1) = noempl(u2)

- **T-désynchro** : le franchissement de cette transition provoque
  M(P-réseau)' = M(P-réseau)-u1-u2+u3+u4
  avec noempl(u1) = noempl(u3) et noempl(u2) = noempl(u4)

Donc, le **Lemme 4 est vérifié** pour le modèle du service de la couche Réseau.

Lemme 5:

Toute désynchronisation affectant le dernier paquet émis est détectée par le service de la couche Réseau.

L5: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} , $\forall$ ui $\in$ Voie(M(P-réseau))
si Dernier(Voie(M(P-réseau)))=ui
et nodonnée(ui)$\neq$Voie(M(P-émetteur))-1 alors :

- L5.1: soit $\exists$ uk $\in$ Voie(M(P-réseau)) tel que
  typaquet(uk)=REINIT et noempl(uk)<noempl(ui);
- L5.2: soit $\exists$ uk $\in$ Voie(M(P-réseau)) tel que
  état(uk)=DESYNCHRO et noempl(uk)<noempl(ui);

Interprétation: Soit le paquet le plus près de l'émetteur, s'il comporte un numéro de paquet non inférieur de un à la valeur du compteur d'émission (il ne vient pas d'être émis):

- soit il existe un paquet de réinitialisation entre lui et l'émetteur;
- soit il existe un emplacement en état de désynchronisation entre lui et l'émetteur.

Démonstration :

A **l'état initial**, comme la place P-réseau ne contient que des marques de type vide, le lemme L5 est vérifié par défaut.

Vérification de la conservation du lemme, après le franchissement de toutes les transitions du modèle:

- **T-entrée**, son franchissement provoque:
  M(P-émetteur)'=M(P-émetteur)+1 et
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  nodonnée(u2)=M(P-émetteur) et noempl(u1)=noempl(u2)=1.

- Soit ui≠u2,
  l'uplet ui est inchangé par le franchissement de la transition, donc s'il vérifiait le lemme avant le franchissement, il existait un uplet uk:
  - soit uk≠u2,
    l'uplet uk vérifiait le lemme avant le franchissement, comme le franchissement de la transition le laisse inchangé, il vérifie le lemme après le franchissement.
  - soit uk=u2,
    il peut donc respecter une des deux propositions:
    - L5.1, impossible pour l'uplet u2 et non modification de la proposition par l'uplet u1 car

typaquet(u1)=VIDE, typaquet(u2)=DONNEE;

- L5.2,

l'uplet u1 avant le franchissement de la transition vérifiait le lemme, d'après les conditions de déclenchement des transitions et du lemme d'unicité des emplacements: état(u1)=état(u2). Il découle que l'uplet u2 se substitue à l'uplet u1 pour vérifier le lemme L5.

- Soit ui=u2,

d'après les conditions de franchissement, noempl(u2)=1 et typaquet(u2)=donnée , par construction de la fonction Dernier, on a Dernier(M(P-réseau)')=u2.

D'après les prédicats et les arcs associés à la transition, on a nodonnée(u2)=M(P-émetteur)=M(P-émetteur)'-1,

ce qui vérifie le lemme par défaut, car l'uplet ui ne peut être égal à l'uplet u2.

Nous venons de démontrer le lemme L5 pour la transition T-entrée.

- **T-transit**, qui provoque l'échange suivant :

M(P-réseau)'=M(P-réseau)-u1-u2+u3+u4 , avec noempl(u1)=noempl(u3) et noempl(u2)=noempl(u4)=noempl(u1)+1, typaquet(u2)=typaquet(u3)=VIDE et nodonnée(u4)=nodonnée(u1).

- Soit Dernier(M(P-réseau)')=u4, et nodonnée(u4)≠M(P-émetteur)'-1.

D'après les conditions de franchissement de la transition, noempl(u4)=noempl(u1)+1 et nodonnée(u4)=nodonnée(u1) , noempl(u1)=noempl(u3) et typaquet(u3)=VIDE, en appliquant le lemme L4, on en déduit Dernier(M(P-réseau))=u1. Comme M(P-émetteur)'=M(P-émetteur)+1, on en déduit nodonnée(u1)≠M(P-émetteur).

On peut appliquer le lemme L5 à l'uplet u1 qui vérifiait avant le franchissement :

- soit la proposition L5.1 du lemme. Alors il existait un uplet uk

tel que: typaquet(uk)=REINIT et noempl(uk)<noempl(u1).
Cet uplet uk ne peut être égal à u2, car typaquet(u2)=VIDE.
L'uplet uk, existe toujours et est inchangé, après le franchissement de la transition T-transit.
L'uplet u4 prend la place de l'uplet u1 pour vérifier la proposition L5.1, d'après les règles de franchissement de la transition T-transit.

- soit la proposition L5.2. Alors il existait un uplet uk tel que:
état(uk)=DESYNCHRO et noempl(uk)<noempl(u2).
L'uplet uk, existe toujours et demeure inchangé après le franchissement de la transition T-transit, car la transition ne modifie l'état d'aucun uplet du modèle.
L'uplet u4 prend la place de l'uplet u1 pour vérifier la proposition L5.2, d'après les règles de franchissement de la transition T-transit.

- soit Dernier(M(P-réseau)')=ui et ui≠u4, nodonnée(ui)≠M(P-émetteur)'-1.
Les propositions L5.1 ou L5.2 étaient vérifiées avant le franchissement :
∃ uk ∈ M(P-réseau) tel que
soit typaquet(uk)=REINIT et noempl(uk)<noempl(ui) (L5.1),
soit état(uk)=DESYNCHRO et noempl(uk)<noempl(ui) (L5.2).
- si uk≠(u1 ou u2) alors les propositions sont conservées, car l'uplet uk est inchangé par le franchissement de la transition.
- si uk=u1, on vérifiait avant le franchissement,
- soit la proposition L5.1, et
d'après les conditions de déclenchement :
nodonnée(u1)=nodonnée(u4) et noempl(u1)=noempl(u4)-1 ,
noempl(u1)=noempl(u3) et typaquet(u3)=VIDE ;
d'après le lemme 4, on en déduit qu'après le franchissement de la transition u4 vérifie la proposition L5.1.
- soit la proposition L5.2, d'après les conditions de déclenchement,
état(u1)=état(u3) et noempl(u1)=noempl(u3), et d'après le lemme

L4, on en déduit que l'uplet u3 vérifie la proposition L5.2.

- si uk=u2,
  - la proposition L5.1 est impossible car typaquet(u1)=VIDE.
  - la proposition L5.2 étant vérifiée avant le franchissement, d'après les arcs et prédicats, état(u2)=état(u4) et noempl(u2)=noempl(u4);
    d'après le lemme L4 , on en déduit que l'uplet u4 vérifie la proposition, après le franchissement de la transition.

Le lemme L5 vient d'être vérifié pour le franchissement de la transition T-transit.

- **T-sortie**, qui provoque l'événement suivant:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  noempl(u1)=noempl(u2)=NBEMPL et typaquet(u2)=VIDE.

  - Soit ui=Dernier(M(P-réseau)'), ui≠u2, et nodonnée(ui)≠M(P-émetteur)'-1.

    Comme ui ∈ M(P-réseau), il vérifiait le Lemme L5, donc on avait avant le franchissement de la transition T-sortie:
    - soit les propositions L5.1 ou L5.2, pour lesquelles existe un uplet uk, qui ne peut être l'uplet u1, car noempl(u1)=noempl(u2)=NBEMPL.
      Donc le même uplet uk vérifie les propositions L5.1 ou L5.2 après le franchissement de la transition, car inchangé.
    - Soit Dernier(M(P-réseau)')=u2. Or cela est impossible car, typaquet(u2)=VIDE.

Le franchissement de la transition T-sortie conserve donc bien le lemme L5.

- **T-désynchro,** la transition déclenche :
  M(P-réseau)'=M(P-réseau)-u1+u2 avec

noempl(u1)=noempl(u2) et état(u2)=DESYNCHRO.

- Soit Dernier(Voie(M(P-réseau)'))=u2, et
  nodonnée(u2)≠M(P-émetteur)'-1.
  Comme noempl(u1)=noempl(u2) et typaquet(u1)=typaquet(u2), d'après le lemme 4, l'uplet u1 vérifiait le lemme 5 avant le franchissement de la transition:

  - Une des propositions L5.1 ou L5.2 était vérifiée. Alors, il existait un uplet uk de la place P-réseau les vérifiant avant le franchissement. Comme uk≠(u1 ou u2), d'après le lemme L4, l'uplet uk vérifie la même proposition après le franchissement.

- ∀ ui ∈ M(P-réseau)' et noempl(ui)>noempl(u2), alors la proposition L5.2 est au moins vérifiée car par les conditions de franchissement de la transition, état(u2)=DESYNCHRO.

- ∀ ui ∈ M(P-réseau)' et noempl(ui)<noempl(u2), par hypothèse l'uplet ui devait vérifier avant le franchissement le lemme L5.
  Donc l'uplet ui vérifie une des deux propositions L5.1 ou L5.2.
  Alors, avant le franchissement il existait un uplet uk tel que noempl(uk)<noempl(ui).
  D'après les conditions de franchissement, on a noempl(u2)=noempl(u1)>noempl(ui) et avec le lemme L4, il devient évident que l'uplet uk existe et vérifie toujours la proposition L5.2 ou L5.1 après le franchissement de la transition.

Le lemme L5 est conservé par la transition T-désynchro.

- **T-perte**, cette transition provoque:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec

noempl(u1)=noempl(u2) et typaquet(u2)=VIDE.

- soit Dernier(M(P-réseau)')=ui avec ui≠u2, alors
  on a une démonstration similaire à celle utilisée pour la transition T-désynchro. Car d'après les arcs et prédicats associés à la transition, on a état(u2)=DESYNCHRO et noempl(u1)=noempl(u2), qui laissent inchangé le lemme.

- soit u2≠Dernier(M(P-réseau)') car typaquet(u2)=VIDE.

La transition T-perte vérifie le lemme L5.

- **T-resynchro,** qui déclenche:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  noempl(u1)=noempl(u2) et typaquet(u2)=REINIT.

La démonstration est identique à celle utilisée pour la transition T-perte, car typaquet(u2)=REINIT qui vérifie la proposition L5.2, noempl(u2)=noempl(u1) qui laisse le lemme inchangé, et Dernier(M(P-réseau)')≠u2 car typaquet(u2)=REINIT.

La transition T-resynchro conserve le Lemme L5.

Le **lemme L5** étant prouvé à l'état initial, et conservé par le franchissement de toutes les transitions du modèle, c'est un bien **invariant.**

Lemme 6:

Les deux organes de stockage d'une même station sont toujours dans le même état.

L6: ∀ M ∈ A, ∀ ui ∈ Voie1(M(P-réseau)), ∀ uj ∈ Voie2(M(P-réseau)),
si noempl(ui)=NBEMPL-noempl(uj)+1 alors état(ui)=état(uj).

Interprétation: Soient deux uplets de chacun des deux sens de transmission, s'ils appartiennent au même site, ils sont dans le même état.

Démonstration :

A l'état initial, ∀ ui ∈ Mo(P-réseau) état(ui)=SYNCHRO, donc le lemme 6 est facilement vérifié.

Vérifications de la conservation du lemme, après le franchissement de toutes les transitions du modèle:

Les seules transitions modifiant le champ état des uplets dans le modèle Réseau sont les transitions T-désynchro et T-resynchro. Leurs comportements étant identiques par rapport au lemme, nous les prouverons simultanément. IL suffit donc de substituer l'état DESYNCHRO à l'état SYNCHRO.

- **T-désynchro (T-resynchro)**, à son franchissement le marquage devient:
M(T-désynchro> = M', et
M'(P-réseau) = M(P-réseau) -u1-u2+u3+u4, avec
u1,u3 ∈ VOIE1 et u2,u4 ∈ VOIE2,
noempl(u1)=noempl(opposé(u2))=noempl(u3)=noempl(opposé(u4)),
état(u1) et état(u2) quelconques, état(u3)=état(u4)=DESYNCHRO.

∀ ui ∈ M'(P-réseau),

- soit ui ≠ u3,
  - soit ui ≠ u4,

    alors ui et uj ∈ M(P-réseau), il respectait le lemme avant le franchissement, comme ils sont inchangés par la transition, il le respecte nécessairement après.

    ∃ uj tel que noempl(uj)=noempl(ui) et état(ui)=état(uj).
    Aucune modification de ces uplets n'étant intervenue, et d'après, l'unicité des couples modélisant les stations (Lemme 4), le lemme L6 est ici vérifié.
  - soit uj=u4,
    c'est impossible car d'après les prédicats et les arcs noempl(u3)=noempl(opposé(u4)) or comme ui≠u3 et uj=u4, il y a incohérence.
- soit ui=u3, alors d'après les prédicats et les arcs, on a uj=u4, car noempl(u3)=noempl(opposé(u4)) et d'après l'unicité des emplacements.
  Alors le couple d'uplets u3, u4 vérifie trivialement le lemme 6, car d'après les arcs de la transition :
  état(u3)=état(u4)=DESYNCHRO.

Le lemme 6 est donc vérifié sur les transitions T-désynchro et T-resynchro, et par défaut (ces transitions n'intervenant jamais sur le champ état des uplets) sur toutes les autres transitions. Comme il est aussi vérifié à l'état initial, **le lemme L6 est un invariant** du modèle.

4.24 Les preuves

Dans ce paragraphe, nous allons apporter la preuve que notre modèle possède bien les quatre théorèmes T1,T2,T3,T4. Nous allons utiliser les lemmes du paragraphe précédent.

Théorème 1:

Le modèle conserve la séquentialité des paquets de donnée.

T1: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} ,

$\forall$ ui $\in$ Voie(M(P-réseau)) , $\forall$ uj $\in$ Voie(M(P-réseau)) avec
typaquet(ui)=DONNEE et typaquet(uj)=DONNEE

si noempl(ui)>=noempl(uj) alors nodonnée(ui)<=nodonnée(uj).

Démonstration :

Soit U l'ensemble des uplets respectant les conditions suivantes:
u $\in$ U si u $\in$ Voie(M'(P-réseau)) et typaquet(u)=DONNEE.

A l'état initial Mo, tous les emplacements de la place P-réseau sont vides, donc l'assertion T1 est vérifiée par défaut.

$\forall$ u $\in$ Mo(P-réseau) typaquet(u)=VIDE.

Pour l'ensemble des transitions du modèle :

- **T-entrée**, le déclenchement provoque :
M(P-émetteur)'=M(P-émetteur)+1;

M(P-réseau)'=M(P-réseau)-u1+u2 avec
noempl(u1)=noempl(u2)=1 , typaquet(u2)=DONNEE ,
typaquet(u1)=VIDE et nodonnée(u2)=M(P-émetteur).

- ∀ ui ∈ U-{u2} ,

  - ∀ uj ∈ U-{u2},

    l'assertion étant vérifiée avant le franchissement de la transition, les deux uplets ui et uj n'ayant subit aucune modification par le franchissement, l'assertion T1 est toujours vérifiée.

  - soit uj ∈ {u2},

    d'après la condition de franchissement, on a nodonnée(u2)=M(P-émetteur) et noempl(u2)=1, or

    d'après le lemme L4 et le lemme L1 ,∀ uk ∈ M(P-réseau)' noempl(uk)>=1 et nodonnée(uk)<M(P-émetteur).

    Donc ∀ uk ∈ M(P-réseau)', on a les inégalités suivantes noempl(uk)<=noempl(u2) et nodonnée(uk)<=nodonnée(u2).

  L'assertion T1, pour le franchissement de la transition T-entrée, pour ui différent de l'uplet u2 est vérifiée.

- soit ui ∈ {u2},

  cet événement est impossible, car

  /∃ uk ∈ M(P-réseau)' tel que noempl(uk)<noempl(u2)=1.

Donc le théorème T1 est vérifié pour la transition T-entrée.

- **T-transit**, le déclenchement de la transition provoque :
M(P-réseau)'=M(P-réseau)-u1-u2+u3+u4 avec
noempl(u1)=noempl(u3) et noempl(u2)=noempl(u4)=noempl(u1)+1,
typaquet(u1)=typaquet(u4)=DONNEE et typaquet(u2)=typaquet(u3)=VIDE.

- $\forall$ ui $\in$ U-{u3,u4} ,

  - soit $\forall$ uj $\in$ U-{u3,u4} , alors le type et le contenu des emplacements ui et uj demeurent inchangés malgré le franchissement de la transition, donc le théorème T1 est trivialement conservé.

  - soit uj=u4. Avant le franchissement, le lemme devait être vérifié, notamment par l'uplet u1:
    si noempl(ui)>=noempl(u1) alors
    nodonnée(ui)<=nodonnée(u1).
    Après le franchissement de la transition, d'après les conditions de déclenchement:
    noempl(u4)=noempl(u1)+1, nodonnée(u4)=nodonnée(u1) ,
    noempl(u3)=noempl(u1) et typaquet(u3)=VIDE.
    Donc pour les uplets ui tel que:
    - noempl(ui)>noempl(u1), l'assertion T1 est conservée par l'uplet u4: noempl(ui)>noempl(u4).
    - noempl(ui)=noempl(u1), d'après le Lemme 4 qui identifie les emplacements de manière unique, ui=u3.
    Or typaquet(u3)=VIDE donc le théorème T1 est vérifié par défaut.

  - uj ne peut être égal à u3, car typaquet(u3)=VIDE.

L'assertion T1 est conservée pour ui≠(u3 ou u4) par le franchissement de la transition T-transit.

- soit ui=u4,
  Avant le franchissement, d'après le théorème T1 on avait pour u1:
  $\forall$ uj $\in$ M(P-réseau) si noempl(u1)>=noempl(uj) alors
  nodonnée(u1)<=nodonnée(uj).
  D'après les conditions de franchissement de T-transit:
  noempl(u4)=noempl(u1)+1, nodonnée(u4)=nodonnée(u1).

Donc l'assertion T1 est conservée,pour ui=u4, par la transition T-transit.

- on ne peut avoir ui=u3, car typaquet(u3)=VIDE.

L'assertion T1 est vérifiée pour le franchissement de la transition T-transit.

- **T-sortie**, son déclenchement provoque:
M(P-réseau)'=M(P-réseau)-u1+u2 avec
typaquet(u1)=DONNEE et typaquet(u2)=VIDE,
noempl(u1)=noempl(u2)=NBEMPL.

- ∀ ui ∈ U-{u2},
  - ∀ uj ∈ U-{u2}, l'assertion T1 est vérifiée, car conservée trivialement par l'absence de modification des uplets ui et uj par le franchissement de la transition.
  - soit uj=u2 ,alors , comme typaquet(u2)=VIDE, le théorème T1 est vérifié par défaut.

- soit ui=u2 , de même, typaquet(u2)=VIDE, T1 est vérifié par défaut.

L'assertion T1 est vérifiée par la transition T-sortie.

- **T-désynchro**, son franchissement crée:
M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec
état(u2)=état(u4)=DESYNCHRO , noempl(u1)=noempl(u2) et opposée(u1,u3).

Le comportement étant identique pour les voies VOIE1 et VOIE2, nous les traitons simultanément.

- $\forall$ ui $\in$ U-{u2} ,
  - $\forall$ uj $\in$ U-{u2}, l'assertion T1 est vérifiée, car
    conservée trivialement par la non modification des uplets
    ui et uj par le franchissement de la transition.
  - uj=u2, idem ui=u2.

- Soit ui=u2, d'après les conditions de franchissement,
  typaquet(u1)=typaquet(u2) et noempl(u1)=noempl(u2) .
  Par hypothèse, u1 vérifiait T1 avant le franchissement,
  l'assertion ne porte que sur les numéros d'emplacements et de
  paquets, ces informations étant transmises de u1 a u2, elle
  est conservée par le franchissement de la transition
  T-désynchro.

Donc la transition T-désynchro conserve le théorème T1.

- **T-perte**, qui provoque lors de son franchissement:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  typaquet(u2)=VIDE et noempl(u1)=noempl(u2).

Sa conservation est démontrée de manière identique à la transition T-sortie car typaquet(u2)=VIDE.

- **T-resynchro**, qui provoque:
  M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec opposée(u1,u3) ,
  typaquet(u2)=typaquet(u4)=REINIT et noempl(u1)=noempl(u2).

Sur chacune des deux voies la conservation de l'assertion se démontre de manière similaire à la transition T-sortie, avec typaquet(u2)=REINIT.

L'assertion T1 est donc conservée après tout franchissement d'une des transitions du modèle. Etant vraie à l'état initial, **l'assertion T1** est un **invariant** du modèle.

Théorème 2:

Le modèle ne duplique pas les paquets de données.

T2: $\forall$ M $\in$ A , $\forall$ Voie $\in$ {Voie1,Voie2} ,

$\forall$ ui $\in$ Voie(M(P-réseau)) , $\forall$ uj $\in$ Voie(M(P-réseau)) avec

typaquet(ui)=DONNEE et typaquet(uj)=DONNEE ,

si noempl(ui)$\neq$noempl(uj) alors nodonnée(ui)$\neq$nodonnée(uj).

Démonstration :

Soit U l'ensemble des uplets respectant les conditions suivantes:

u $\in$ U si u $\in$ Voie(M'(P-réseau)) et typaquet(u)=DONNEE.

Les uplets ui et uj du théorème T2 ayant un rôle parfaitement symétrique dans le libellé, il suffit d'effectuer les démonstrations sur le premier uplet.

A l'état initial, les emplacements de la place P-réseau sont vides, donc l'assertion T2 est vérifiée par défaut.

$\forall$ u $\in$ Mo(P-réseau) typaquet(u)=VIDE.

Pour le franchissement des transitions:

- **T-entrée**, le déclenchement provoque :
M(P-émetteur)'=M(P-émetteur)+1;
M(P-réseau)'=M(P-réseau)-u1+u2 avec
noempl(u1)=noempl(u2)=1 , typaquet(u2)=DONNEE ,
typaquet(u1)=VIDE et nodonnée(u2)=M(P-émetteur).

- Soit ui=u2,
  - ∀ uj ∈ U tel que typaquet(uj)=DONNEE.
    On avait avant le franchissement, d'après le lemme L1:
    nodonnée(uj)<M(P-émetteur). Or d'après les
    prédicats: nodonnée(u1)=P-émetteur>nodonnée(uj), donc
    aucun uplet ne peut vérifier la condition.
    Le théorème est vérifié par défaut.
- Soit ui ∈ U-{u2},
  - si uj=u2, nous nous replaçons dans le cas précédent.
  - si uj ∈ U-{u2}, alors
    la transition ne modifiant ni l'uplet ui, ni l'uplet uj,
    la validité du théorème T2 est conservée.

Nous venons de prouver que le théorème T2 est conservé par le franchissement de la transition T-entrée.

- **T-transit**, le déclenchement provoque :
  M(P-réseau)'=M(P-réseau)-u1-u2+u3+u4 avec
  noempl(u1)=noempl(u3) et noempl(u2)=noempl(u4)=noempl(u1)+1,
  nodonnée(u1)=nodonnée(u4) et typaquet(u2)=typaquet(u3)=VIDE.

  - Soit ui=u3, une telle égalité est impossible car
    typaquet(u3)=VIDE.
  - Soit ui=u4,
    Avant le déclenchement de la transition le théorème était
    valide pour l'uplet u1 dans M(P-réseau):
    ∀ uj tel que typaquet(uj)=DONNEE
    si noempl(uj)≠noempl(u1) alors nodonnée(uj)≠nodonnée(u1).
    D'après le lemme 4 d'unicité des emplacements,
    comme noempl(u3)=noempl(u1) et typaquet(u3)=VIDE,

comme noempl(u4)=noempl(u1)+1 et nodonnée(u1)=nodonnée(u4),
l'uplet u1 échange sa condition avec l'uplet u4. Toute
relation valide sous M(P-réseau) par l'uplet u1 est valide
sous M(P-réseau) par l'uplet u4 ,donc:

$\forall$ uj $\in$ M(P-réseau) tel que typaquet(uj)=DONNEE,
si noempl(uj)$\neq$noempl(u4) alors nodonnée(uj)$\neq$nodonnée(u4).

- Soit ui $\in$ U-{u3,u4},
alors les uplets ui et uj ne sont pas modifiés par le déclenchement de la transition T-transit. Le théorème T2 est conservé après le franchissement.

Nous venons de prouver que le franchissement de la transition T-transit conserve la validité du théorème T2 sur le modèle.

- **T-sortie**, son déclenchement provoque:
M(P-réseau)'=M(P-réseau)=u1+u2 avec
typaquet(u1)=DONNEE et typaquet(u2)=VIDE,
noempl(u1)=noempl(u2)=NBEMPL.

- Soit ui=u2 ou uj=u2, ces égalités sont impossibles car typaquet(u2)=VIDE.
- Soit ui $\in$ U-{u2} et uj $\in$ U-{u2},
alors les uplets ui et uj ne sont pas modifiés par le déclenchement de la transition T-transit.
Quelque soit ui' tel que ui(T-sortie>ui' , on a
noempl(ui)=noempl(ui') et nodonnée(ui)=nodonnée(ui').

Le théorème T2 est conservé après le franchissement de la transition T-sortie.

- **T-désynchro**, son franchissement crée:
M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec opposée(u1,u3) ,
état(u2)=état(u4)=DESYNCHRO et noempl(u1)=noempl(u2).

Le comportement des uplets étant identique pour les voies VOIE1 et VOIE2, et le théorème traitant de manière indépendante les deux voies, les preuves pour u1 et u2 se font de manière similaire sur u3,u4.

- $\forall$ ui $\in$ U,
  les uplets ne sont pas modifiés par le déclenchement
  de la transition T-transit.
  Quelque soit ui' tel que ui(T-sortie>ui' , on a
  noempl(ui)=noempl(ui') et nodonnée(ui)=nodonnée(ui').
  Le théorème T2 est conservé après le franchissement de la transition T-désynchro.

- **T-perte**, qui provoque à son franchissement:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  typaquet(u2)=VIDE et noempl(u1)=noempl(u2).

  - Soit ui=u2, l'égalité est impossible car typaquet(u2)=VIDE.
  - Soit ui $\in$ U-{u2} et uj $\in$ U-{u2},
    alors les uplets ui et uj ne sont pas modifiés par le déclenchement
    de la transition T-transit.
    Quelque soit ui' tel que ui(T-sortie>ui' , on a
    noempl(ui)=noempl(ui') et nodonnée(ui)=nodonnée(ui').
    Le théorème T2 est conservé après le franchissement de la transition T-perte.

- **T-resynchro**, qui provoque:
  M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec opposée(u1,u3) ,
typaquet(u2)=typaquet(u4)=REINIT et noempl(u1)=noempl(u2).

Le comportement des uplets étant identique pour les voies VOIE1 et VOIE2, et le théorème traitant de manière indépendante les deux voies, les preuves pour u1 et u2 se font de manière similaire sur u3, u4.

- Soit ui=u2, c'est impossible car typaquet(u2)=REINIT.

- Soit ui ∈ U-{u2} et uj ∈ U-{u2},
  alors les uplets ui et uj ne sont pas modifiés par le déclenchement de la transition T-transit.
  Quelque soit ui' tel que ui(T-sortie>ui' , on a
  noempl(ui)=noempl(ui') et nodonnée(ui)=nodonnée(ui').

Le théorème T2 est conservé après le franchissement de la transition T-resynchro.

L'assertion T2 est donc conservée après tout franchissement d'une des transitions du modèle. Etant vrai à l'état initial, **l'assertion T2** est prouvée être un **invariant** du modèle.

Théorème 3:

Toute désynchronisation est correctement détectée.

T3: ∀ M ∈ A , ∀ Voie ∈ {Voie1,Voie2} ,

∀ ui ∈ Voie(M(P-réseau)) avec typaquet(ui)=DONNEE,

si ∃ uj ∈ Voie(M(P-réseau)) tel que

Suivant(ui,Voie(M(P-réseau)))=uj ,

typaquet(uj)=DONNEE et nodonnée(ui)≠nodonnée(uj)-1 alors :

- T3.1: soit ∃ uk ∈ Voie(M(P-réseau)) tel que
  typaquet(uk)=REINIT et noempl(uk)<noempl(ui);
- T3.2: soit ∃ uk ∈ Voie(M(P-réseau)) tel que
  état(uk)=DESYNCHRO et noempl(uk)<noempl(ui);

Démonstration :

Soit Ui l'ensemble des uplets respectant les conditions suivantes:

u ∈ Ui si u ∈ Voie(M'(P-réseau)) et typaquet(u)=DONNEE.

Soit Uj l'ensemble des uplets respectant les conditions suivantes:

u ∈ Uj si u ∈ Voie(M'(P-réseau)) tel que typaquet(u)=DONNEE,

Suivant(ui,Voie(M(P-réseau)))=uj et nodonnée(ui)≠nodonnée(uj)-1.

A **l'état initial**, la place P-réseau ne contient que des marques dont le type de paquet est vide. Donc l'assertion T3 est vérifiée par défaut.

Pour l'ensemble des transitions du modèle :

- **T-entrée**, le franchissement provoque:

M(P-réseau)'=M(P-réseau)-u1+u2 avec,
noempl(u1)=noempl(u2)=1 , typaquet(u2)=DONNEE ,
typaquet(u1)=VIDE et nodonnée(u2)=M(P-émetteur).

- soit ui ∈ {u2},
  il n'existe pas d'uplet uj qui suit l'uplet u2, car le numéro d'emplacement de l'uplet u2 est égal a1 (application de lemme L4), /∃ uk ∈ M(P-réseau)' tel que noempl(uk)<noempl(u2)=1 .

- soit ui ∈ Ui-{u2},
  - soit uj=u2, c'est-à-dire Suivant(ui,M(P-réseau)')=u2,
    Avant le franchissement
    de la transition T-entrée, d'après le Lemme 4 d'unicité des uplets et la construction de la fonction Dernier, on avait ui=Dernier(M(P-réseau)), donc l'uplet ui vérifiait le Lemme 5.
    Il existait un uplet uk vérifiant les parties optionnelles L5.1 ou L5.2 du Lemme 5.
    Après le franchissement de la transition, l'uplet uk est inchangé :
    - uk ne peut être égal à u1 car typaquet(u1)=VIDE, d'après le prédicat de franchissement.
      Comme uk≠u1, le franchissement de la transition T-entrée conserve pour l'assertion T3 la véracité des parties optionnelles du Lemme 5.

- Soit uj ∈ Uj-{u2},
  les uplets ui et uj (éléments de P-réseau avant le franchissement) vérifiaient l'assertion T3 :
  Il existe l'uplet uk vérifiant les parties T3.1 ou T3.2, alors
  - soit uk≠u1, alors T3 est vérifié trivialement, par conservation après le franchissement.
  - soit uk=u1 avant franchissement, alors après le

franchissement :
typaquet(u1)=typaquet(u2) et état(u1)=état(u2),
l'uplet u2 se substitue à l'uplet u1 pour M(P-réseau)',
∃ uk'=u2 qui vérifie soit T3.1 soit T3.2,
donc l'assertion T3 est vérifiée.

Nous venons de prouver la conservation de l'assertion après le franchissement de la transition T-entrée.

- **T-transit**, nous avons le changement d'état suivant:
M(P-réseau)'=M(P-réseau)-u1-u2+u3+u4 avec
noempl(u1)=noempl(u3) et noempl(u2)=noempl(u4)=noempl(u1)+1,
typaquet(u1)=typaquet(u4)=VIDE et typaquet(u2)=typaquet(u3)=DONNEE.

L'uplet u3 n'intervient pas dans la vérification de l'assertion T3,
car typaquet(u3)=VIDE.

- soit ui=u4,
  - soit uj ∈ Uj, donc Suivant(u4,M(P-réseau)')=uj et
    nodonnée(u4)≠nodonnée(uj)-1.

    Or d'après les conditions de franchissement, on a :
    noempl(u3)=noempl(u1), noempl(u4)=noempl(u1)+1 et
    typaquet(u3)=VIDE , nodonnée(u4)=nodonnée(u1).
    D'après le Lemme L4, chaque emplacement est identifié de manière unique, et comme la fonction Suivant ne porte que sur des paquets non-vides, nous déduisons qu'avant le franchissement, on avait : Suivant(u4,M(P-réseau))=uj.

    Nous en obtenons la conservation des deux parties optionnelles T3.1 et T3.2, car l'existence de l'uplet uk persiste:

- l'uplet uk est inchangé par le franchissement de la transition,
- l'uplet u4 a le même numéro de donnée que l'uplet u1,
- le type de l'uplet u3 est vide.

- ∀ ui ∈ Ui-{u4,u3},
- soit uj=u4,

par un raisonnement symétrique au précédent, on obtient
si Suivant(ui,M(P-réseau))=u1 alors Suivant(ui,M(P-réseau)')=u4.
De même, la conservation de la partie optionnelle T3.1 et T3.2
est réalisée par la persistance de l'uplet uk :

- la transition laisse inchangé l'uplet,
- l'uplet u4 a le même numéro de donnée que l'uplet u3 (donc lui est substitué),
- l'uplet u4 contient un paquet vide.

- soit uj ∈ Uj-{u4},

les uplets ui et uj sont inchangés, car ils ne participent pas au franchissement de la transition.
L'assertion s'appliquait avant le franchissement de la transition T-transit, donc il existait un uplet uk respectant T3.

- soit uk=u1 avant le franchissement, alors comme nodonnée(u1)=nodonnée(u4), l'uplet u4 vérifie T3 après le franchissement.
- l'uplet uk ne peut être u3, car typaquet(u3)=VIDE.
- pour uk ∈ M(P-réseau)' et uk≠(u3 ou u2),

les uplets ui, uj et uk sont inchangés par la transition,
l'assertion se conserve aisément.

Nous venons de démontrer la validité de l'assertion après le franchissement de la transition T-transit.

- **T-sortie**, qui modifie le modèle ainsi :

M(P-réseau)'=M(P-réseau)-u1+u2 avec
typaquet(u1)=DONNEE , typaquet(u2)=VIDE et,
noempl(u1)=noempl(u2)= NBEMPL.

- ui=u2 est impossible, car /∃ uj ∈ M(P-réseau)' tel que
  Suivant(u2,M(P-réseau)')=uj, parce que typaquet(u2)=VIDE.

- ∀ ui ∈ Ui-{u2} tel que Suivant(ui,M(P-réseau)')=uj,
  - si uj ∈ Uj-{u2}, les uplets ui, uj étant éléments de la place P-réseau avant le franchissement, ils vérifiaient l'assertion T3. Ils ne sont pas modifiés par le franchissement de la transition, donc s'ils vérifiaient une des propositions T3.1 ou T3.2, il existait un uplet uk qui est différent de u1, car noempl(u1)= NBEMPL. Cet uplet uk est inchangé par le franchissement de la transition, donc il vérifie toujours l'une des deux propositions T3.1 ou T3.2.
  - uj=u2 est impossible, car /∃ uj ∈ M(P-réseau)' tel que Suivant(u2,M(P-réseau)')=uj car typaquet(u2)=VIDE.

Nous montrons donc, que la transition T-sortie conserve l'assertion T3.

- **T-désynchro**, alors :
  M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec opposée(u1,u3) ,
  état(u2)=état(u3)=DESYNCHRO et noempl(u1)=noempl(u2).

Comme le théorème T3 traite les voies séparément, la preuve peut être établie sur une voie (u1,u2), puis étendue à l'autre (u3,u4).

- Soit ui=u2,

- soit uj ∈ Uj tel que Suivant(u2,M(P-réseau)')=uj et
  nodonnée(u2)≠nodonnée(uj)-1.
  D'après les conditions de franchissement:
  noempl(u1)=noempl(u2) et typaquet(u1)=typaquet(u2).
  D'après le lemme L4, il y a unicité des emplacements, on obtient:
  Suivant(u2,M(P-réseau)')=Suivant(u1,M(P-réseau))=uj.
  Par hypothèse, on avait avant le franchissement de la transition,
  pour u1 l'assertion T3:
  - soit une des propositions T3.1 ou T3.2. L'uplet uk les vérifiant
    ne pouvant être l'uplet u1, car noempl(u1)=noempl(u2)>noempl(uk).
    Le franchissement de la transition conserve l'assertion T3.

- ∀ ui ∈ Ui-{u2},
  - ∀ uj ∈ Uj tel que Suivant(ui,M(P-réseau)')=uj,
    et nodonnée(uj)≠nodonnée(ui)-1 :
    - ∀ ui tel que noempl(ui)>noempl(u2), il existe un uplet
      vérifiant la proposition T3.2. Cet uplet est u2, car
      état(u2)=DESYNCHRO.
    - ∀ ui tel que noempl(ui)<noempl(u1), avant le franchissement
      de la transition s'il existait un uplet uk, il est
      conservé car noempl(u1)=noempl(u2)>noempl(ui)>noempl(uk),
      et d'après le lemme d'unicité des emplacements.
      Cet uplet uk rend valide, une des deux propositions, comme
      avant le franchissement.

Nous venons de vérifier la conservation du théorème 3 durant le franchissement de la transition T-désynchro.

- **T-perte**, son franchissement provoque:
  M(P-réseau)'=M(P-réseau)-u1+u2 avec
  typaquet(u2)=VIDE et noempl(u1)=noempl(u2).

- ui=u2 est impossible, car typaquet(u2)=VIDE.

- ∀ ui ∈ Ui-{u2},
  - uj=u2 est impossible, car typaquet(u2)=VIDE.
  - ∀ ui ∈ Uj-{u2}, alors Suivant(ui,M(P-réseau)')=uj et nodonnée(ui)≠nodonnée(uj)-1:
  - ∀ ui tel que noempl(ui)>noempl(u2), il existe un uplet qui valide la proposition T3.2. Cet uplet est u2, car état(u2)=DESYNCHRO.
  - ∀ ui tel que noempl(ui)<noempl(u2), s'il existait un uplet uk avant le franchissement qui vérifie l'une des deux propositions T3.1 ou T3.2, alors il est conservé par la transition T-perte car:
    - d'après le lemme 4 d'unicité des emplacements,
    - noempl(u1)=noempl(u2)>noempl(ui)>noempl(uk).

    Comme uk≠u1, il peut vérifier toujours le théorème 3.

Nous venons de prouver que le théorème 3 est conservé après le franchissement de la transition T-perte.

- **T-resynchro**, cette transition provoque les événements suivants:
  M(P-réseau)'=M(P-réseau)-u1+u2 -u3+u4 avec opposée(u1,u3) ,
  noempl(u1)=noempl(u2), noempl(u3)=noempl(u4),
  état(u2)=état(u4)=SYNCHRO, typaquet(u2)=typaquet(u4)=REINIT.

Du fait l'indépendance des voies pour le théorème T3, les preuves sur les uplets u1 et u2 peuvent être appliquées aux uplets u3 et u4.

- ui=u2 est impossible, car typaquet(u2)=REINIT.
- ∀ ui ∈ Ui-{u2},

- uj=u2 est impossible, car typaquet(u2)=REINIT.
- ∀ ui ∈ Uj-{u2}, alors Suivant(ui,M(P-réseau)')=uj et nodonnée(ui)≠nodonnée(uj)-1:
  - ∀ ui tel que noempl(ui)>noempl(u2), il existe un uplet qui valide la proposition T3.1. Cet uplet est u2, car typaquet(u2)=REINIT.
  - ∀ ui tel que noempl(ui)<noempl(u2), s'il existait un uplet uk avant le franchissement qui vérifie l'une des deux propositions T3.1 ou T3.2, alors il est conservé par la transition T-perte car:
    - d'après le lemme 4 d'unicité des emplacements,
    - noempl(u1)=noempl(u2)>noempl(ui)>noempl(uk).

    Comme uk≠u1, il peut vérifier toujours le théorème 3 après le franchissement.

Nous venons de prouver que le théorème 3 est conservé après le franchissement de la transition T-resynchro.

L'assertion T3 est donc conservée après tout franchissement d'une transition du modèle. Etant vrai à l'état initial, nous venons de prouver que l'assertion **T3** est un **invariant** du modèle.

Théorème 4 :

Le modèle est vivant.

T4: $\forall M \in A, \forall t \in T, \exists S \in T^*$ tel que $M(St>$.

Démonstration :

Nous prouvons la vivacité du modèle à l'aide de deux lemmes: le modèle possède un état d'accueil E (Lemme 10), le modèle est quasi-vivant à partir de l'état d'accueil E (Lemme 11); et de la propriété des réseaux de Petri qui démontre que si un modèle possède un état d'accueil et s'il est quasi-vivant à partir de cet état, alors il est vivant.

Lemme 10 : Le modèle possède un ensemble **E d'états d'accueil.**

Nous définissons l'ensemble E par le marquage des deux places du modèle tel que :

- Les uplets de la place P-réseau sont vides de contenu, leur état synchrone (sans erreur):

$\forall M \in E, \forall u \in M$(P-réseau), état(u)=SYNCHRO , typaquet(u)=VIDE.

- Le marquage de la place P-émetteur peut être quelconque.

On note que l'état initial fait partie de l'ensemble d'accueil.

Pour prouver que E est un ensemble d'accueil, il faut montrer qu'à partir d'un état quelconque de l'ensemble A des états accessibles, il est toujours possible d'atteindre un des états de l'ensemble d'accueil.

Nous allons définir une norme W qui nous permettra de faire la preuve que cet état d'accueil existe [Keller 76]. La norme W est définie comme suit :

$\forall u \in$ P-réseau , W(u) = (rang(u).(état(u)+type(u))) avec

la définition des fonctions numériques suivantes :

rang(u) = 1/noempl(u) ;
état(u) = si état(u)==SYNCHRO alors 0 sinon 4 ;
type(u) = si typaquet(u)==DONNEE alors 1
si typaquet(u)==REINIT alors 2
sinon 0 ;

Nous vérifions d'abord que la norme W appliquée à un marquage M quelconque de l'ensemble E d'accueil est bien nulle :

$\forall u \in$ M(P-Réseau) état(u)=SYNCHRO et typaquet(u)=VIDE, donc W(u)=0.

Puis nous vérifions que de tout marquage M, il est possible de faire décroître la norme W. Cette démonstration est effectuée en deux étapes:

Etape 1 : Il est possible de resynchroniser (si nécessaire) les emplacements désynchronisés de la place P-réseau (l'ensemble des emplacements devient synchrone).

Nous supprimons les emplacements en état de désynchronisation (on veut que $\forall$ $u \in$ M(P-réseau) état(u)=SYNCHRO).

Tant qu'il existe des emplacements en état de désynchronisation (c.a.d $\exists$ $u \in$ M(P-réseau) tel que état(u)=DESYNCHRO):

Soit d1 un emplacement en état de désynchronisation ($d1 \in$ M(P-réseau) tel que état(d1)=DESYNCHRO}). D'après le lemme L6, les deux emplacements d1 et d2, de numéro opposé (appartenant à la même station de transport) sont dans le même état de désynchronisation (opposée(d1,d2) et état(d1)=état(d2)). La transition T-resynchro est donc franchissable, ce qui produit sur chacune des deux voies de transmission un paquet de réinitialisation

($\exists$ r1 $\in$ Voie1(M(P-réseau)) et $\exists$ r2 $\in$ Voie2(M(P-réseau)) tel que typaquet(r1)=typaquet(r2)=REINIT (lemme L3)).
Le réseau comporte alors un emplacement de moins en état de désynchronisation.

Et ce, jusqu'à disparition totale de tous les emplacements en état de désynchronisation.

La transition T-resynchro fait décroître la norme W :

On vient de voir qu'il faut, dans un premier, temps franchir la transition T-resynchro. Il faut donc prouver que la norme décroît au franchissement de la transition T-resynchro.

Cette transition modifie le marquage de la façon suivante:
M'(P-réseau) = M(P-réseau) - (u1+u2) + (u3+u4) avec
noempl(u1)=noempl(u3)=n , noempl(u2)=noempl(u4)=opposé(n) ,
état(u1)=état(u2)=DESYNCHRO , état(u3)=état(u4)=SYNCHRO ,
typaquet(u1)=p1 , typaquet(p2)=p2 , et
typaquet(u3)=typaquet(u4)=REINIT.

La norme W se calcule comme suit :
W(u1) = (4+p1)/n , W(u2) = (4+p2)/opposé(n) et
W(u3) = (0+2)/n , W(u4) = (0+2)/opposé(n) .

Donc W(u1)+W(u2) > W(u3)+W(u4) d'où
W(M(P-réseau))+W(u1)+W(u2) > W(u3)+W(u4) et
W(M(P-réseau) > W(M'(P-réseau) , preuve que la transition
T-resynchro fait bien décroître la norme W.

Etape 2 : Il est possible de délivrer aux récepteurs (s'ils existent) l'ensemble des paquets présents dans la place P-réseau (l'ensemble des emplacements devient vide);

Nous vidons les emplacements de la place P-réseau.(c.a.d on veut que ∀ u on ait typaquet(u)=VIDE). On effectue les actions suivantes pour la VOIE1, puis pour la VOIE2. Cela est licite, car toutes les transitions que nous avons à franchir ne mettent en cause qu'une seule voie de transmission.

Tant que la place P-réseau contient encore des paquets (c.a.d tant que ∃ u ∈ M(P-réseau) tel que typaquet(u)≠VIDE) :

Soit p l'uplet contenant le paquet le plus proche du récepteur (c.a.d p=Premier(M(P-réseau))), tant que l'uplet p n'est pas en bout de file (côté récepteur) (c.a.d tant que noempl(p)≠NBEMPL), on fait progresser l'uplet p dans le réseau. C'est possible, car par construction l'uplet p étant le premier paquet encore contenu par le réseau, les emplacements précédents sont vides. Donc la transition T-transit est franchissable et provoque l' échange entre p et l'emplacement vide le précédant. Le paquet p progresse d'un emplacement, et il conserve son statut de premier paquet (d'après le libellé des arcs et prédicats de la transition) :

noempl(p)=noempl(p)+1).

Le premier paquet p arrive, donc, en bout de réseau (c.a.d p=premier(M(P-réseau)), typaquet(p)=paquet et noempl(p)=NBEMPL):
la transition T-sortie devient franchissable, ses prédicats sont vérifiés. Le réseau contient un paquet de moins.

Et ce, jusqu'à délivrance de tous les paquets contenus par le réseau.

La transition T-transit fait décroître la norme W :

On vient de voir qu'il faut, dans un premier temps, franchir la transition T-transit associée à l'uplet le plus proche du récepteur. Il faut donc prouver que la norme décroît au franchissement de la transition T-transit.

Cette transition modifie le marquage de la façon suivante:
M'(P-réseau) = M(P-réseau) - (u1+u2) + (u3+u4) avec
noempl(u1)=noempl(u3)=n , noempl(u2)=noempl(u4)=n+1 ,
typaquet(u4)=typaquet(u1)=PAQUET=p , typaquet(u2)=typaquet(u3)=VIDE ,
et état(u1)=état(u3)=s1 , état(u2)=état(u4)=s2 .

La norme W se calcule comme suit :
W(u1) = (s1+p)/n , W(u2) = (s2+0)/(n+1) et
W(u3) = (s1+0)/n , W(u4) = (s2+p)/(n+1) .

Donc W(u1)+W(u2) > W(u3)+W(u4), d'où
W(M(P-réseau) > W(M'(P-réseau) ,
preuve que la transition T-transit fait bien décroître la norme.

La transition T-sortie fait décroître la norme W :

Une fois l'uplet arrivé au dernier emplacement, il franchit la transition T-sortie.

Cette transition modifie le marquage de la façon suivante :
M'(P-réseau) = M(P-réseau) - u1 + u2 avec
noempl(u1)=noempl(u2)=NBEMPL et état(u1)=état(u2)=s ,
typaquet(u1)=PAQUET=p , typaquet(u2)=VIDE.

La norme W se calcule comme suit :
W(u1) = (s+p)/NBEMPL , W(u2) = (s)/NBEMPL .

Donc W(u1) > W(u2)
W(M(P-réseau) + W(u1) > W(M(P-réseau) + W(u2)
W(M(P-réseau) > W(M(P-réseau) - W(u1) + W(u2)
W(M(P-réseau) > W(M'(P-réseau) ,
ce qui prouve que le franchissement de la transition T-sortie fait bien décroître la norme.

Ceci nous permet de montrer que l'ensemble E est bien un ensemble d'états d'accueil et que **le lemme L10 est vrai.**

Lemme 11 : Dans un état quelconque de l'ensemble E d'accueil, le modèle est quasi-vivant.

Pour prouver cette propriété, nous devons de franchir successivement toutes les transitions du modèle:

Dans un état quelconque de l'ensemble des états d'accueil, la place P-réseau ne contient, que des emplacements vides. La transition **T-entrée** est franchissable et provoque l'apparition d'un paquet de données dans l'emplacement numéro 1.

La transition **T-désynchro** est toujours franchissable. Nous la déclenchons sur l'emplacement 1, qui devient désynchronisé.

De ce fait, la transition **T-perte** peut être franchie pour l'emplacement 1. Le paquet de données est donc perdu.

Nous décidons de franchir la transition **T-resynchro**, ce qui provoque l'apparition d'un paquet de réinitialisation sur chacune des deux voies de transmission.

La transition **T-transit** permet de les propager à travers le réseau jusqu'à leur entité destinatrice.

La transition **T-sortie** devient alors franchissable.

Le **lemme L11 est donc vérifié.**

Nous venons donc de prouver que : premièrement notre modèle possède un état d'accueil E (Lemme 10), et deuxièmement que le modèle est quasi-vivant à partir de l'état E (Lemme 11). Nous en déduisons que notre modèle du service de la couche Réseau est vivant et donc que le **théorème T4 est vérifié.**

4.3 La concordance du modèle

Nous allons montrer que l'ensemble des théorèmes joints aux lemmes prouvés ci-dessus, induisent les quatre propriétés définies par la spécification du service Réseau. Cette dernière étape constitue la preuve de la concordance du modèle.

Les théorèmes ayant été conçus dans le but de prouver que le modèle possède bien l'ensemble des propriétés caractérisant le service de la couche Réseau, le rapprochement des propriétés et des théorèmes est aisé.

La **propriété R-1** est issue de la preuve du théorème T1. Le théorème T1 prouve que le modèle conserve la séquentialité des paquets, donc tous les paquets émis dans un certain ordre seront reçus dans le même ordre (R-1).

La **propriété R-2** est apportée par la preuve du théorème T2. Le théorème T2 prouve que le modèle ne duplique pas les paquets, donc tous paquets émis s'ils sont reçus ne le seront qu'une et une seule fois (R-2).

La **propriété R-3** caractérise le comportement du service Réseau face aux désynchronisations. Nous prouvons, à l'aide des lemmes 2, 3 et des trois derniers théorèmes que le modèle est conforme à sa spécification. Nous savons que toute phase de désynchronisation est concrétisée par la perte de paquets circulant sur le Réseau. D'après le Lemme 2, la transition T-perte n'est franchissable que s'il existe un emplacement en état de désynchronisation. D'après le Lemme 3, la phase de resynchronisation (transition T-Resynchro) insère un paquet de réinitialisation dans chacune des voies de communication. Le théorème T3 confirme que ces insertions sont effectuées après toute désynchronisation. L'ordre relatif des paquets est conservé jusqu'à leur délivrance à l'extrémité réceptrice (théorème T1), et ce, sans duplication (théorème T2).

La **propriété R-4** est directement obtenue par la démonstration du théorème T4 sur la vivacité du modèle.

Nous venons de prouver que le modèle possède les propriétés caractérisant le service Réseau (Concordance de modèle). Ce qui met un point final à la construction du modèle du service de la couche Réseau conformément à notre méthodologie.

Ces propriétés ont été établies en utilisant les invariants issus du modèle. La preuve d'invariance des assertions a été grandement facilitée par la puissance d'expression des Réseaux de Petri à prédicats, qui a considérablement réduit le nombre de transitions du modèle, qui cependant intègre un large ensemble de fonctionnalités du Service de la couche Réseau en phase transfert de données.

5. CONCLUSION

Nous avons proposé et prouvé, dans cette partie, un modèle du service de la couche Réseau en phase de transfert de données. Ce modèle, bien que concis, possède l'ensemble des propriétés essentielles du service (Figure B-3.1).

Nous avons appliqué en quatre étapes notre méthodologie pour établir ce modèle. La première étape a permis de définir le système que nous allions étudier : le service de la couche Réseau en mode connexion durant la phase de transfert de données, tel qu'il est spécifié dans les normes internationales sur les télécommunications.

La deuxième étape a précisé notre champ d'investigation en définissant les quatre propriétés caractérisant le service. Ces propriétés sont énumérées rapidement, la conservation de la séquentialité des paquets (R-1), leur non-duplication (R-2), un bon comportement en phase de resynchronisation (R-3), et le non-blocage du service Réseau (R-4).

La modélisation a constitué la troisième étape et a permis, ainsi, de construire le modèle à l'aide des Réseaux de Petri à prédicats. Ce modèle d'un graphisme relativement simple, grâce à la puissance d'expression des Réseaux de Petri à prédicats, comporte néanmoins l'ensemble des propriétés définies à l'étape précédente. C'est ce que prouve l'étape suivante.

La dernière étape extrait du modèle un ensemble de propriétés, qui permettent, en les rapprochant des propriétés du système précisées à la deuxième étape, d'établir la concordance du modèle, but ultime de notre méthode.

S'il est évident que tout le Service de la couche Réseau n'est pas complètement représenté par notre modèle (nous nous sommes restreint à la phase de transfert de données), notre méthode, par l'application de la concordance de modèle, nous oblige à définir précisément et à exhiber les propriétés de la description du système.

L'étude du Service de la couche Réseau nous a confronté a un problème typiquement asynchrone : les désynchronisations du circuit virtuel. Nous en donnons un modèle concis et validé.

Un tel modèle peut désormais être intégré à la modélisation du protocole de la couche Transport. Les assertions relatives au modèle de la couche Réseau sont conservées dans le modèle de la couche Transport. Elles pourront être utilisées dans la démonstration des propriétés de la couche Transport.

C - TROISIEME PARTIE

Le PROTOCOLE TRANSPORT

Son Modèle

Sa Validation

PLAN
de la Troisième Partie

1.INTRODUCTION

Nous allons modéliser le fonctionnement du protocole de télécommunication appelé Transport, défini par la couche quatre de la norme internationale pour l'interconnexion des systèmes ouverts [ECMA 72].

De nombreux autres protocoles de communication ont déjà été modélisés, cependant ils intégraient généralement des fonctionnalités assez simples (bit alterné [Girault 81], HDLC [Berthelot 81b]). Nous allons nous intéresser principalement au mécanisme typique de cette couche : le crédit variable et réductible. Ce mécanisme permet de gérer le contrôle de flux des messages de manière très souple, il autorise la réquisition de crédit déjà attribué.

Un tel phénomène peut facilement occasionner un comportement incohérent si l'on y prend garde.

Le mécanisme du contrôle de flux n'entre en jeu qu'en phase de transfert de donnée, c'est pourquoi nous y limiterons notre étude. Les autres phases de connexion et de déconnexion peuvent être considérées comme indépendantes. De plus, elles présentent moins de difficultés donc moins d'intérêt, et ont déjà fait l'objet d'étude [Berthelot 81b].

Toutefois, toute modélisation de processus nécessite une intégration préalable de son milieu d'exécution. Dans le cas d'une modélisation de protocole de communication, le milieu est représenté par le protocole de la couche sous-jacente. Du fait de la décomposition en couches des protocoles de télé-communication, une couche donnée n'a de relation directe qu'avec les couches immédiatement supérieure et immédiatement inférieure. Dans la partie précédente, nous avons eu l'occasion de spécifier, de modéliser et de prouver les propriétés de la couche Réseau.

Ce sous-modèle a le comportement suivant:

- Tout message lui étant transmis, après un délai de transmission, est, soit délivré à l'entité réceptrice, soit perdu.
- Toute perte de paquet pendant une désynchronisation du réseau provoque ultérieurement après sa resynchronisation l'apparition de signaux de réinitialisations vers les entités communiquantes.

En accord avec notre méthode, nous allons dans un premier temps décrire le protocole de la couche Transport, puis l'analyser afin d'en tirer les propriétés caractéristiques. Notre description du protocole de Transport, faisant suite à la description plus générale de la première partie de présentation de notre thèse, se focalise sur les problèmes essentiels de la phase de transfert des données de la couche Transport et notamment la gestion de la fenêtre à crédit réductible.

Nous allons, ensuite, construire un modèle du protocole de Transport en phase de transfert de données, en lui adjoignant au fur à mesure l'ensemble de ses fonctionnalités. Ce modèle utilise les réseaux de Petri à prédicats, qui ont déjà montré leurs nombreux avantages dans la partie précédente. Ce modèle est nettement plus complexe que celui du service Réseau, car il intègre un ensemble de fonctionnalités jamais encore modélisées.

Enfin, nous validons le modèle final, d'abord pour établir la concordance du modèle, puis l'adéquation du service. Pour ce faire, nous analysons le modèle pour obtenir les propriétés qui permettent de prouver la concordance, puis celles qui permettent de prouver le service. Ces deux ensembles de propriétés représentent, le premier les propriétés attachées aux fonctionnalités internes du protocole, le deuxième les propriétés attachées aux fonctionnalités externes du service.

2. Le PROTOCOLE de la COUCHE TRANSPORT

2.1 L'Etude

Nous voulons donc modéliser le protocole de la couche Transport, mais avant tout, en accord avec notre méthode, il va falloir définir le champ de notre étude.

La première partie nous a permis de situer le protocole de Transport parmi ceux de la norme OSI, et d'évaluer ses fonctionnalités. Nous nous apercevons, qu'en fait, la phase de transfert est l'état prépondérant d'un protocole. C'est aussi l'état où la gestion du protocole est la plus critique. C'est pourquoi nous avons choisi de modéliser uniquement la phase de transfert de données (état connecté) du protocole de la couche Transport.

Notre choix s'explique ainsi:

- La modélisation et la preuve de la partie de protocole traitant de la phase de connexion ayant déjà été traitées dans toutes leurs spécificités [Berthelot 81b].
- La phase de déconnexion (état en cours de déconnexion) étant simple, elle nous semble peu intéressante.
- L'état déconnecté n'appartenant pas en propre au protocole (aucune action à effectuer).
- Seule l'étude de la phase de transfert du protocole Transport dans toutes ses particularités n'a jamais été faite et offre des difficultés techniques intéressantes.

Comme notre étude se limite à la phase de transfert de donnée, le protocole de la couche Transport suppose que la connexion est déjà établie et qu'elle le restera. C'est-à-dire que nous ne traitons pas, ici, des phases d'établissement, ni de rupture de connexion. De même, cette connexion étant basée sur une connexion Réseau, cette dernière se doit d'exister. C'est pourquoi nous introduisons dans notre modèle du protocole de Transport un sous-modèle du service Réseau. Enfin, l'interface du protocole Transport avec la couche Session se limite aux passages des messages à transmettre.

2.2 Le modèle du service Réseau

Nous avons inclus dans notre modèle du protocole de la couche Transport un modèle du service fonctionnel de la couche Réseau, en accord avec notre méthode et le modèle de référence de l'OSI : tout protocole repose sur le service de la couche inférieure. Nous avons étudié ce modèle au cours de la partie précédente. En suivant notre méthode, nous avons prouvé la concordance du modèle avec la description du service de la couche Réseau, service décrit par les quatre propriétés suivantes :

- R-1 : Il ne désordonne pas les paquets durant le transfert.
- R-2 : Il ne duplique pas les paquets durant le transfert.
- R-3 : Toute perte de paquet est détectée et les entités supérieures en sont prévenues.
- R-4 : Le service de la couche Réseau n'est jamais en situation de blocage irrémédiable.

Ce qui nous permet de l'intégrer à notre modèle, et de se servir de ses propriétés pour prouver l'ensemble de notre prochain modèle.

Nous devons, toutefois, nous assurer préalablement à l'insertion du modèle réseau, qu'il puisse effectivement conserver ses propriétés.

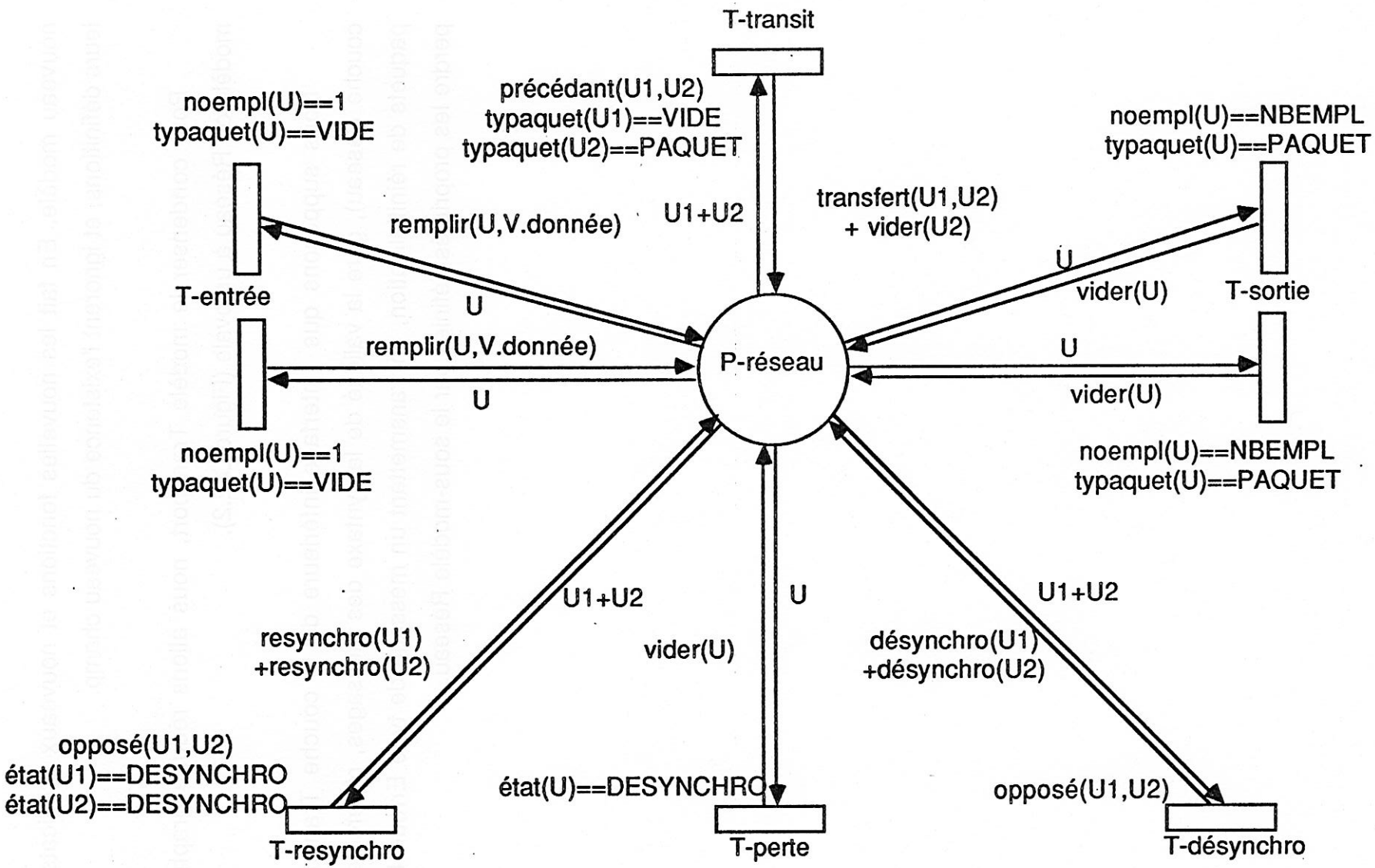
Le résultat d'équivalence de comportement, que nous avons démontré sur l'abstraction des réseaux de Petri à prédicats, nous permet d'affirmer que le modèle Réseau conserve ses propriétés, quelque soit le type de message de la couche Transport qu'il véhicule.

Le modèle de la partie précédente (Figure B-3.1) se transforme en un nouveau modèle (Figure C-2.1) qui conserve le graphe bi-partie et où les uplets possèdent un champ supplémentaire qui contient les T-PDU du protocole de la couche Transport. L'ensemble des fonctions et prédicats définis sur le modèle initial sont étendus au

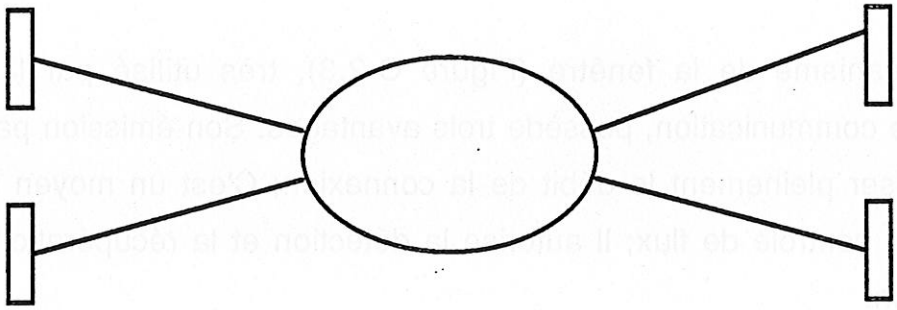
nouveau modèle. En fait les nouvelles fonctions et nouveaux prédicats conservent leurs définitions, et ignorent l'existence du nouveau champ.

Pour condenser le modèle Transport, nous allons réduire graphiquement le modèle du Réseau à un ovale (Figure C-2.2).

Nous supposons que l'interface inférieure de la couche Transport (avec la couche Réseau), gère la validité de la syntaxe des messages, répercute l'arrivée de paquets de réinitialisation, en transmettant un message de type Erreur. Et cela, sans perdre les propriétés définies par le sous-modèle Réseau.



- Figure C-2.1 - Le sous-modèle du Service Réseau



- Figure C-2.2- Abréviation du sous-modèle réseau-

2.3 La FENETRE

Le mécanisme de la fenêtre (Figure C-2.3), très utilisé par la plupart des protocoles de communication, possède trois avantages: Son émission par anticipation permet d'utiliser pleinement le débit de la connexion; C'est un moyen sûr et simple d'effectuer un contrôle de flux; Il autorise la détection et la récupération d'erreur par retransmission.

Les messages sont transmis en les numérotant à partir d'un compteur de l'émetteur s'incrémentant modulo "N". Ce numéro devient l'identité du message, qui permet aux entités Transport de le repérer de manière unique (tant que la largeur maximale de la fenêtre ne dépasse pas la moitié de N) [Steining 77].

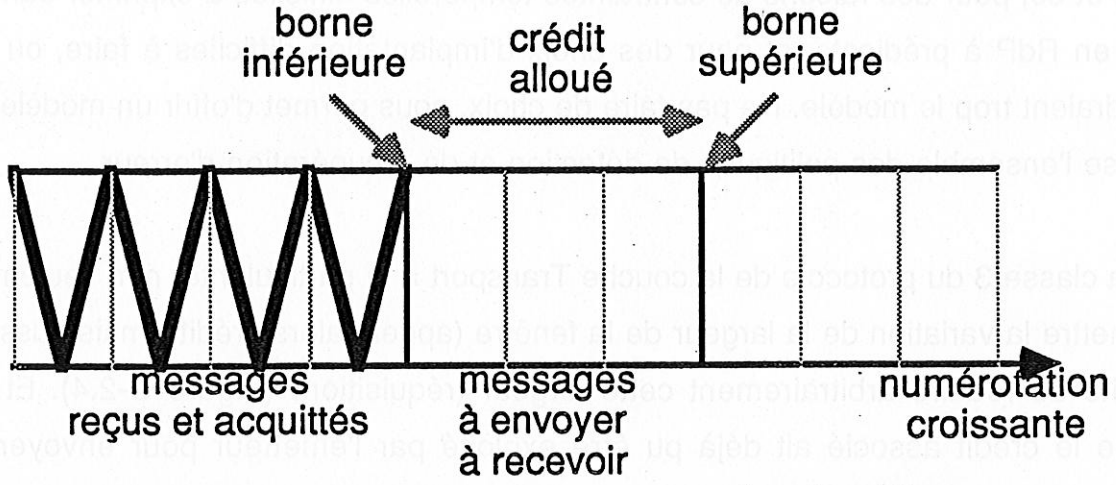
Le récepteur acquitte les messages reçus. L'émetteur est autorisé à envoyer "f" messages (f étant appelé largeur de la fenêtre) par anticipation à partir du numéro du dernier message acquitté. Le récepteur peut à tout moment émettre un message d'acquittement comportant le numéro du prochain message attendu. Cet acquittement prouve qu'il a reçu tous les messages de numéro strictement inférieur.

A la réception un mécanisme de détection d'erreur vérifie la validité des messages reçus (leur syntaxe interne, leur adéquation avec l'état du protocole du récepteur, l'absence de corruption du contenu du message, etc...). Dans le cas où la détection s'avère positive un rejet pur et simple du message s'opère (parfois, dans le cas où il y a désynchronisation entre le récepteur et l'émetteur, le protocole émet un message de rejet permettant une resynchronisation des deux partenaires).

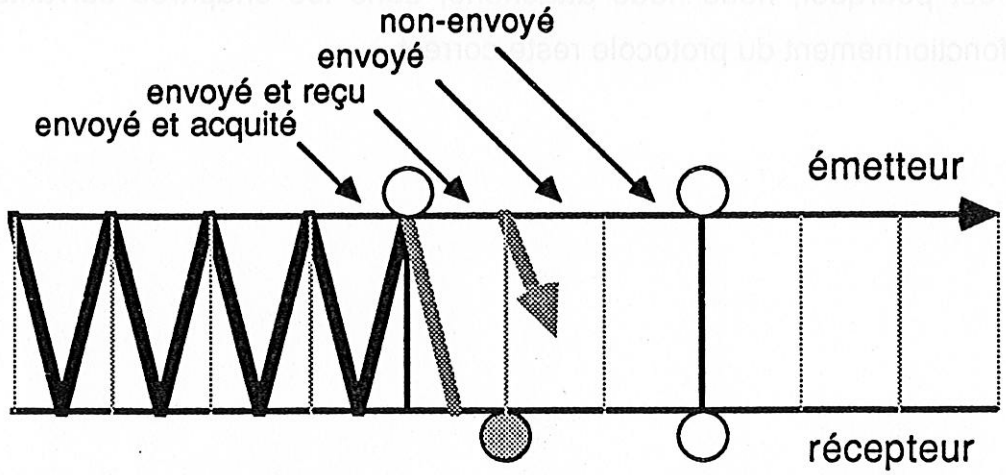
A l'émission, un mécanisme de reprise d'erreur provoque la retransmission par l'émetteur des messages situés à l'intérieur de la fenêtre d'émission. Ce mécanisme est déclenché après que le laps de temps, que l'on considère comme maximal pour la transmission d'un message et de son acquittement soit écoulé, ou qu'un message de rejet provoque une resynchronisation. Ces deux mécanismes de détection et de récupération d'erreur, bien que pris en compte dans la modélisation que nous faisons du protocole, ne sont pas modélisés de manière explicite. Seul l'est leur aspect

externe, et ce, pour des raisons de contraintes temporelles difficiles à exprimer dans le modèle en RdP à prédicats, et pour des choix d'implantation difficiles à faire, ou qui restreindraient trop le modèle. Ne pas faire de choix, nous permet d'offrir un modèle qui synthétise l'ensemble des politiques de détection et de récupération d'erreur.

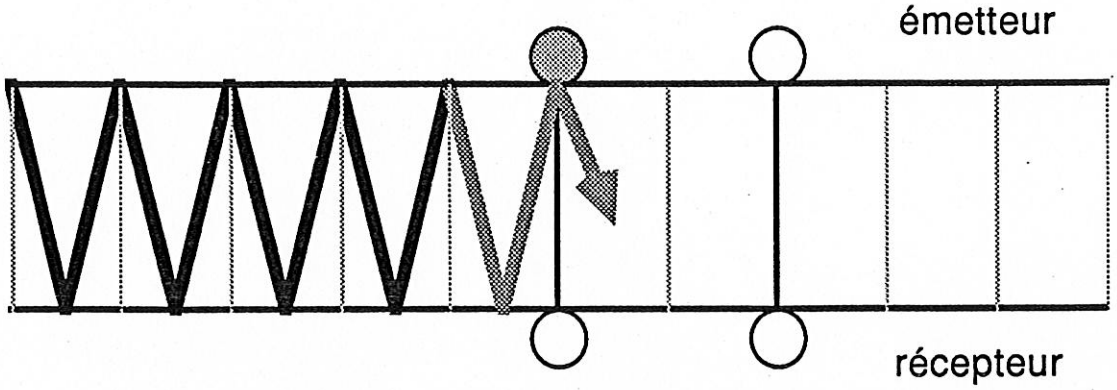
La classe 3 du protocole de la couche Transport à la particularité, non seulement de permettre la variation de la largeur de la fenêtre (appelé alors crédit), mais aussi, la possibilité de réduire arbitrairement cette largeur (réquisition) (Figure C-2.4). Et ce, bien que le crédit associé ait déjà pu être exploité par l'émetteur pour envoyer un message. Le message devient illégal soudainement. On comprend que cette technique, bien que très souple et permettant un contrôle de flux très fin, est dangereuse. C'est pourquoi, nous nous attachons, dans les chapitres suivants, à prouver que le fonctionnement du protocole reste correct.



La fenêtre



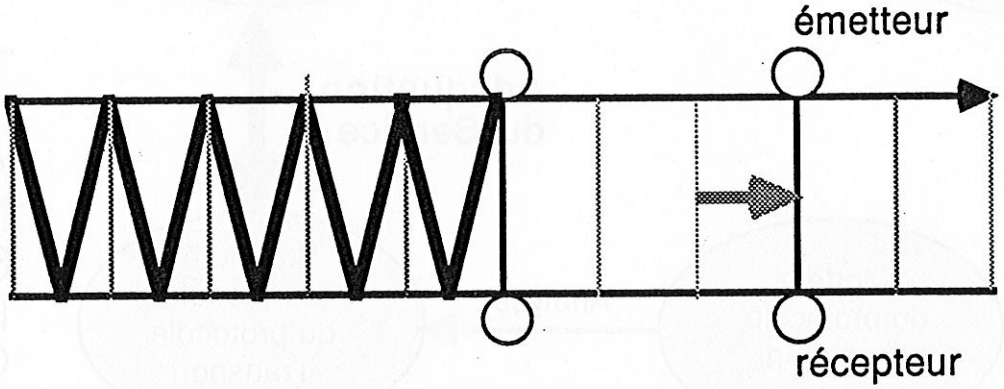
augmentation de la borne inférieure du récepteur



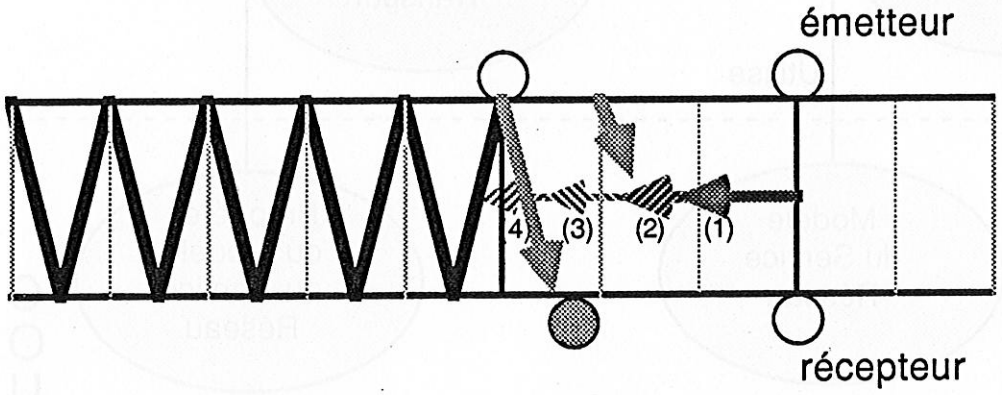
augmentation de la borne inférieure de l'émetteur

-C-2.4- Attribution de crédit -

Augmentation de la borne supérieure : l'attribution de crédit

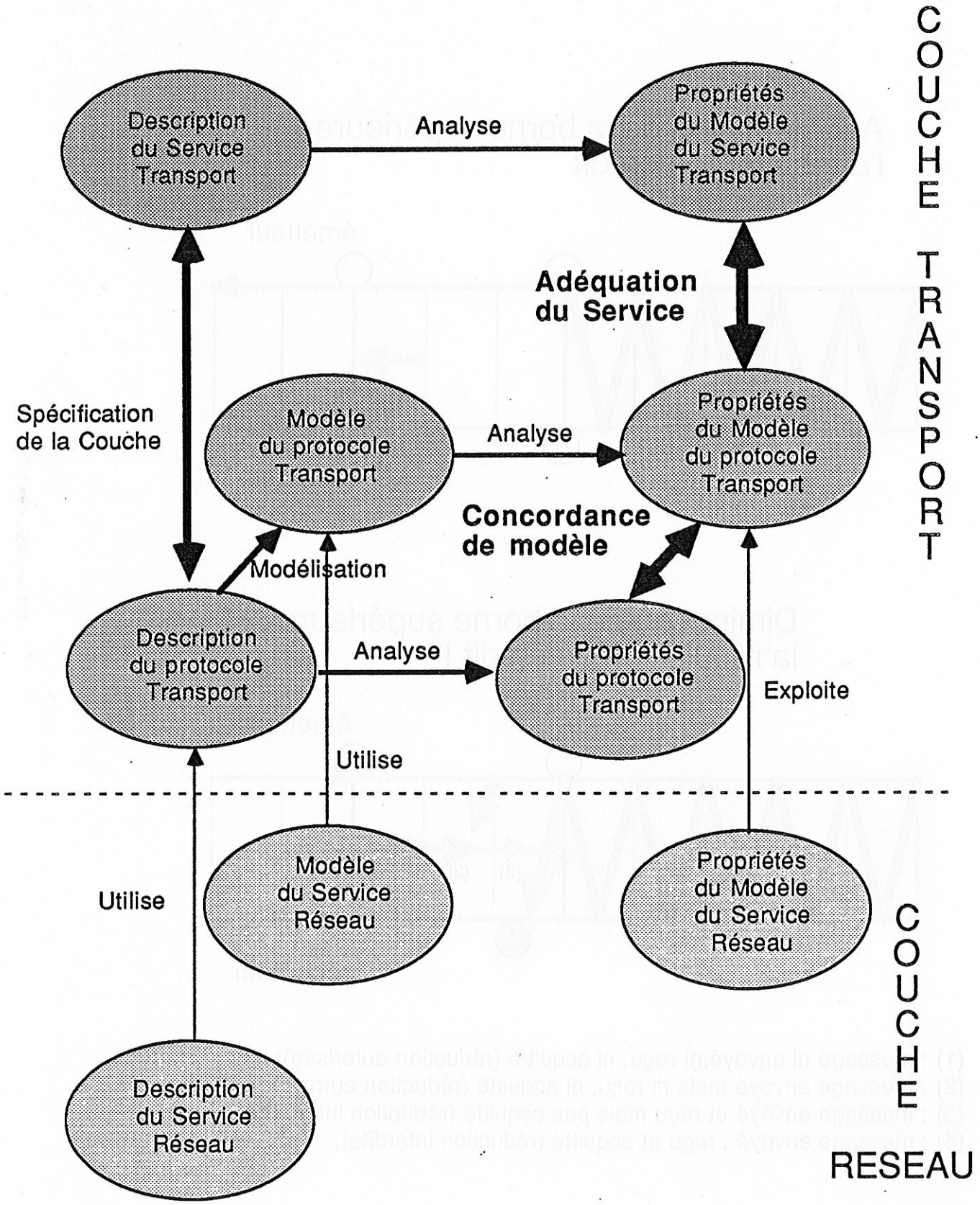


Diminution de la borne supérieure : la réquisition de crédit !?!



(1) : message ni envoyé,ni reçu, ni acquitté (réduction autorisée);
(2) : message envoyé mais ni reçu, ni acquitté (réduction autorisée);
(3) : message envoyé et reçu mais pas acquitté (réduction interdite);
(4) : message envoyé , reçu et acquitté (réduction interdite);

- Figure C-2.5- Adéquation de service
et Concordance de la couche Transport



2.4 Les spécifications du protocole Transport

Conformément aux normes internationales régissant le protocole de la couche Transport, et en application à notre méthode de modélisation et de validation (Figure C-2.5), nous allons définir un ensemble de propriétés qui caractérisent les fonctionnalités du protocole en phase de transfert de données.

Ces propriétés expriment le comportement interne du protocole de Transport et permettront d'établir dans le quatrième paragraphe la concordance du modèle.

E0 : Le protocole Transport attribue un numéro à chaque message de données qu'il manipule. Ce numéro est défini à partir d'un compteur incrémenté modulo N.

E1 : Le protocole Transport possède une fenêtre d'émission de largeur maximale f. Chaque nouveau message de données à transmettre incrémente la borne supérieure de la fenêtre, jusqu'à concurrence du crédit alloué. A chaque message de données acquitté, la borne inférieure augmente.

E2 : Tout message compris dans la fenêtre d'émission peut être émis ou réémis vers l'entité réceptrice du protocole de Transport.

E3 : Les messages d'acquittement doivent avoir un numéro d'acquittement compris dans la fenêtre d'émission, et un crédit positif, c'est-à-dire supérieur ou égal au précédent.

E4 : Les messages de rejet doivent avoir un numéro d'acquittement compris dans la fenêtre d'émission, et peuvent avoir un crédit négatif, d'où une réduction de la largeur de la fenêtre d'émission.

E5 : L'ensemble des messages n'ayant pas une syntaxe correcte sont rejetés.

R1 : Le protocole de la couche Transport définit une fenêtre de réception, de largeur maximale f.

R2 : Tout message de données, dont le numéro est compris dans l'intervalle formé par la fenêtre de réception, est mémorisé par le récepteur.

R3 : Chaque accusé de réception ou message d'acquittement comporte la valeur de la borne inférieure et de la borne supérieure de la fenêtre de réception.

R4 : Les messages de rejet comportent aussi les bornes inférieure et supérieure de la fenêtre de réception. Ils sont émis après diminution du crédit ou détection d'une erreur dans le protocole.

R5 : L'ensemble des messages n'ayant pas une syntaxe correcte est rejeté, ainsi que les messages de données dupliqués.

R6 : La détection d'un type de message imprévu ou la signalisation par la couche Réseau d'une resynchronisation, fait passer le protocole dans un état permettant la reprise.

2.5 Les spécifications du service Transport

Comme pour le service Réseau, les fonctionnalités du service de la couche Transport sont beaucoup plus simples que celles du protocole. Cette simplification est l'un des gains de la notion de service. Les propriétés du service peuvent être vues comme les propriétés externes du protocole.

Nous allons les définir ici pour pouvoir établir à la fin du quatrième paragraphe l'adéquation du modèle construit au paragraphe trois, conformément à notre méthode.

T1 : Le protocole de la couche Transport assure en permanence son service, il ne se bloque pas.

T2 : Le protocole de la couche Transport délivre les messages de données sans omission, ni duplication, dans l'ordre même où ils ont été remis.

3. Les MODELES du PROTOCOLE de la couche TRANSPORT

3.1 Introduction

Nous allons construire un modèle du protocole de Transport en phase de transfert de données, en lui adjoignant au fur à mesure l'ensemble de ses fonctionnalités. Ce modèle utilise les réseaux de Petri à prédicats, qui ont déjà montré leurs nombreux avantages dans la partie précédente. Ce modèle est nettement plus complexe que celui du service Réseau, car il intègre un ensemble de fonctionnalités jamais encore modélisées.

Nous partons d'un modèle où le contrôle de flux et d'erreur est basé sur le mécanisme bien connu de la "fenêtre" employé par exemple dans les protocoles de couche inférieure (HDLC ou X25 par exemple). Cette fenêtre est de largeur constante, c'est-à-dire que le contrôle de flux se fait sur un nombre fixe de messages .

Le deuxième modèle intègre la possibilité de variation de cette largeur de fenêtre, ce qui permet au récepteur d'affiner le contrôle de flux en fonction de ses possibilités. La valeur de la largeur s'appelle le crédit, elle peut varier de 0 à f largeur maximale de la fenêtre.

Le troisième modèle permet au récepteur de recevoir les messages de manière non-séquentielle, c'est-à-dire dans un ordre différent de leur ordre d'émission. Cette fonctionnalité est impérative si le média déséquentialise les messages qu'il transporte, mais utile aussi, si le média perd ou duplique les messages.

Le quatrième et dernier modèle autorise la diminution de la largeur de la fenêtre de manière autoritaire par le récepteur. Cette diminution est précisément le point le plus critique du protocole, comme nous le verrons par la suite, car il remet en cause un accord préalable sur l'attribution d'un certain crédit.

Cette méthode de construction du modèle n'établit en rien sa validité. Elle

permet seulement de le concevoir aussi méthodiquement que possible, et surtout, d'introduire petit à petit l'ensemble des fonctionnalités du protocole sans exhiber, ex-abrupto, le modèle entier réalisé.

Toutefois, si l'on disposait de l'ensemble de la théorie et des outils de réduction développés pour les RdP ordinaires, sur les RdPàP, cette méthode de construction du modèle pourrait être un guide pour appliquer les réductions adaptées!

Nous avons construit graphiquement le modèle de l'émetteur séparément de la partie réceptrice, bien qu'en fait, une connexion de Transport soit une liaison bidirectionnelle. Ainsi toute entité Transport connectée au réseau est à la fois émettrice et réceptrice. Cependant, l'indépendance de chaque partie autorise notre démarche, ce qui facilite la conception, la lecture et la preuve du modèle. Pour construire un modèle dans sa totalité, il faut pour chaque entité communiquante: une partie émettrice, et une partie réceptrice.

Nous avons nommé les places et transitions du modèle en respectant une syntaxe particulière, qui permet de reconnaître par son nom une place ou une transition, et de savoir si elle appartient à la partie émettrice ou réceptrice du modèle. Nous avons les préfixes suivants:

- "Pe-" et "Te-" : Place et transition de l'émetteur;
- "Pr-" et "Tr-" : Place et transition du récepteur;

Notre modèle du protocole de la couche Transport possédant pour chaque arc un arc inverse, afin d'avoir un graphisme plus concis, nous avons choisi de ne pas orienter les arcs, ni de les dupliquer. Seul, les uplets associés aux arcs possèdent un symbole (<,>,v,^) qui permet d'en déduire le sens.

3.2. Premier modèle (la fenêtre)

3.2.1 Fonctionnalités

Ce premier modèle a les fonctions bien connues de protocole de type simple (HDLC). Il est basé essentiellement sur le mécanisme de fenêtre explicité au cours du paragraphe précédent.

Nous construisons notre modèle personnel à partir du modèle du protocole de la couche liaison [Berthelot 81], en renommant les places et en le modifiant quelque peu.

On y retrouve, comme au cours des quatre modèles qui suivent, une partie émettrice (appelée ainsi car c'est elle qui émet les messages de données) et une partie réceptrice (elle reçoit les messages de données), et faisant le lien entre elles un sous-modèle du service Réseau.

Le modèle de la partie émettrice comporte:

- une interface avec la couche supérieure Session (réception des messages);
- une fenêtre d'émission avec une borne inférieure et une borne supérieure, et son mécanisme de retransmission;
- une interface avec la couche inférieure Réseau (émission des données, réception des acquittements).

Le modèle de la partie réceptrice comporte:

- une interface avec la couche supérieure Session (délivrance des messages);
- une fenêtre de réception avec une borne inférieure et une borne supérieure, et son mécanisme d'acquittement;
- une interface avec la couche inférieure Réseau (réception des données, émission des acquittements).

Nous avons rencontré des problèmes de choix de modélisation, notamment

pour la représentation des bornes des fenêtres d'émission et de réception, mais aussi pour le mécanisme de retransmission des messages et de traitement d'erreur.

Les fenêtres ont une borne inférieure et supérieure qui peuvent être modélisées, soit par la valeur d'une des bornes et la largeur de la fenêtre, soit par la valeur de chacune des deux bornes. Nous avons choisi la deuxième solution pour la facilité de mise à jour qu'elle procure.

La norme ne précise pas très explicitement la méthode à employer quant à l'ordre de transmission et de retransmission des messages par l'émetteur. On peut comprendre entre les lignes et par simple bon sens, que l'on émette un message dès que possible, et qu'on retransmette la suite de messages dès qu'un message de rejet est reçu. Cela aurait pour conséquence de transmettre les messages toujours suivant un ordre croissant.

Nous n'avons pas choisi cette solution, trop dépendante à notre avis de choix d'implantation. Nous avons préféré ne pas imposer d'ordre ni de transmission, ni de retransmission. Notre modèle prouve ainsi que le protocole ne dépend pas de ces choix, cependant il est clair qu'un choix judicieux est propice à de bonnes performances, mais ce n'est pas notre propos de les mesurer.

Le traitement des erreurs consiste, soit à ignorer (détruire) les messages reçus qui semblent incohérents vis a vis du protocole, soit à notifier à l'entité distante par l'émission d'un message de rejet la survenue d'une telle incohérence.

Tout au cours de notre modélisation nous avons tenté de faire apparaître le maximum de parallélisme de traitement, entre les parties émettrice et réceptrice, mais aussi pour le traitement des différents types de messages puis de l'action engendrée.

3.2.2 Modélisation (figure C-3.1a et C-3.1b)

Nous allons énumérer l'ensemble des places et transitions composant les

modèles des parties émettrice et réceptrice, en essayant d'expliciter leurs rôles.

Les places de l'émetteur :

La place Pe-compt attribue à chaque message un numéro. Le compteur est incrémenté cycliquement à chaque message provenant de le couche supérieure Session.

La place Pe-inf est la borne inférieure de la fenêtre. Tous les messages de numéro strictement inférieur ont déjà été transmis et acquittés (nota: les numéros supérieurs sont soit à émettre, soit n'ont pas encore été attribués à un message).

La place Pe-fenêtre contient l'ensemble des messages contenus par la fenêtre d'émission. Cette place modélise l'organe de stockage pour la réémission ultérieure des messages.

Les transitions de l'émetteur :

La transition Te-Session est franchie quand l'interface Session délivre à l'entité Transport un message à émettre (T-SDU-data.req). Ce message est placé dans l'organe de stockage (Pe-fenêtre), si celui-ci n'est pas surchargé. Le modèle maintient au maximum 'f' messages, largeur maximale de la fenêtre d'émission.

La transition Te-donnée est franchie à chaque émission d'un message vers le récepteur. On insère, alors, le message dans le sous-modèle Réseau. Cependant une copie du message est conservée en vue d'une retransmission ultérieure.

La transition Te-acq est franchie par les messages d'acquittement issus du récepteur. L'arrivée de l'acquittement provoque la mise à jour de la borne inférieure de la fenêtre d'émission (Pe-inf), à l'aide du numéro d'acquittement que contient le message.

La transition Te-rej est franchie par les messages de rejet issus du récepteur. Le

franchissement met à jour la borne inférieure de la fenêtre d'émission (Pe-inf), à l'aide du numéro d'acquittement que contient le message de rejet.

La transition Te-erreur est franchie par tous les messages ne pouvant pas franchir les autres transitions Te-acquit et Te-rejet. Elle permet de recevoir du médium Réseau des paquets, qui ne sont ni des acquittements, ni des rejets (tous les paquets erronés).

Les places du récepteur :

La place Pr-inf mémorise la borne inférieure de la fenêtre de réception. La valeur est exactement le numéro du prochain message à recevoir de l'entité émettrice Transport. A chaque réception d'un message correct le compteur s'incrémente (Tr-donnée).

La place Pr-état mémorise l'état de la connexion du protocole de Transport. L'état du protocole devient incorrect à la réception d'un message de réinitialisation issu de la couche Réseau, ou sur tout événement déclenché par un message inattendu ou incorrect. Le protocole devra alors émettre un message de rejet, afin de se resynchroniser avec l'émetteur (Tr-rej).

Les transitions du récepteur :

La transition Tr-donnée modélise à la fois la réception d'un message correct et sa mise à disposition à l'interface Session (T-SDU-data.ind). La qualité correcte d'un message est déterminée par l'égalité de son numéro et de celui attendu par le récepteur (Pr-donnée).

La transition Tr-hors représente la réception et la détection d'un message déjà reçu ou non-attendu (hors des bornes de la fenêtre). Le message est supprimé.

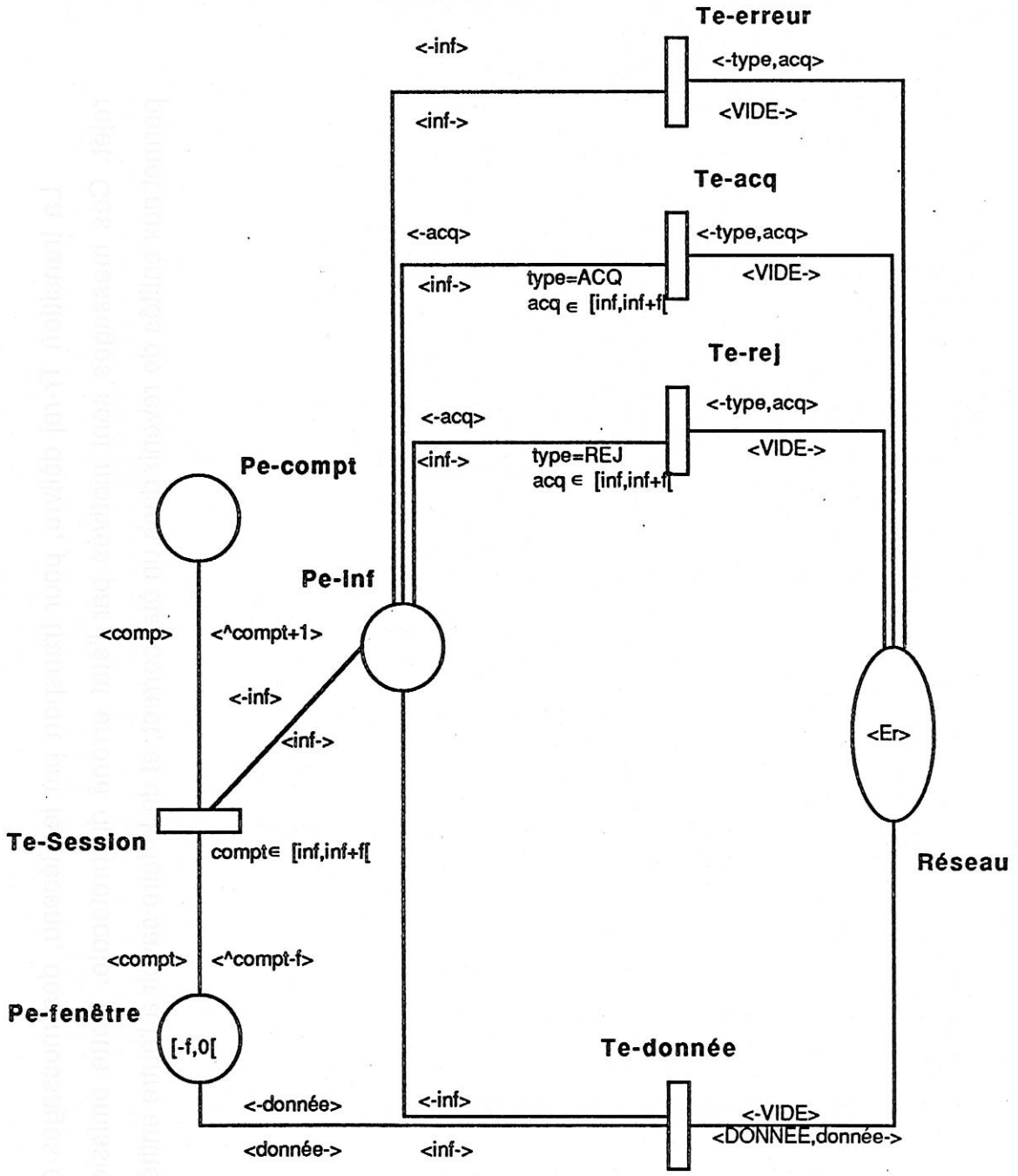
La transition Tr-erreur est franchie pour chaque message reçu, erroné inattendu, ou provenant d'une désynchronisation. Elle joue le même rôle que la

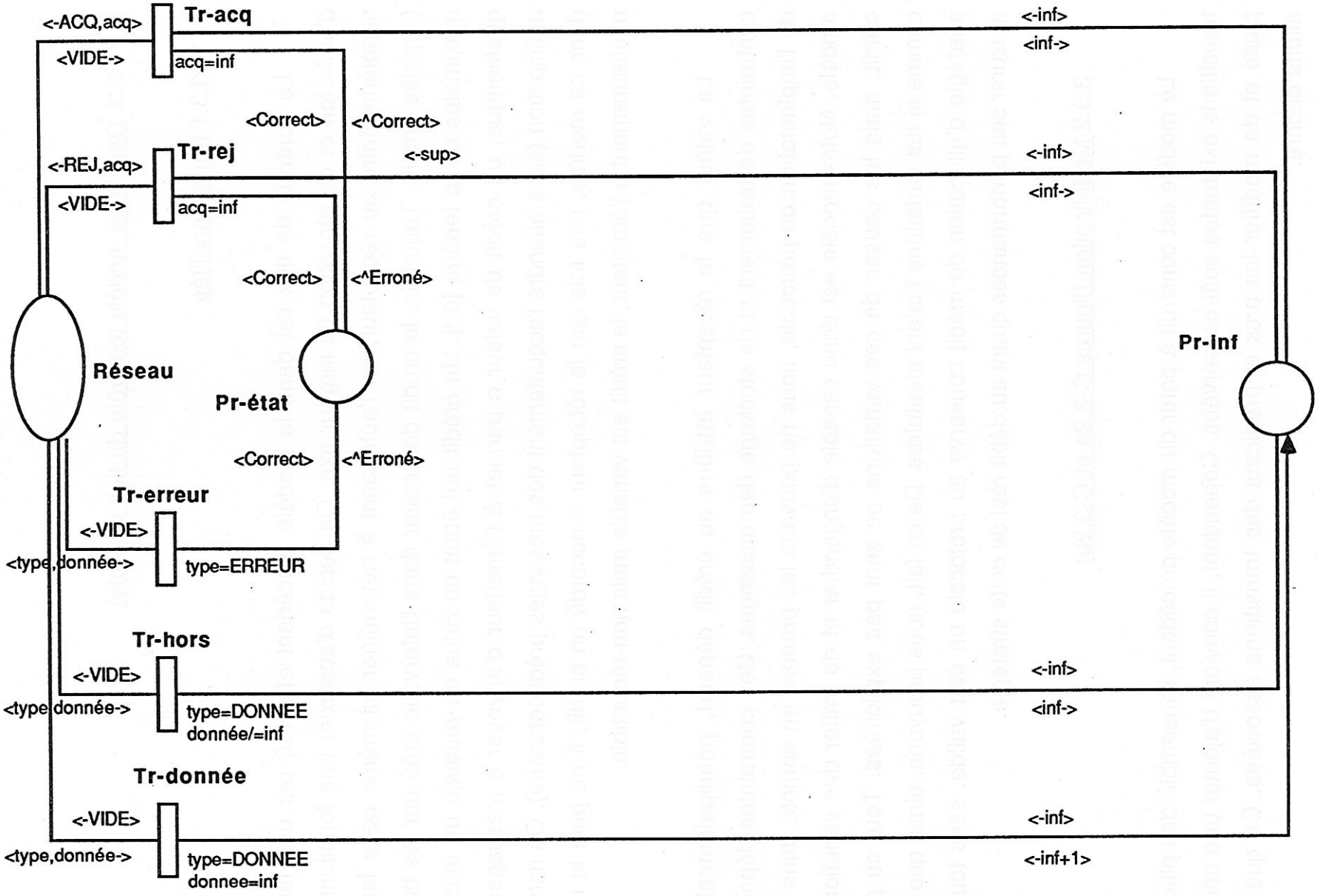
transition Te-erreur en acceptant du médium tous les messages inattendus ou erronés.

La transition Tr-acq délivre, pour transport par le Réseau, les messages d'acquittement. Le numéro du message acquitté est basé sur la valeur de la borne inférieure de la fenêtre de réception (Pr-inf).

La transition Tr-rej délivre, pour transport par le Réseau, des messages de rejet. Ces messages sont motivés par l'état erroné du protocole. Cette émission permet aux entités de revenir dans un état correct, et de le faire savoir à l'autre entité.

- Figure C-3.1a -
Modèle émetteur du protocole Transport





- Figure C-3.1b - Modèle Récepteur du protocole Transport

3.3. Deuxième modèle (contrôle de flux variable)

3.3.1 Fonctionnalités

Le contrôle de flux est dans le modèle précédent effectué par une fenêtre d'émission et de réception de largeur fixe. On décide d'accorder une fonctionnalité supplémentaire au récepteur, en l'autorisant à déterminer lui-même cette largeur (appelée crédit). Toutefois, le crédit doit rester dans l'intervalle créé par les bornes maximales de la fenêtre [0,f]. Un crédit nul étant un refus de recevoir un message quelconque, un crédit de valeur c permet à l'émetteur d'envoyer c messages par anticipation (sans attendre l'acquittement des messages prédécesseurs). On note que dans ce modèle, une fois que le récepteur a accordé un crédit, il ne peut le retirer ultérieurement à l'émetteur, le crédit est variable mais non-réductible.

La valeur que le récepteur attribue au crédit dépend, principalement des contraintes de traitement et de stockage des messages. Ces contraintes dépendant de l'implantation du protocole, nous ne pouvons les prendre en compte dans notre modèle. Notre modèle est donc capable d'engendrer et de traiter des variations de crédit, mais les causes de ces variations ne sont pas explicitées. Tout se passe comme si les variations étaient aléatoires. De ce fait, nous prouvons, ainsi, que toute stratégie d'allocation de crédit conserve au protocole un état valide, sans toutefois mesurer ses performances quant au débit réel de cette stratégie.

3.3.2 Modélisation (figures C-3.2a et C-3.2b)

Le modèle est construit à partir du modèle précédent, l'ensemble des places et transitions existantes sont conservées. Cependant, il convient d'ajouter une nouvelle place et de modifier les arcs et prédicats des transitions associées. Ce que nous allons décrire.

Les places de l'émetteur :

La place Pe-sup modélise la borne supérieure de la fenêtre d'émission. Sa marque contient la valeur du crédit alloué par le récepteur. Dans le modèle précédent, cette place était implicite car la valeur du crédit était constante.

Les transitions de l'émetteur :

La transition Te-donnée modélise l'interface Transport/Réseau. L'envoi de message est maintenant conditionné par le crédit d'émission (Pe-sup). Le numéro du message à émettre doit être dans l'intervalle de la fenêtre d'émission.

Les transitions Te-acq et Te-rej réceptionnent les messages d'acquittement et de rejet s'ils sont valides (leur numéro d'acquittement et de crédit sont valides si l'acquittement est compris dans les bornes de la fenêtre d'émission, le crédit est non-décroissant et inférieur à la largeur maximale de la fenêtre), et mettent à jour les nouvelles valeurs des bornes inférieure et supérieure de la fenêtre (Pe-inf et Pe-sup).

La transition Te-erreur réceptionne tous les autres messages, comme dans le premier modèle.

Pour le récepteur, on ajoute de même la place Pr-sup :

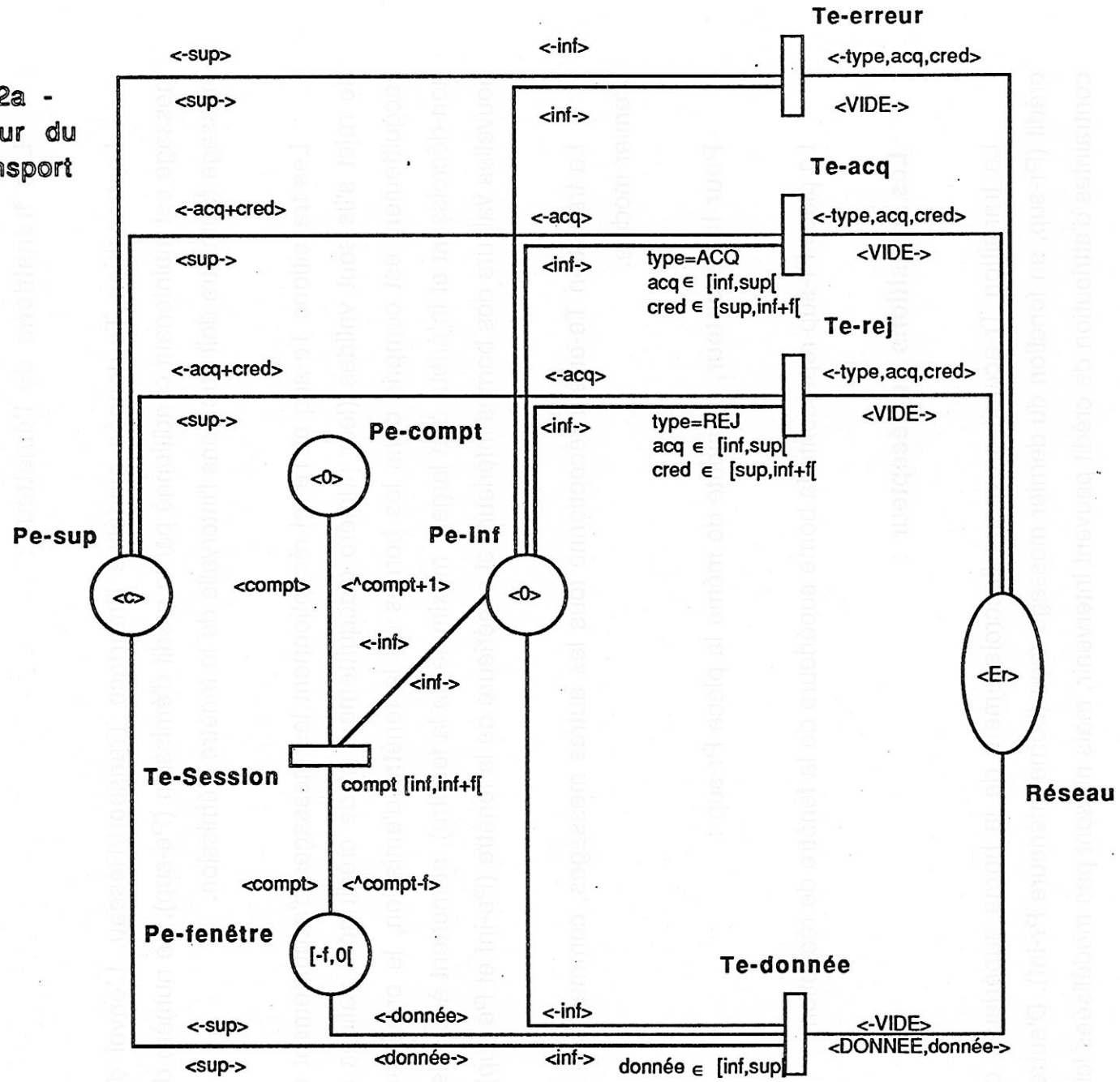
La place Pr-sup représente la borne supérieure de la fenêtre de réception.

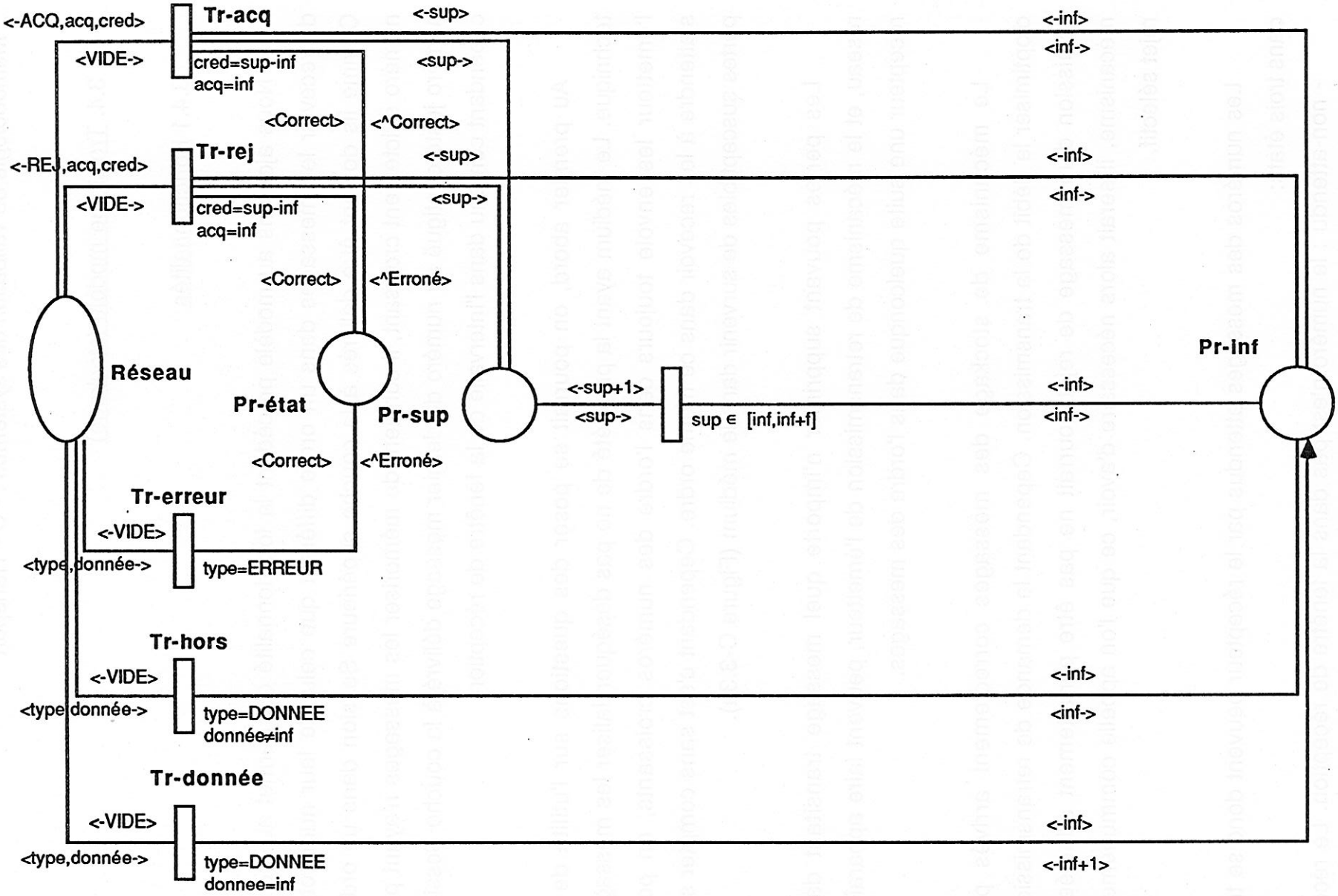
Les transitions du récepteur :

La transition Tr-accroit provoque l'accroissement de la borne supérieure ou crédit (Pr-sup, en fonction du dernier message reçu (borne inférieure Pr-inf). D'autres contraintes d'attribution de crédit peuvent intervenir, mais ne sont pas modélisées ici.

Les transitions Tr-acq et Tr-rej sont complétées de manière à positionner le nouveau champ crédit des messages d'acquittement et de crédit avant leur émission. Ce positionnement est effectué en fonction de la borne supérieure et inférieure de la fenêtre de réception (Pr-sup, Pr-inf).

- Figure C-3.2a -
Modèle émetteur du protocole Transport





- Figure C-3.2b - Modèle Transport Récepteur

3.4. Troisième modèle (stockage)

3.4.1 Fonctionnalités

Nous ajoutons au modèle précédent la fonctionnalité qui permet au récepteur de recevoir les messages dans un ordre différent que celui de leur numérotation. Comme ils doivent être délivrés à la couche supérieure Session dans un ordre de numéro strictement croissant, il convient de mémoriser les messages n'ayant pas un numéro juste contiguë au numéro du dernier message délivré à la couche Session, et cependant contenu dans l'intervalle de la fenêtre de réception.

Au premier abord, on pourrait se poser des questions sur l'utilité de cette technique. Le médium ayant la propriété de ne pas déséquentialiser les messages, si l'émetteur les envoie toujours dans l'ordre des numéros croissants, on pourrait s'attendre à les recevoir dans ce même ordre. Cependant c'est sans compter sur les pertes susceptibles de survenir dans le médium (Figure C-3.3a).

Les pertes pouvant supprimer n'importe quel message transitant dans le réseau, et le mécanisme de retransmission de l'émetteur, peuvent faire apparaître au récepteur une suite quelconque dans l'ordre des messages.

Le mécanisme de stockage des messages correctement arrivés permet d'optimiser le débit de la transmission. Cependant la demande de retransmission par l'émission d'un message de rejet pourrait ne pas être parfaitement adaptée à ce mécanisme, il serait alors nécessaire d'avoir, ce que l'on appelle communément, un rejet sélectif.

Les numéros des messages attendus par le récepteur peuvent donc se trouver dans trois états:

- non-attendu : le numéro n'est pas dans la fenêtre de réception. Le récepteur ne doit recevoir aucun message comportant un numéro de ce type.
- attendu et non-reçu : le numéro est dans la fenêtre de réception, mais le message correspondant n'a pas été encore reçu. Le récepteur mémorisera tout

message comportant un numéro de ce type.

- attendu et reçu : le numéro est dans la fenêtre de réception, et le message a déjà été reçu par le récepteur. Le récepteur considère tout nouveau message reçu comportant un numéro de ce type comme un message dupliqué. De ce fait, il l'ignorera.

Chaque numéro passe cycliquement par les trois états:

- Un numéro "non-attendu" passe dans l'état "attendu, non-reçu" quand le récepteur augmente le crédit de telle manière que le numéro appartienne, à partir de cet instant, à l'intervalle de la fenêtre de réception (Tr-accroit).

- Un numéro "attendu, non-reçu" devient "attendu, reçu" à la réception du message correspondant à ce numéro (Tr-reçu).

- Un numéro "attendu, reçu" passe dans l'état "non-attendu" après qu'on ait délivré effectivement le message associé à la couche Session (Tr-Session). Simultanément à cette délivrance, la borne inférieure de la fenêtre de réception est augmentée, ce qui explique que le message n'appartienne plus à cette fenêtre.

3.4.2 Modélisation (figure C-3.3b)

Afin, de modéliser précisément la gestion des numéros de messages, nous adjoignons deux nouvelles places au modèle précédent. De nouvelles transitions sont également nécessaires pour assurer la fonctionnalité supplémentaire et certains prédicats doivent être complétés. Cette fonctionnalité affecte exclusivement la partie réceptrice du modèle.

Les places du récepteur :

La place Pr-attendu mémorise l'ensemble des numéros des messages attendus (dont les numéros sont dans l'intervalle de la fenêtre de réception), mais non encore reçus par le récepteur.

La place Pr-reçu mémorise l'ensemble des numéros des messages attendus et reçus par le récepteur, mais pas encore délivrés à l'entité de la couche supérieure.

On note que l'union des numéros des uplets de ces deux places représentent exactement l'intervalle qui s'inscrit entre les bornes inférieure et supérieure de la fenêtre de réception.

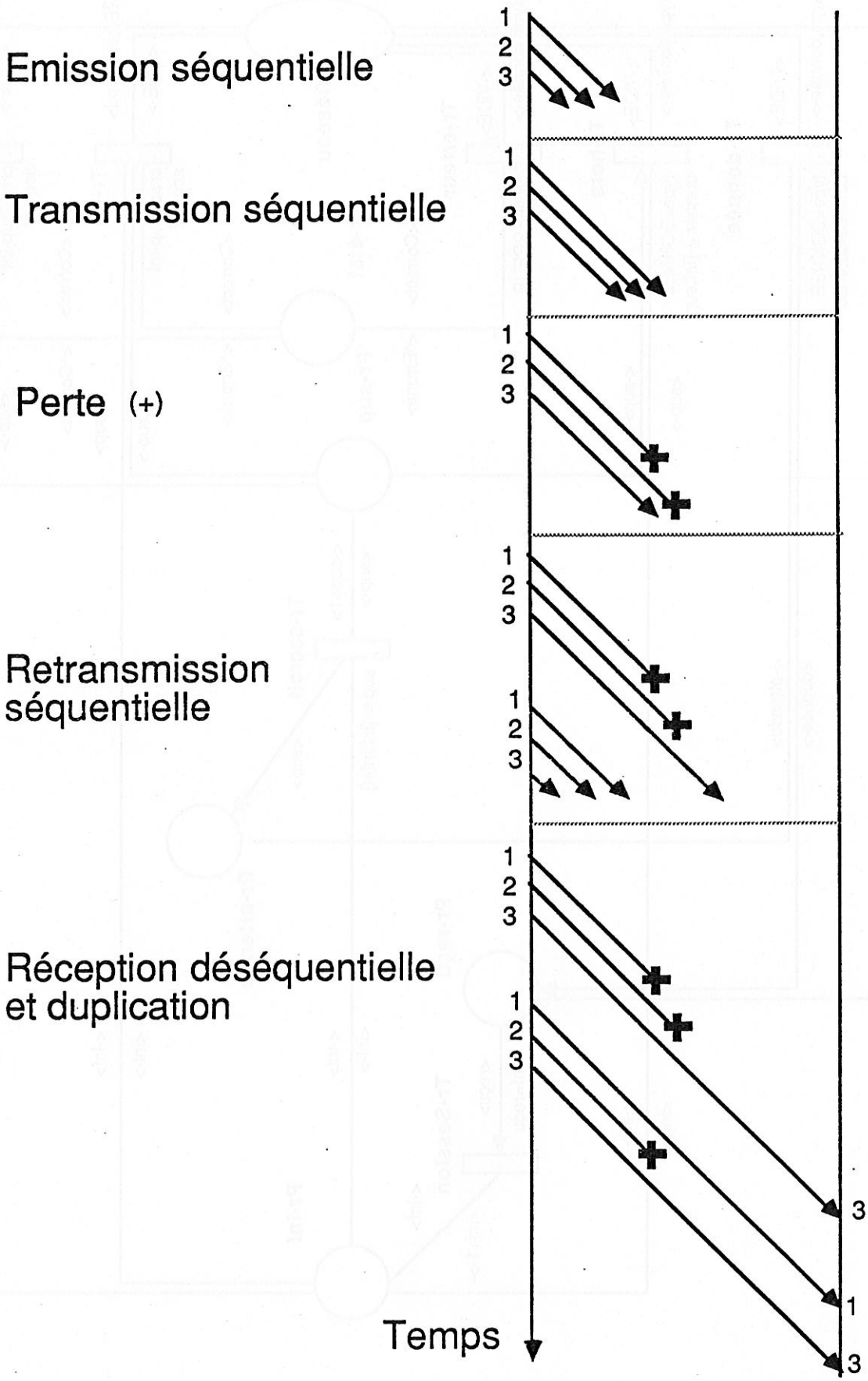
Les transitions du récepteur :

La transition Tr-accroit provoque l'accroissement de la borne supérieure de la fenêtre de réception (Pr-sup). Cette augmentation de crédit provoque l'apparition dans la place Pr-attendu d'une marque contenant le nouveau numéro autorisé à la réception. L'accroissement du crédit est une décision locale, qui se fait d' une seule unité à chaque fois. Un accroissement de plusieurs unités de crédit s'effectue par un franchissement multiple de la transition Tr-accroit.

La transition Tr-Session délivre les messages à la couche supérieure Session. Elle assure que la livraison respecte la séquentialité voulue (tout message transitant avant tout autre message au travers de l'interface émetteur Session/Transport, franchira de même l'interface récepteur Transport/Session avant ces dits messages). Chaque accroissement de la borne inférieure par délivrance d'un message à la couche Session, provoque le retrait de la place Pr-reçu du jeton de même numéro.

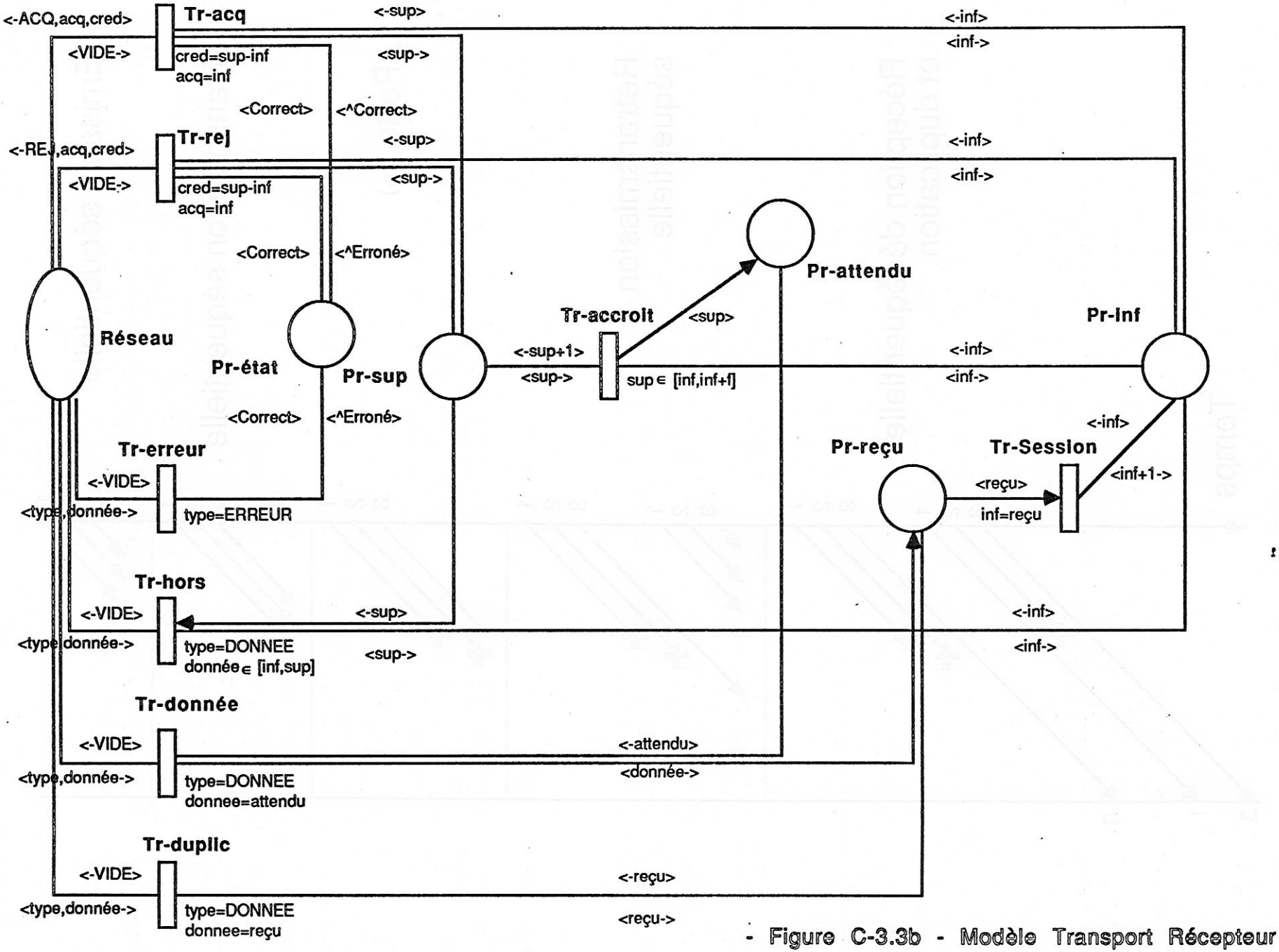
La transition Tr-donnée reçoit les messages en provenance de l'émetteur via le service Réseau. Le message franchit effectivement la transition, uniquement si son numéro appartient bien à l'intervalle de la fenêtre de réception, et s'il n'a pas déjà été reçu (son numéro est contenu par la place Pr-attendu). Le message est mémorisé alors, pour savoir qu'on l'a reçu et pour le délivrer plus tard à la couche Session.

La transition Tr-duplic permet de traiter les messages appartenant à l'intervalle de la fenêtre de réception qui ont déjà été reçus. Cependant ces messages n'ont pas encore été délivrés à la couche Session. Ces doubles sont détruits à la réception. La transition Tr-duplic empêche une double mémorisation des messages reçus en vérifiant leur présence dans l'organe de stockage du récepteur (Pr-reçu).



3 * SEQUENTIEL + PERTE = DESEQUENTIEL + DUPLICATION

- C-3.3a - Réception désequentielle des messages -



- Figure C-3.3b - Modèle Transport Récepteur

3.5. Quatrième modèle (diminution de crédit)

3.5.1 Fonctionnalités

Une des fonctionnalités les plus intéressantes de la classe 3 de la couche Transport, c'est la possibilité de diminution du crédit unilatéralement par le récepteur, après qu'il l'ait accordé. Il revient donc sur sa première décision.

En fait, l'émetteur peut avoir eu avant l'autorisation d'émettre certains messages (dont le numéro faisait alors partie de la fenêtre d'émission), puis n'avoir plus ce droit (la fenêtre s'étant refermée), après que le récepteur ait retiré à l'émetteur le crédit qu'il lui avait préalablement attribué.

Cette technique, bien que définie par la norme du protocole de la couche Transport, est rarement utilisée. Elle permet de résoudre certains problèmes d'engorgements critiques de la voie de transmission et du récepteur, en contrôlant très fermement le contrôle de flux.

Cette diminution ne peut descendre plus bas que la borne inférieure de la fenêtre de réception, c'est-à-dire sans mettre en cause les messages déjà acquittés. L'émetteur est prévenu de cette diminution en recevant, en provenance du récepteur, un message de rejet, comportant un crédit de valeur inférieure à celui précédemment reçu. Il doit alors mettre à jour la borne inférieure de la fenêtre d'émission.

Le récepteur ayant autorisé une certaine valeur de crédit puis moins, l'émetteur peut déjà avoir envoyé les messages correspondants.

Les messages se trouvant dans cet intervalle critique peuvent se être dans une des deux situations suivantes :

- le récepteur peut les avoir déjà reçus et mémorisés. Il conviendra donc en diminuant le crédit de supprimer ces messages redondants.
- le récepteur est susceptible de recevoir de ces messages encore en cours de

transmission au moment de la diminution. Il conviendra de les refuser à la réception.

De manière identique à l'accroissement de crédit, cette décision de diminution est locale à l'entité Transport, et est fixée par des contraintes d'implantation et de performance non décrites dans ce modèle.

3.5.2 Modélisation (figures C-3.4a et C-3.4b)

Le nouveau modèle, qui est le dernier, ne complète le précédent qu'en lui adjoignant deux transitions lui permettant de traiter la diminution du crédit et en ne modifiant que quelques prédicats.

Les places et le transitions de l'émetteur :

Ces places et transitions sont inchangées, seuls maintenant les prédicats validant la réception de message de rejet sont modifiés (Te-rej). Ils acceptent, comme valides, ces messages comportant un champ de crédit dont la valeur est en diminution par rapport à la valeur du message précédent.

Les places du récepteur sont inchangées.

Les transitions du récepteur :

La transition Tr-dec-reç permet au récepteur de diminuer le crédit accordé à l'émetteur. Ayant déjà reçu les messages dont il veut refuser la transmission, il doit les supprimer de l'organe de stockage (Pr-reçu). De même supprimant la marque associée à ce message, il oublie sa réception. Le numéro associé sort de la fenêtre de réception.

La transition Tr-dim-att retire le droit de recevoir les messages hors crédit, en supprimant les marques ayant valeur hors de la fenêtre de réception nouvellement diminuée.

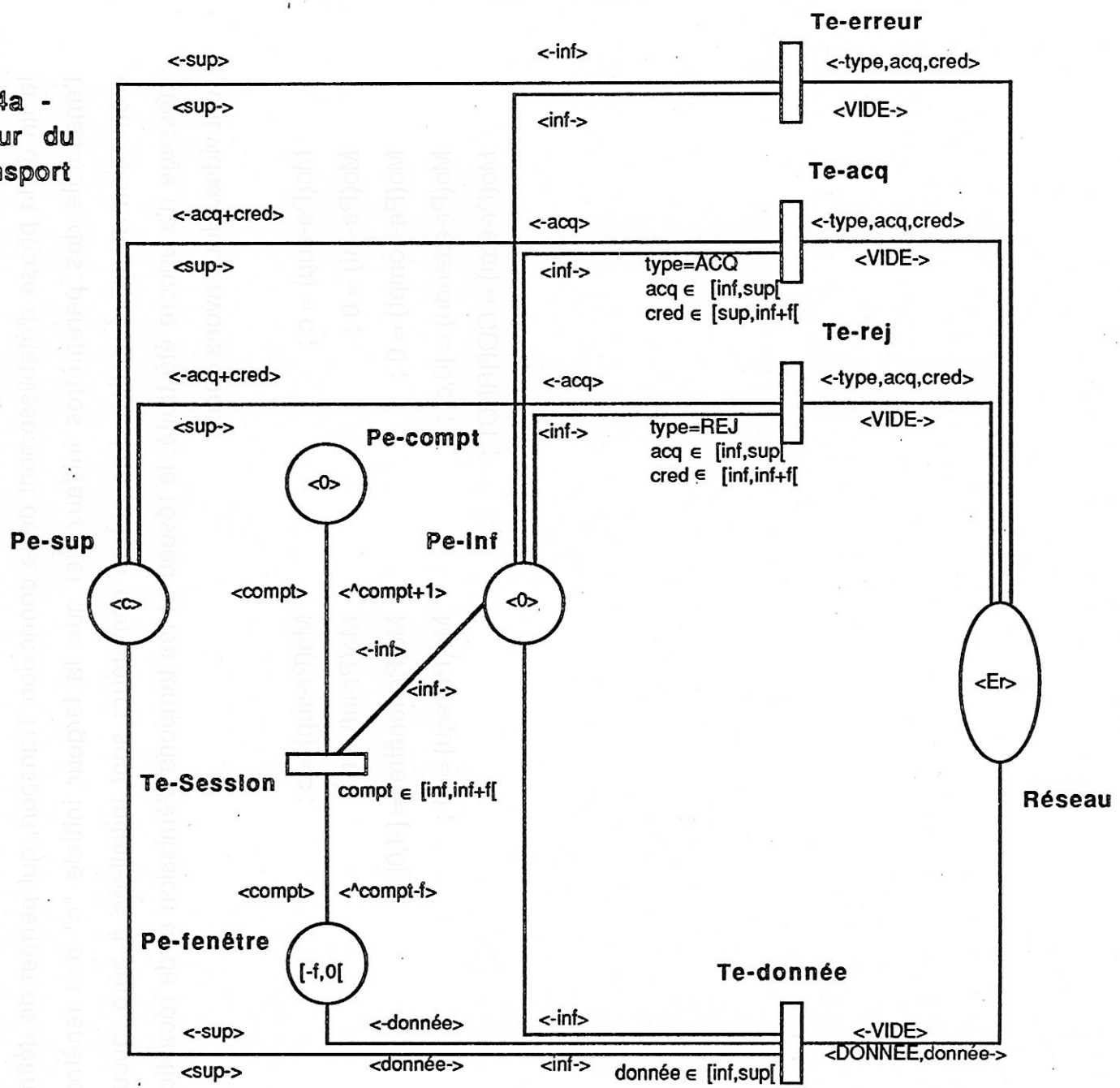
Maintenant que nous avons défini complètement notre modèle, il convient de définir le marquage des places à l'état initial Mo du protocole Transport.

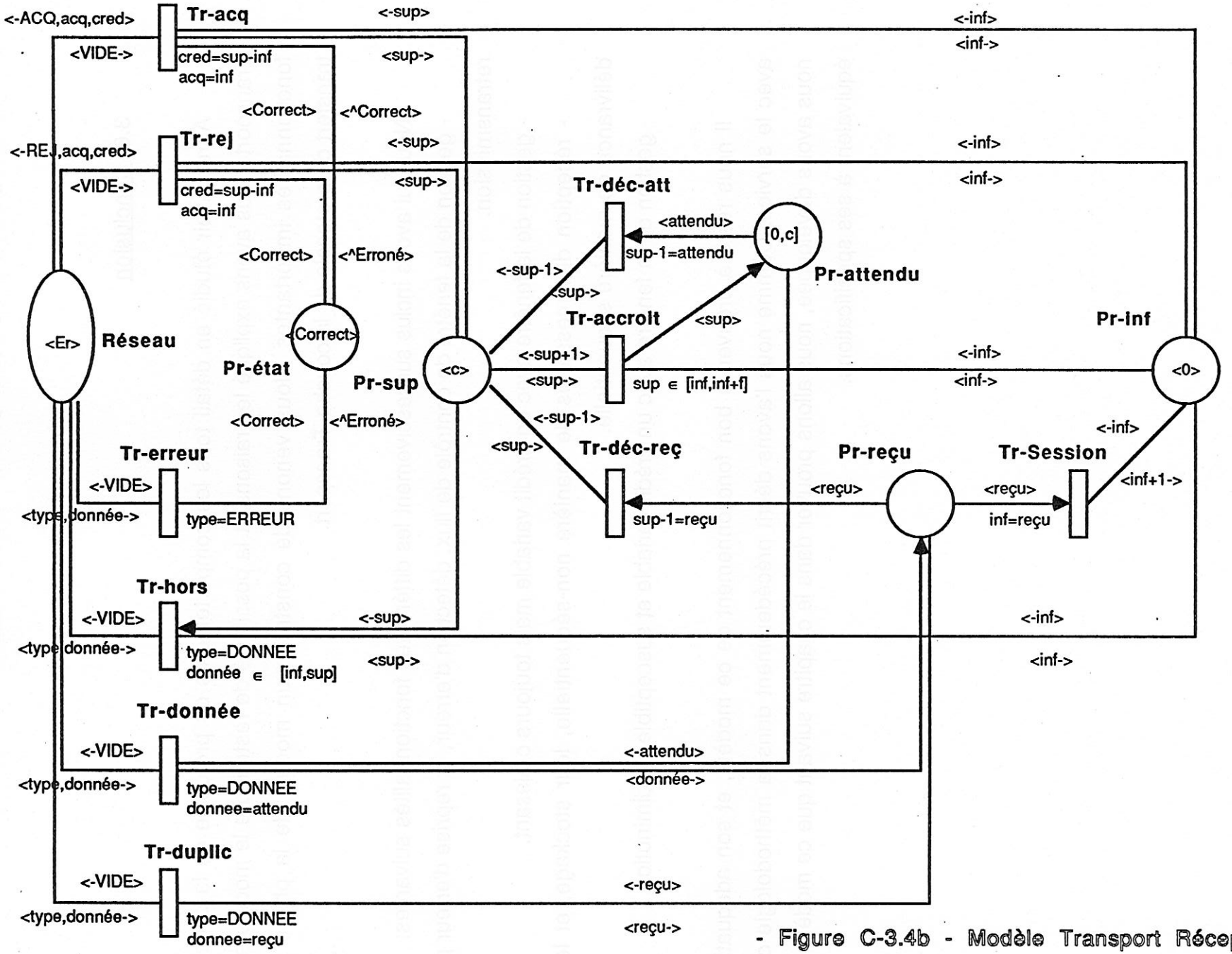
La phase de transfert de données du protocole de la couche Transport débute à la fin de la phase d'établissement de la connexion Transport, qui permet de définir l'ensemble des paramètres initiaux, tel que la largeur initiale "c" des fenêtres d'émission et de réception. Les autres compteurs sont initialisés à zéro, aucun message n'a encore été traité, le réseau et les tampons d'émission et de réception sont vides. Nous avons donc :

Mo(Pe-sup) = c ;
Mo(Pe-inf) = 0 ;
Mo(Pe-compt) = 0 ;
Mo(Pr-attendu) = [0,c[;
Mo(Pr-état) = CORRECT ;

Mo(Pr-sup) = c ;
Mo(Pr-inf) = 0 ;
Mo(Pe-fenêtre) = [-f,0[;
Mo(Pr-reçu) = Ø ;

- Figure C-3.4a -
Modèle émetteur du protocole Transport





- Figure C-3.4b - Modèle Transport Récepteur

3.6. Conclusion

Après avoir étudié en détail tous les comportements du protocole de la couche Transport, nous avons expliqué les raisons et la possibilité de restreindre le modèle aux fonctionnalités importantes. Nous venons de construire un modèle de la phase de transfert du protocole de la couche Transport.

Nous lui avons inclus successivement les différentes fonctionnalités suivantes:

- gestion de la fenêtre de contrôle de flux, détection d'erreur, et reprise d'erreur par retransmission.
- gestion de la fenêtre avec un crédit variable mais toujours croissant.
- réception des messages de manière non-séquentielle, leur stockage, et leur délivrance de manière séquentielle.
- gestion de la fenêtre avec un crédit variable et susceptible de diminution.

Il nous reste à prouver le bon fonctionnement de ce modèle, et son adéquation avec le service. Comme nous l'avons décrit précédemment dans la méthodologie que nous avons développée, nous allons prouver dans le chapitre suivant que ce modèle est équivalent à ses spécifications.

4. VALIDATION du protocole de Transport

4.1. Introduction

Nous allons ici établir dans un premier temps la concordance du modèle du paragraphe précédent, puis l'adéquation de service, après avoir analysé le modèle pour en prouver les propriétés.

En application de notre méthode, cette concordance et cette adéquation seront établies par rapprochement avec les propriétés développées au deuxième chapitre de cette dernière partie.

4.2 Concordance du modèle

Cette concordance de modèle va nous permettre d'établir que le modèle proposé au cours du paragraphe trois correspond bien avec les propriétés issues des spécifications dues aux normes du protocole de la couche Transport, classe trois, en phase de transfert de données. Le modèle ayant été construit conformément aux mêmes fonctionnalités, il est aisé d'établir la concordance du modèle et des spécifications.

La propriété E0 est modélisée par la place Pe-compt et le franchissement de la transition Te-Session, qui attribuent aux messages un numéro modulo N, comme la structure des uplets et le prédicat de la transition le prouvent.

La propriété E1 est établie par la présence des places Pe-inf et Pe-sup modélisant les bornes supérieure et inférieure de la fenêtre d'émission et par le franchissement des transitions Te-acq et Te-rej, qui modifient ces bornes.

La propriété E2 est obtenue directement par le prédicat de la transition Te-donnée, qui autorise l'émission sur la connexion réseau des messages compris

dans la fenêtre d'émission.

De même, les propriétés E3 et E4 sont obtenues respectivement par les prédicats des transitions Te-acq et Te-rej, qui modélisent la réception de messages d'acquittement ou de rejet.

La transition Te-erreur traite la propriété E5, en faisant disparaître les uplets ayant une syntaxe incorrecte.

La propriété R1 est établie par la présence des places Pr-inf et Pr-sup modélisant les bornes supérieure et inférieure de la fenêtre de réception et par le franchissement des transitions Tr-dec, Tr-accroit et Tr-Session, qui modifient ces bornes.

La propriété R2 est réalisée par la transition Tr-donnée, qui mémorise les messages figurant dans la fenêtre de réception comme le prouve le prédicat. Ces messages sont stockés dans la place Pr-reçu.

Les transitions Tr-acq et Tr-rej émettent des uplets comportant les valeurs des places Pr-inf et Pr-sup, qui modélisent la fenêtre de réception. Ces deux transitions modélisent donc la propriété R3.

La propriété R4 est obtenue par les transitions Tr-hors, Tr-duplic et Tr-erreur qui permettent de rejeter les messages de données ayant un numéro hors de la fenêtre de réception, les messages de données dupliqués, ou encore les messages inattendus ou incorrects.

La transition Te-erreur fait bien passer le protocole de l'état correct à l'état incorrect, ce qui plus tard provoque l'émission d'un message de rejet par la transition Tr-rej. Ces deux transitions jointes réalisent bien la propriété R6.

Nous venons de prouver ici que le modèle proposé possède l'ensemble des propriétés qui ont été définies durant la phase de spécification, nous venons donc d'établir la concordance du modèle.

Bien évidemment, il est clair que les quelques propriétés que nous avons définies et modélisées, ne suffisent pas à construire la totalité du protocole Transport. Néanmoins, nous sommes sûrs maintenant que notre modèle a au moins ces quelques bonnes propriétés, qui vont permettre (c'est la preuve du paragraphe suivant) d'obtenir le service de la couche Transport.

Un raffinement du modèle imposerait, suivant notre méthodologie, de nouvelles spécifications. Ce qui permettrait toujours d'établir la concordance du modèle.

4.3 Les Invariants du modèle

Nous allons définir ici un ensemble d'invariants qui nous seront utiles pour prouver que notre modèle du protocole Transport possède de bonnes propriétés.

Assertion A1

A1 : La fenêtre d'émission contient les f derniers messages reçus de la couche Session.

nomess(M(Pe-fenêtre)) = [M(Pe-compt) ~-~ f , M(Pe-compt) [

L'ensemble des numéros de messages contenus par la fenêtre d'émission (nomess(M(Pe-fenêtre))) est égale à l'intervalle borné par le numéro du dernier message reçu de la couche Session (M(Pe-compt)) et réduit de la largeur maximale de la fenêtre d'émission (~-~f).

Preuve :

A **l'état initial**, par définition du marquage M : la place Pe-fenêtre contient un ensemble de marque formant l' intervalle [-f, 0 [; et la place Pe-compt contient une marque nulle ; ce qui prouve que l'assertion A1 est vérifiée à l'état initial.

Le franchissement de la transition **Te-donnée** retire puis remet une même marque de la place Pe-fenêtre, ce qui laisse son marquage inchangé. Si l'assertion était vérifiée avant le franchissement, elle est conservée après le franchissement de la transition Te-donnée.

Le franchissement de la transition **Te-Session** retire de la place Pe-fenêtre une marque ayant pour valeur le marquage de la place Pe-compt, y place une marque ayant pour valeur le marquage de la place Pe-compt plus la largeur maximale de la fenêtre, et incrémente le marquage de la place Pe-compt. En fait,

le tampon d'émission vient de se décaler d'un rang, pour accompagner la marque de la place Pe-compt :

M(Te-Session>M' avec M'(Pe-compt) = M(Pe-compt) ~+~ 1 , et
M'(Pe-fenêtre)=M(Pe-fenêtre)-u1+u2 tel que nomess(u1)=M(Pe-compt)
et nomess(u2) =M(Pe-compt)~+~f.

Si le marquage M respectait l'assertion A1, donc :
nomess(M(Pe-fenêtre)) = [M(Pe-compt) ~-~ f , M(Pe-compt) [
alors le marquage M', obtenu après franchissement de la transition Te-Session vérifie aussi l'assertion A1.

L'ensemble des **autres transitions** du modèle ne modifie pas le marquage des places Pe-compt et Pe-fenêtre, donc si l'assertion était vérifiée avant leur franchissement, elle est conservée après.

Nous venons de prouver que l'assertion A1 est vérifiée à l'état initial, et qu'elle est conservée par le franchissement de toutes les transitions du modèle du protocole Transport. **L'assertion A1** est donc un **invariant** du modèle.

Assertion A2

A2: Les acquittements sont compris dans l'intervalle formé par les bornes inférieures des fenêtres d'émission et de réception.

$\forall$ u $\in$ M(Réseau) , si type(u)=ACQ ou type(u)=REJ alors

noacq(u) $\in$ [M(Pe-inf) , M(Pr-inf)]

Tous les messages d'acquittement ou de rejet circulant dans le Réseau (u) ont un numéro d'acquittement (noacq(u)) compris entre le numéro du dernier message acquitté de bout en bout (M(Pe-inf) et le numéro du dernier message correctement reçu par le récepteur (M(Pr-inf)).

Preuve :

A **l'état initial**, le modèle du service Réseau est vide, donc il n'y a aucun message d'acquittement ou de rejet en transit pour appliquer l'assertion A2. L'assertion A2 est vérifiée par défaut à l'état initial.

Les transitions **Te-acq, Te-rej** modifient de manière similaire le marquage du modèle, elles enlèvent une marque du Réseau et la remplace par une marque VIDE. La place Pe-inf prend pour valeur le numéro d'acquittement de la marque ayant franchi la transition.

M(Te-acq ou Te-rej>M' : M'(Réseau)=M(Réseau)-u1+u2 avec
typmess(u1)=ACQ et typmess(u2)=VIDE , et
M'(Pe-inf)=noacq(u1) .

D'après le prédicat de la transition, nous savons que :
noacq(u1) = [M(Pe-inf) , M(Pe-sup) [

et d'après l'assertion A12 nous savons que les messages d'acquittements forment une suite croissante et d'après les propriétés du modèle Réseau, nous savons que le message délivré possède le rang le plus grand (NBEMPL) ,donc:

$\forall$ u $\in$ Réseau , noacq(u) >= noacq(u1)

Nous en déduisons que si l'ensemble des marques présentes dans le Réseau vérifiait l'assertion A2 avant le franchissement d'une des transitions Te-acq, Te-rej, l'ensemble réduit du Réseau et la nouvelle valeur de la place Pe-inf vérifie l'assertion tout autant après le franchissement.

La transition **Te-erreur** retire et remet la marque de la place Pe-inf sans en modifier la valeur. Le franchissement supprime aussi du Réseau un message. Nous en déduisons que si l'ensemble des marques présentes dans le Réseau vérifiait l'assertion A2 avant le franchissement d'une des transitions Te-acq, Te-rej,

l'ensemble réduit vérifie l'assertion tout autant après le franchissement.

La transition **Te-Session** ne modifie pas le marquage des places utilisées par l'assertion A2.

La transition **Te-donnée** modifie le marquage de la voie opposée à celle contenant les messages d'acquittement ou de rejet. Elle n'intervient pas dans le marquage des places exprimant l'assertion A2.

Les transitions **Tr-erreur, Tr-hors, Tr-donnée et Tr-duplic** modifient le marquage de la voie opposée à celle contenant les messages d'acquittement ou de rejet. Elles n'interviennent pas dans le marquage des places exprimant l'assertion A2.

Les transitions **Tr-dec-att et Tr-dec-rec** n'interviennent pas dans le marquage des places exprimant l'assertion A2.

Les transitions **Tr-acq et Tr-rej** modifient de manière similaire le marquage du Réseau, en y plaçant un message d'acquittement ou de rejet comportant un numéro d'acquittement, ayant la valeur de la borne inférieure de la fenêtre de réception.
M(Tr-acq ou Tr-rej>M' : M'(Réseau)=M(Réseau)-u1+u2 avec
d'après le prédicat de la transition typmess(u1)=VIDE ,
typmess(u2)=ACQ ou REJ et noacq(u2)=M(Pr-inf).

Si l'assertion est vérifiée avant le franchissement d'une des transitions Tr-acq ou Tr-rej, le franchissement provoque l'émission d'un message de rejet ou d'acquittement dont le numéro respecte les conditions de l'assertion A2. L'assertion est vérifiée après le franchissement.

La transition **Tr-accroit** retire, puis remet la marque de la place Pr-inf modélisant la borne inférieure de la fenêtre de réception. Elle laisse inchangé le marquage des places intervenant dans l'assertion A2.

La transition **Tr-Session** accroît la valeur de la place Pr-inf. Donc, si l'assertion A2 était vérifiée avant le franchissement, l'augmentation de la borne supérieure de l'intervalle par le franchissement de la transition Tr-Session et la modification de la place Pr-inf conservent l'assertion .

Nous venons de prouver que l'assertion A2 est vérifiée à l'état initial, et qu'elle est conservée par le franchissement de toutes les transitions du modèle du protocole Transport. L'assertion **A2** est donc un **invariant** du modèle.

Assertion A3

A3: La valeur de la borne inférieure de la fenêtre de réception est comprise entre la valeur de la borne inférieure de la fenêtre d'émission et le numéro du dernier message à émettre.

$$M(\text{Pr-inf}) \in [\, M(\text{Pe-inf}) \,,\, M(\text{Pe-compt}) \,]$$

Le numéro du prochain message à recevoir au récepteur (M(Pr-inf)) est compris dans l'intervalle formé par le numéro du dernier message acquitté de bout en bout à l'émetteur (M(Pe-inf)) et le dernier message reçu de la couche Session (M(Pe-compt)) .

Preuve :

A **l'état initial**, les marques des places Pe-inf, Pe-compt et Pr-inf sont nulles, donc l'assertion est vérifiée au marquage initial.

Les transitions **Te-donnée, Te-erreur** de la partie émettrice et **Tr-hors, Tr-donnée, Tr-duplic, Tr-erreur, Tr-accroit, Tr-dec-att, Tr-dec-rec, Tr-acq, Tr-rej** de la partie réceptrice n'interviennent pas dans la formation de

l'assertion A3.

Le déclenchement de la transition **Te-Session** accroît la valeur de la marque Pe-compt sans modifier les places Pe-inf et Pr-inf. La valeur de la place Pe-compt étant la borne supérieure de l'intervalle défini par l'assertion A3, cette assertion est trivialement conservée.

M(Te-Session>M' : M'(Pe-compt)=M(Pe-compt)~+~1 , donc

si M(Pr-inf) ∈ [M(Pe-inf) , M(Pe-compt)]

alors M'(Pr-inf) ∈ [M'(Pe-inf) , M'(Pe-compt)] .

Le déclenchement de la transition **Te-acq** (respectivement Te-rej) modélise la réception par la partie émettrice d'un message d'acquittement (de rejet), ce qui met à jour les bornes inférieures et supérieures de la fenêtre d'émission modélisées par les places Pe-inf et Pe-sup.

M(Te-acq ou Te-rej>M' : M'(Réseau)=M(Réseau)-u1+u2 ,

typmess(u1)=ACQ ou REJ et M'(Pe-inf)=noacq(u1) .

D'après l'assertion A2, nous savons que

∀ u ∈ M(Réseau) , noacq(u) ∈ [M(Pe-inf) , M(Pr-inf)] ,

donc nous avons M'(Pe-inf) ∈ [M(Pe-inf) , M(Pr-inf)] .

Nous venons d'apporter la preuve que l'assertion A3 est conservée par les transitions Te-acq et Te-rej.

Le déclenchement de la transition **Tr-Session** incrémente la borne inférieure de la fenêtre de réception modélisée par la place Pr-inf.

M(Tr-Session>M' : M'(Pr-reçu)=M(Pr-reçu) -u avec

nomess(u)=M(Pr-inf) et M'(Pr-inf)=M(Pr-inf) ~+~ 1 .

D'après l'assertion A4 nous savons que

∀ u ∈M(Pr-reçu) , nomess(u) ∈ [M(Pr-inf) , M(Pe-compt) [,

donc nous sommes sûrs que M(Pr-inf) ≠ M(Pe-compt) ,

d'où M'(Pr-inf) ∈ [M'(Pe-inf) , M'(Pe-compt)] ;

L'assertion A3 est vérifiée pour la transition Tr-Session.

Nous venons de prouver la véracité de l'assertion A3 à l'état initial, et sa conservation par le franchissement de toute les transitions du modèle. L'assertion **A3** est donc bien un **invariant.**

Assertion A4

A4: L'ensemble des messages susceptibles d'être stockés dans le tampon de réception est borné par l'intervalle formé par la borne inférieure de la fenêtre d'émission et le numéro du prochain message à transmettre.

$\forall u \in M(Pr\text{-}reçu)$, nomess(u) $\in$ [M(Pr-inf) , M(Pe-compt) [

Preuve :

A **l'état initial**, la place Pr-reçu est vide donc l'assertion A4 est vérifiée par défaut.

Le franchissement de la transition **Te-Session** incrémente le compteur de message modélisé par la place Pe-compt, ce qui a pour conséquence d'accroître l'intervalle défini par l'assertion.
M(Te-Session)M' : M'(Pe-compt)=M(Pe-compt)~+~1 , donc la borne supérieure augmente et l'assertion est facilement conservée.

L'ensemble des **autres transitions** de la partie **émettrice** du modèle Transport ne modifie pas le marquage utile pour l'assertion A4, qui est conservée trivialement.

La transition **Tr-donnée** place le message reçu dans la place Pr-reçu.
M(Pr-donnée>M' : M'(Pr-reçu)=M(Pr-reçu)+u et
M'(Pr-attendu)=M(Pr-attendu)-v avec nomess(u)=v .

D'après le prédicat de franchissement de la transition nous savons que nomess(u) ∈ [M(Pr-inf) , M(Pr-sup) [,et d'aprés l'assertion A1 : nomess(u) ∈ [M(Pe-compt)~-~ f,M(Pe-compt) [, plus les conditions de conservation des messages par le Réseau, nous prouvons que le transition Tr-donnée conserve l'assertion A4.

La transition **Tr-Session** retire un message du tampon de réception modélisé par la place Pr-reçu.
M(Pr-reçu>M' : M'(Tr-Session)=M(Tr-Session) - u , avec
nomess(u)=M(Pr-inf) et M'(Pr-inf)=M(Pr-inf) ~+~ 1 .
Les places Pr-inf et Pr-reçu sont donc modifiées parallèlement :
nomess(M'(Pr-reçu)) = nomess(M(Pr-reçu)) - M(Pr-inf) ,
= [M(Pr-inf) , M(Pe-compt) [- M(Pr-inf) ,
= [M(Pr-inf) ~-~ 1 , M(Pe-compt) [,
= [M'(Pr-inf) , M'(Pe-compt) [,
ce qui prouve que l'assertion A4 est conservée par le franchissement de la transition Tr-Session.

La transition **Tr-dec-rec** supprime un message du tampon de réception modélisé par la place Pr-reçu.
M(Tr-reçu>M' : M'(Pr-reçu) = M(Pr-reçu) -u .
L'ensemble des marques de la place Pr-reçu décroît :
∀ u ∈ M'(Pr-reçu) , u ∈ M(Pr-reçu) , ou encore par hypothèse sur l'assertion A4 avant le franchissement :

u ∈ [M(Pr-inf) , M(Pe-compt) [, et

comme le marquage des places Pr-inf et Pe-compt est conservé :

u ∈ [M'(Pr-inf) , M'(Pe-compt) [.

L'assertion A4 est conservée par le franchissement de la transition Tr-dec-rec.

Le déclenchement de l'ensemble des **autres transitions** de la partie **réceptrice** du modèle du protocole Transport ne modifie pas l'assertion A4.

L'assertion **A4** est donc un **invariant** du modèle, car elle est vérifiée à l'état initial, et est conservée par toutes les transitions.

Assertion A5

A5: Le dernier message reçu de la couche Session est compris dans l'intervalle formé par le numéro du dernier message acquitté par le récepteur et le numéro du dernier message d'acquittement reçu par l'émetteur.

M(Pe-compt) ∈ [M(Pr-inf) , M(Pe-inf) ~+~ f]

Du fait, de la possibilité de recevoir des messages de rejet comportant des valeurs de crédit inférieures aux valeurs précédentes, les messages admis en transmission ne dépendent plus directement de la borne supérieure de la fenêtre, mais de sa largeur maximale.

Preuve :

A **l'état initial,** les marques des trois places Pe-compt, Pe-inf et Pr-inf sont nulles, donc l'assertion est aisément vérifiée.

Les transitions **Te-donnée, Te-erreur, Tr-erreur, Tr-hors, Tr-donnée, Tr-duplic et Tr-rej, Tr-acq** ne modifient pas le marquage des places mises en jeu par l'assertion A5.

La transition **Te-Session** incrémente le compteur de transmission modélisé par la place Pe-compt.
M(Te-Session>M' : M'(Pe-compt)=M(Pe-compt) ~+~ 1 , en laissant inchangé la place Pe-inf. D'après le prédicat, nous savons que:
M(Pe-compt) ∈ [M(Pe-inf) , M(Pe-inf) ~+~ f [, donc après le

franchissement de la transition :

M(Pe-compt) ~+~ 1 ∈ [M(Pe-inf) , M(Pe-inf) ~+~ f ~+~ 1 [, d'où

M'(Pe-compt) ∈ [M'(Pr-inf) , M'(Pe-inf) ~+~ f] .

Le franchissement de la transition Te-Session conserve l'assertion A5.

La transition **Te-acq** (respectivement **Te-rej**) modélise la réception d'un message d'acquittement (de rejet) en modifiant les bornes de la fenêtre d'émission modélisées par les places Pe-inf et Pe-sup. Pour l'une ou l'autre transition :

M(Te-acq ou Te-rej>M' : M'(Réseau)=M(Réseau)-u1+u2 ,

typmess(u1)=ACQ ou REJ et M'(Pe-inf)=noacq(u1) .

D'après l'assertion A2 nous savons que

∀ u ∈ Réseau , noacq(u) ∈ [M(Pe-inf),M(Pr-inf)] , donc

la borne de la fenêtre s'accroît, et l'on conserve

M'(Pe-compt) ∈ [M'(Pr-inf) , M'(Pe-inf) ~+~ f] .

Le franchissement de la transition Te-Acq (ou Te-rej) conserve l'assertion A5.

Au franchissement de la transition **Tr-Session** la borne inférieure de la fenêtre de réception est incrémentée.

M(Tr-Session>M' : M'(Pr-inf)=M(Pr-inf) ~+~ 1 et

M'(Pr-reçu) = M(Pr-reçu) -u avec nomess(u)=M(Pr-inf).

D'après l'assertion A4 nous savons que

∀ u ∈ M(Pr-reçu) , nomess(u) ∈ [M(Pr-inf) , M(Pe-compt) [,

donc que M(Pr-inf) ≠ M(Pe-compt) ,

d'où M'(Pe-compt) ∈ [M'(Pr-inf) , M'(Pe-inf) ~+~ f] .

L'assertion A5 est vérifiée pour la transition Tr-Session.

L'assertion a été vérifiée à l'état initial, puis prouvée après le franchissement de toutes les transitions du modèle, donc l'assertion **A5** est bien un **invariant**.

NOTA : Les invariants suivants mettent en jeu les places Pe-Session et Pr-Session , ainsi que le champ supplémentaire relatif aux messages, qui permettent de mettre en évidence la bonne délivrance des dits messages (Figure C-4a et C-4b). Ces modifications sont motivées au paragraphe suivant.

Assertion A6

A6: Il existe une relation de modulo entre le numéro qui correspond au rang d'émission du message par la couche Session vers la partie émettrice de la couche Transport et le numéro attribué par la couche Transport à ce message.

M(Pe-compt) = M(Pe-Session) modulo N

Preuve:

A **l'état initial**, les deux places Pe-compt et Pe-Session contiennent une marque nulle, l'assertion est donc vérifiée.

L'ensemble des **transitions** du modèle laisse inchangé les places Pe-compt et Pe-Session.

Seule la transition **Te-Session** modifie leur marquage en incrémentant les deux places simultanément.

M(Te-Session>M' avec M'(Pe-Session)=M(Pe-Session)+1 ,et

M'(Pe-compt)=M(Pe-compt) ~+~ 1 .

Les deux compteurs évoluent parallèlement en s'incrémentant .

L'assertion **A6** vient d'être prouvée vraie, à l'état initial, puis conservée par le franchissement des transitions du modèle, c'est donc un **invariant.**

Assertion A7

A7: Il existe une relation de modulo entre le numéro qui correspond au rang d'émission du message par la couche Session et le numéro attribué au message par la couche Transport.

$\forall$ u $\in$ M(Réseau) U M(Pe-fenêtre) U M(Pr-reçu)
si typmess(u)=DONNEE alors nomess(u) = mess(u) modulo N

Preuve :

A **l'état initial**, le réseau et les tampons de stockage d'émission et de réception sont vides de messages significatifs. L'assertion est vérifiée par défaut.

La transition **Te-Session** provoque la constitution d'un message issu de la couche Session.
M(Te-Session>M' : M'(fenêtre)=M(fenêtre) -u1 +u2 avec
nomess(u2)=M(Pe-compt) et mess(u2)=M(Pe-Session).
D'après l'assertion précédente A6, la relation qui lie les deux places, lie aussi le nouvel uplet, d'où
nomess(u2)=mess(u2) modulo N .
L'assertion est conservée par le franchissement de la transition Te-Session.

Du fait de l'adjonction du nouveau champ dans les messages traités par notre modèle, de manière indépendante aux prédicats des autres transitions, l'assertion est facilement conservée par l'ensemble de ces autres transitions.

Les transitions **Te-acq, Te-rej, Te-erreur, Tr-accroit, Tr-dec-att, Tr-dec-rec, Tr-acq, Tr-rej** ne modifient pas le marquage relatif à l'assertion A7, qui est trivialement conservée.

La transition **Te-donnée** manipule les uplets relatifs à l'assertion, mais elle

ne modifie pas le marquage de la place Pe-fenêtre et ajoute dans le réseau un message, où par hypothèse l'association est respectée.

D'après les hypothèses d'utilisation du modèle Réseau, il transfert de manière transparente les messages qui lui sont confiés, donc l'association est maintenue par le réseau.

Les transitions **Tr-erreur, Tr-hors, Tr-donnée, Tr-duplic** modifient le marquage relatif à l'assertion, en retirant un message du réseau, mais conservent à tout moment l'association (nomess/mess) des uplets traités.

La transition **Tr-Session** qui supprime une marque de la place Pr-reçu vérifie par défaut l'assertion A7.

L'assertion **A7** vient d'être vérifiée à l'état initial, puis conservée par le franchissement de l'ensemble des transitions du modèle, c'est donc un **invariant**.

Assertion A8

A8: Il existe une relation de modulo entre le numéro qui correspond au rang de réception du message par la couche Session et le numéro de la borne inférieure de la fenêtre de réception de la couche Transport.

M(Pr-inf) = M(Pr-Session) modulo N

Preuve :

La preuve est identique à celle apportée pour démontrer l'assertion **A6**. A **l'état initial**, les deux places Pr-inf et Pr-Session contiennent une marque nulle, l'assertion est donc vérifiée.

L'**ensemble des transitions** du modèle laisse inchangé les places Pr-inf

et Pr-Session.

Seule la transition **Tr-Session** modifie leur marquage en incrémentant les deux places simultanément.

M(Tr-Session>M' avec M'(Pr-Session)=M(Pr-Session)+1 ,et

M'(Pr-inf)=M(Pr-inf) ~+~ 1 .

Les deux compteurs évoluent parallèlement en s'incrémentant .

L'assertion **A8** vient d'être prouvée vraie, à l'état initial, puis conservée par le franchissement des transitions du modèle, c'est donc un **invariant.**

Assertion A9

A9: Le numéro qui correspond au rang du dernier message émis par la couche Session est compris dans l'intervalle formé par le numéro qui correspond au rang du dernier message reçu par la couche Session et ce même nombre augmenté de la largeur de la fenêtre.

M(Pe-Session) ∈ [M(Pr-Session) , M(Pr-Session) + f]

Preuve :

A **l'état initial**, les deux places Pe-Session et Pr-Session possèdent deux marques de valeurs nulles, ce qui vérifie aisément l'assertion A9.

La transition **Te-Session** incrémente la marque de la place Pe-Session, pour modéliser la transmission d'un nouveau message en provenance de la couche Session.

M(Te-Session>M' : M'(Pe-Session)=M(Pe-Session) + 1 , et

M'(Pe-compt)=M(Pe-compt) ~+~ 1 .

- Soit M(Pe-Session) ∈ [M(Pr-Session),M(Pr-Session)+f [,alors

M'(Pe-Session)=M(Pe-Session)+1 ∈ [M(Pr-Session),M(Pr-Session)+f+1[

d'ou M'(Pe-Session)) ∈ [M'(Pr-Session),M'(Pr-Session)+f] .

- Soit M(Pe-Session) = M(Pr-Session) + f , alors
d'après l'assertion A6 : Pe-compt=Pe-Session modulo N et
d'après l'assertion A8 : Pr-inf = Pr-Session modulo N ce qui
donne : Pr-inf~+~f = Pe-compt (a).
D'après l'assertion A3 , M(Pr-inf) ∈ [M(Pe-inf),M(Pe-compt)]

,ce qui produit M(Pr-inf)~+~f ∈ [M(Pe-inf)~+~f,M(Pe-compt)~+~f]

avec (a) : M(Pe-compt) ∈ [M(Pe-inf)~+~f,M(Pe-compt)~+~f] (b).

D'après l'assertion A5 , M(Pe-compt) ∈[M(Pr-inf),M(Pe-inf)~+~f]
joint à (b) nous obtenons finalement :
M(Pe-compt) = M(Pe-inf)~+~ f , ce qui est en contradiction avec le prédicat de la transition Te-Session, qui n'a pu être franchie. Cette démonstration par l'absurde prouve que l'on ne peut avoir M(Pe-Session) = M(Pr-Session) + f. La transition Te-Session conserve donc l'assertion A9.

La transition **Tr-Session** délivre à la couche Session un message, ce qui est modélisé par l'incrémentation de la place Pr-Session.
M(Tr-Session>M' : M'(Pr-Session)=M(Pr-Session) + 1 , et
M'(Pr-inf)=M(Pr-inf) ~+~ 1 , M'(Pr-reçu)=M(Pr-reçu) -u avec
nomess(u)=M(Pr-inf).

- Soit M(Pe-Session) ∈] M(Pr-Session),M(Pr-Session)+f] ,alors

M'(Pe-Session) ∈ [M(Pr-Session)+1,M(Pr-Session)+f+1]

d'ou M'(Pe-Session)) ∈ [M'(Pr-Session),M'(Pr-Session)+f] .

- Soit M(Pe-Session) = M(Pr-Session) , alors d'après les
assertions A6 : M(Pe-compt) = M(Pe-Session) modulo N et

A8 : M(Pr-inf) = M(Pr-Session) modulo N , nous obtenons l'égalité suivante : M(Pr-inf) = M(Pe-compt) (c).

D'après l'assertion A4 : $\forall$ u $\in$ (Pr-reçu) ,

u $\in$ [M(Pr-inf) , M(Pe-compt) [, ce qui donne avec (c) :

u $\in$ [M(Pr-inf) , M(Pr-inf) [. La place Pr-reçu est donc vide, et donc la transition Tr-Session non-franchissable dans les conditions de l'hypothèse.

Pour l'ensemble des **autres transitions** le marquage relatif aux places n'influence pas l'assertion A9.

L'assertion **A9** est vraie à l'état initial, et étant vraie, elle est conservée par le franchissement de toutes les transitions du modèle, c'est un **invariant.**

Assertion A10

A10: Le rang des messages susceptibles d'être reçus par le récepteur de la couche Transport est compris entre le rang du dernier message reçu par la couche Session et le rang du dernier message émis par la couche Session.

$\forall$ u $\in$ M(Pr-reçu), mess(u) $\in$ [M(Pr-Session),M(Pe-Session) [

Preuve :

D'après l'assertion A4, nous savons que :

$\forall$ u $\in$ M(Pr-reçu), nomess(u) $\in$ [M(Pr-inf) , M(Pe-compt) [,

que nous pouvons aussi écrire , pour k $\in$ N :

nomess(u)~+~k $\in$ [M(Pr-inf~+~k , M(Pe-compt)~+~k [, or

d'après les assertions A6, A7, A8 :

M(Pr-inf) = M(Pr-Session) modulo N ;

nomess(u) = mess(u) modulo N ;

M(Pe-compt) = M(Pe-Session) modulo N ;
nous pouvons le réécrire :

mess(u) ∈ [M(Pr-Session),M(Pe-Session) [,
ce qui démontre l'assertion A10.

Assertion A12

A12: Les messages d'acquittements circulant sur une connexion Transport forment une suite croissante avec leur numéro de message acquitté.

∀ ui,uj ∈ M(Réseau) ,
tel que type(ui)=ACQ ou REJ et type(uj)=ACQ ou REJ
si noempl(ui)>noempl(uj) alors noacq(uj) ∈ [noacq(ui),M(Pr-inf)]

Soient deux messages d'acquittements ui et uj, le plus près du récepteur (de numéro d'emplacement supérieur) a un numéro d'acquittement compris entre le numéro d'acquittement de l'autre message. Et de plus, il est borné par la marque de la place Pr-inf.

Preuve :

La transition **Te-acq** (respectivement **Te-rej**) reçoit un message d'acquittement et l'enlève du Réseau.
M(Te-acq ou Te-rej>M' : M'(Réseau)=M(Réseau)-u , avec
noempl(u)=NBEMPL .
Par hypothèse, toutes les marques restant dans le modèle réseau, respectent l'assertion A12 avant le franchissement de la transition, et la respectent après. La marque u, supprimée du Réseau, vérifie l'assertion par défaut après le franchissement.

Les **autres transitions** de la partie émettrice du protocole de Transport

n'interviennent pas dans la preuve de l'assertion A12.

Le modèle Réseau possédant une propriété de séquentialité conserve naturellement l'assertion A12.

La transition **Tr-Session** incrémente le marquage de la place Pr-inf.
M(Tr-Session>M' : M'(Pr-inf)=M(Pr-inf) ~+~ 1 .
Cette transition augmente la borne supérieure de l'intervalle défini par l'assertion A12 , qui est donc conservée aisément.

La transition **Tr-acq** (respectivement Tr-rej) émet un message d'acquittement sur la connexion Transport.
M(Tr-acq ou Tr-rej>M' : M'(Réseau)=M(Réseau) + u , avec
noempl(u)=1 et noacq(u)=M(Pr-inf).
Pour ui≠u et uj≠u , la transition ne modifie pas ces deux uplets et l'assertion demeure vérifiée.
Pour ui≠u et uj=u , comme d'après le prédicat noacq(u)=M(Pr-inf), l'assertion A12 est vérifiée.
Pour ui=u, comme noempl(u)=1, il n'existe pas de uj, l'assertion est vérifiée par défaut.

Les **autres transitions** de la partie réceptrice du protocole de Transport n'interviennent pas dans la preuve de l'assertion A12.

A l'état initial, puis pour le franchissement de toutes les transitions du modèle l'assertion **A12** a été prouvée, c'est donc un **invariant** du modèle du protocole Transport.

Assertion A13

A13: L'ensemble des numéros de messages de données attendus (Pr-attendu) ou reçus (Pr-reçu) par la partie réceptrice du protocole Transport mais

non-délivrés à la couche Session forme un intervalle qui correspond exactement à la fenêtre de réception (Pr-sup, Pr-inf).

M(Pr-attendu) U nomess(M(Pr-reçu)) = [M(Pr-inf) , M(Pr-sup) [

Preuve :

A l'état initial, la place Pr-sup contient une marque de valeur "c", qui modélise le crédit accordé au moment de l'établissement de la connexion Transport, la place Pr-inf contient une marque nulle, qui modélise la borne inférieure de la fenêtre de réception. La place Pr-attendu est bien initialisée par l'intervalle formé par l'ensemble des numéros de messages en attente de réception, la place Pr-reçu étant vide (aucun message de donnée n'a été encore transmis). Ces conditions respectent bien l'assertion A13.

Les transitions de la partie émettrice du modèle **Te-donnée, Te-Session, Te-erreur, Te-acq, Te-rej**, et celles de la partie réceptrice **Tr-duplic, Tr-hors, Tr-erreur, Tr-acq, Tr-rej** ne modifient en rien le marquage des places Pr-attendu, Pr-reçu, Pr-sup et Pr-inf utilisé pour l'assertion A13. Cette assertion est donc conservée trivialement pour l'ensemble de ces transitions.

La transition **Tr-donnée** modélise la réception d'un message de donnée, en déplaçant une marque de la place Pr-attendu vers la place Pr-reçu.
M(Te-donnée>M' : M'(Pr-attendu)=M(Pr-attendu) -u1 ,et
M'(Pr-reçu)=M(Pr-reçu) +u2 avec nomess(u2)=u1 .
Donc l'ensemble des marques contenues par les places Pr-attendu et Pr-reçu est globalement inchangé. L'assertion A13 est conservée par la transition Te-donnée.

La transition **Tr-accroit** accorde un crédit supplémentaire en incrémentant la borne supérieure de la fenêtre de réception, modélisée par la place Pr-sup.
M(Tr-accroit>M' : M'(Pr-sup)=M(Pr-sup)~+~1 , et
M'(Pr-attendu)=M(Pr-attendu) + u avec nomess(u)=M(Pr-sup) .
D'après le franchissement de la transition, on a :

nomess(M'(Pr-attendu) U M'(Pr-reçu))=M(Pr-attendu)+M(Pr-reçu)+u ,
ce qui donne par hypothèse : =[M(Pr-inf) , M(Pr-sup)[+ M(Pr-sup),
donc = [M(Pr-inf) , M(Pr-sup)] ,
= [M(Pr-inf) , M(Pr-sup) ~+~ 1[
et avec le marquage M' : = [M'(Pr-inf) , M'(Pr-sup)[.
La transition Tr-accroit conserve l'assertion A13.

La transition **Tr-dec-att** (respectivement **Tr-dec-rec**) provoque la diminution du crédit, en décrémentant la borne supérieure de la fenêtre de réception modélisée par la place Pr-sup.
M(Tr-dec-xx>M' : M'(Pr-sup)=M(Pr-sup)~-~1 , et
M'(Pr-attendu)=M(Pr-attendu) - u avec nomess(u)=M'(Pr-sup)
ou (respectivement M'(Pr-reçu)=M(Pr-reçu) - u).
D'après le franchissement de la transition, on a :
nomess(M'(Pr-attendu) U M'(Pr-reçu))=M(Pr-attendu)+M(Pr-reçu)-u ,
ce qui donne par hypothèse : = [M(Pr-inf) , M(Pr-sup)[- M'(Pr-sup),
donc = [M(Pr-inf) , M(Pr-sup)~-~1]- M'(Pr-sup),
= [M(Pr-inf) , M'(Pr-sup)]- M'(Pr-sup),
et avec le marquage M' : = [M'(Pr-inf) , M'(Pr-sup)[.
Les transitions Tr-dec-att et Tr-dec-rec conservent bien l'assertion A13.

La transition **Tr-Session** modélise la remise à la couche Session d'un message, en enlevant simultanément le message de la place Pr-reçu et en incrémentant la borne inférieure de la fenêtre de réception modélisée par la place Pr-inf.
M(Tr-Session>M' : M'(Pr-inf)=M(Pr-inf)~+~1 , et
M'(Pr-reçu)=M(Pr-reçu) - u avec nomess(u) = M(Pr-inf) .
D'après le franchissement de la transition, on a :
nomess(M'(Pr-attendu) U M'(Pr-reçu))=M(Pr-attendu)+M(Pr-reçu)-u ,
ce qui donne par hypothèse : =[M(Pr-inf) , M(Pr-sup)[- M(Pr-inf),
donc =]M(Pr-inf) , M(Pr-sup)[,
= [M(Pr-inf)~-~1, M(Pr-sup)[
et avec le marquage M' : =[M'(Pr-inf) , M'(Pr-sup)[.

La transition Tr-Session conserve l'assertion A13.

A l'état initial, puis pour le franchissement de toutes les transitions du modèle l'assertion **A13** a été prouvée, c'est donc un **invariant** du modèle du protocole Transport.

Assertion A14

A14 : Les bornes supérieures des fenêtres d'émission (Pe-sup) et de réception (Pr-sup) sont respectivement comprises dans les intervalles formés par les bornes inférieures des fenêtres d'émission (Pe-inf) et de réception (Pr-inf) et ces mêmes valeurs augmentées de la taille maximale de la fenêtre (f).

A14.1 : M(Pe-sup) ∈ [M(Pe-inf) , M(Pe-inf) ~+~ f [

A14.2 : M(Pr-sup) ∈ [M(Pr-inf) , M(Pr-inf) ~+~ f [

Preuve :

A l'état initial, les deux assertions sont vérifiées.

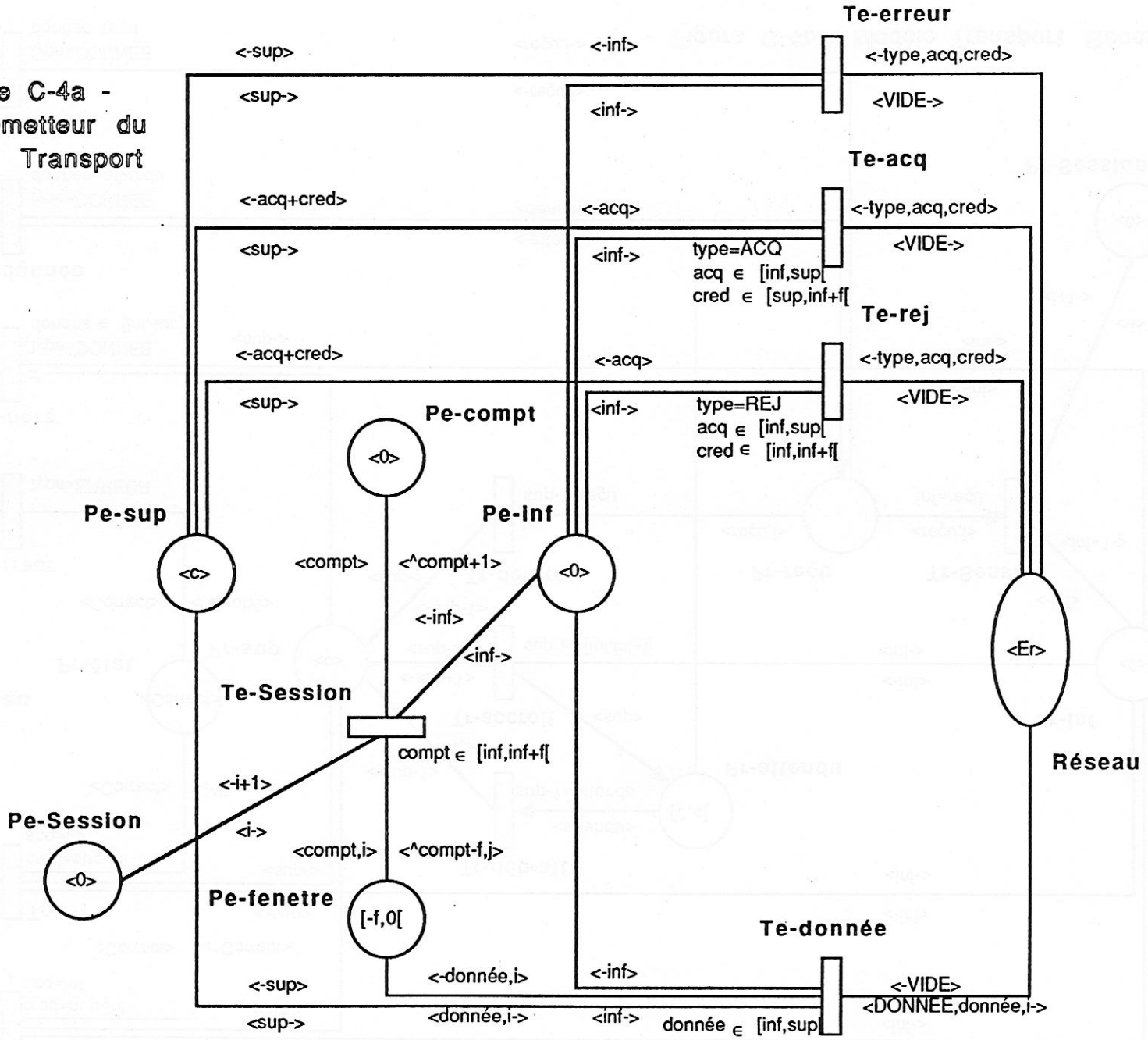
Les transitions **Te-acq et Te-rej** qui modifient la valeur de la place Pe-sup et Pe-inf assurent par leurs prédicats que la propriété du lemme 14.1 est vérifiée.

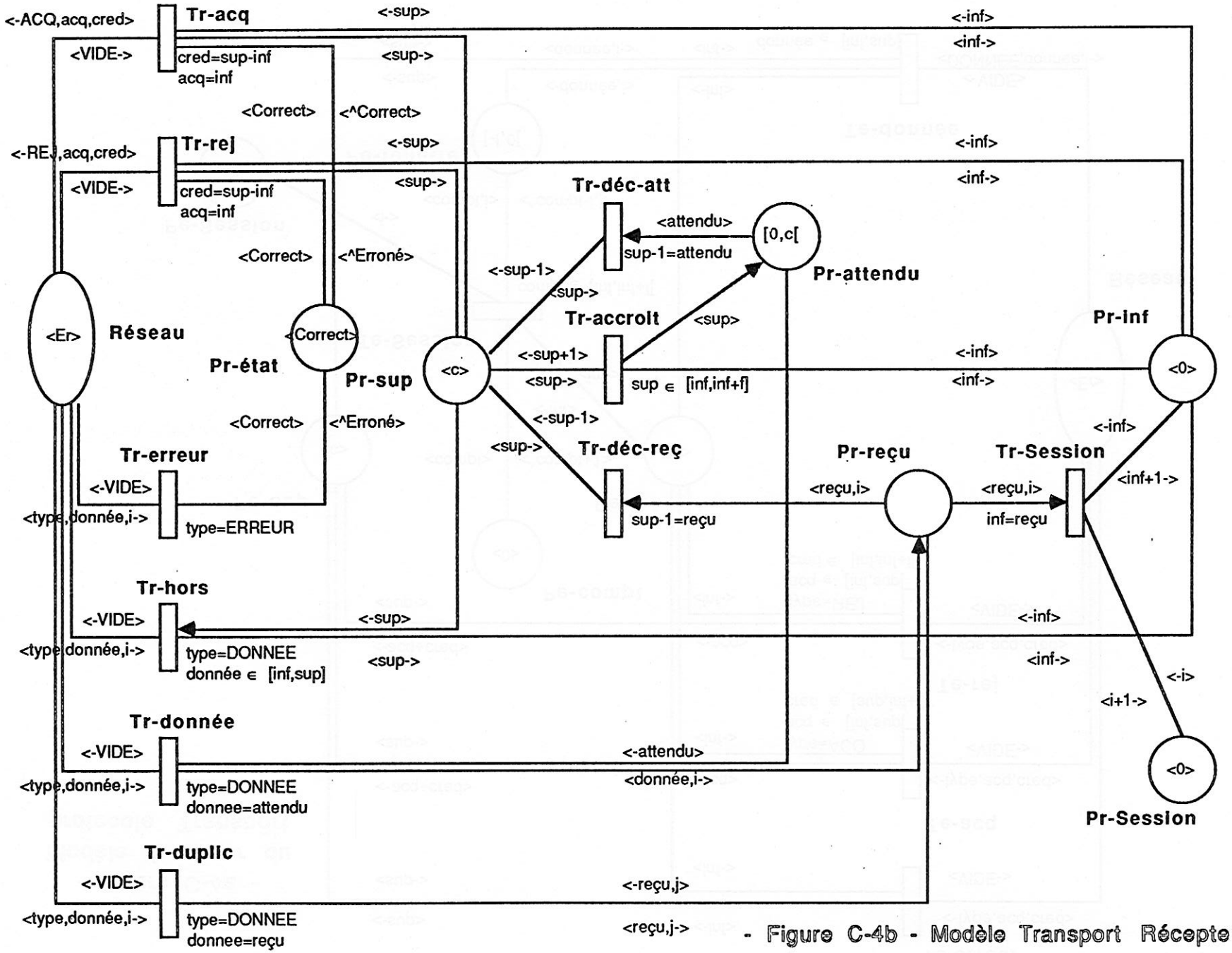
De même les transitions **Tr-dec-xx, Tr-accroit et Tr-Session** incrémentent (ou décrémentent) modulo N les places Pr-inf et Pr-sup. Les prédicats de chacune des transitions et l'assertion A13 assurent la conservation de la propriété du lemme 14.2.

Les **autres transitions** du modèle ne modifient pas le marquages des places Pr-inf, Pr-sup et Pe-inf, Pe-sup.

Les assertions **A14.1** et **A14.2** sont dons des **invariants** du modèle.

- Figure C-4a -
Modèle émetteur du protocole Transport





- Figure C-4b - Modèle Transport Récepteur

4.4 L'Adéquation de service

4.4.1 La séquentialité

Nous allons prouver que le protocole Transport délivre séquentiellement au récepteur les messages qui lui sont transmis par l'émetteur (Propriété T1 de la phase de spécification).

Pour faciliter cette preuve nous allons adjoindre au modèle du protocole Transport deux places. La première place Pe-Session modélise la couche supérieure (appelé Session) côté émetteur, elle est connectée à la transition d'interface Te-Session. La deuxième place Pr-Session modélise la couche Session côté récepteur, elle est connectée à la transition Tr-Session. Ces deux places fonctionnent en fait comme des compteurs constamment croissants, identifiant de manière unique les messages émis ou reçus franchissant Transport/Session (Figure C-4a et Figure C-4b).

C'est ainsi que les marques modélisant les messages véhiculés par le protocole Transport doivent comporter un champ supplémentaire. Ce champ modélise le contenu du message et permet de l'identifier, sa valeur lui est attribuée au franchissement de la transition Te-Session.

Ces adjonctions ne modifient pas le comportement du modèle initial, car elles respectent les conditions d'équivalence de comportement définies au chapitre 6 de la première partie, aucun prédicat du réseau initial ne référence le nouveau champ. De plus les deux places Pe-Session et Pr-Session sont non-contraignantes pour le réseau.

Nous définissons la propriété de séquentialité de la manière suivante:

Il faut que l'ordre de réception des messages à l'interface récepteur Transport/Session soit l'ordre d'émission de ces mêmes messages à l'interface émetteur Session/Transport, sans omission ni duplication. Dans le modèle, la

transition Te-Session (modélisant l'interface Session/Transport) attribue un numéro croissant à chaque message qui la franchit, il est alors facile de contrôler que les messages franchissant la transition Tr-Session (modélisant l'interface Transport/Session) ont un numéro qui suit cet ordre strictement croissant.

Nous exprimons la propriété T1 ainsi:
si M(Tr-Session>M' et M'=M-u+u' alors mess(u)=M(Pr-Session).

Nous allons maintenant prouver cette propriété :

Nous savons par l'assertion A10 que :
si M(Tr-Session>M' et M'=M-u+u' alors
mess(u) ∈ [M(Pr-Session),M(Pe-Session) [

et par l'assertion A9 que :
M(Pe-Session) ∈ [M(Pr-Session, M(Pr-Session) ~+~f [

nous en déduisons que (a) :
si M(Tr-Session>M' et M'=M-u+u' alors
mess(u) ∈ [M(Pr-Session),M(Pr-Session) ~+~f [

De plus, le prédicat conditionnant le franchissement de la transition Tr-Session dit (b): nomess(u) = M(Pr-inf)

or d'après l'assertion A7 on sait que :
∀ u ∈ M(Réseau) U M(Pr-reçu) U M(Pe-fenêtre)
si typaquet(u)=DONNEE alors nomess(u)=mess(u) modulo N

que l'on peut exprimer par (c) :
∃ q ∈ N tel que nomess(u) + q*N = mess(u)

et d'après l'assertion A8 :
M(Pr-inf) = M(Pr-Session) modulo N

ce que l'on peut réécrire (d) :

$\exists p \in N$ tel que M(Pr-inf) + p*N = M(Pr-Session)

si nous réunissons (b),(c),(d) , on obtient (e) :
$\exists k \in Z$ tel que mess(u) = k*N + M(Pr-Session)

Si on rapproche (a) et (e) , pour que les deux équations soient vérifiées, il faut impérativement que k=1 d'où:
mess(u) = M(Pr-Session)

Nous venons d'établir la preuve que les marques franchissent la transition Tr-Session dans un ordre rigoureusement identique à leur franchissement de la transition Te-Session.

Il est important de remarquer que cette propriété ne prouve pas que tous les messages sont délivrés, mais elle prouve que si un message (une marque) est délivré au destinataire (franchit la transition Tr-Session) alors il respecte les contraintes de séquentialité.

Cette preuve établit l'adéquation de service de la **propriété T1** définie pendant la phase de spécification.

4.4.2 La Vivacité

Nous allons prouver la vivacité de notre modèle, preuve que notre modèle ne se bloque pas, donc que le protocole reste dans un état cohérent (Propriété T2 de la phase de spécification). Cette preuve est établie en deux temps :
- dans un premier temps, nous prouvons que le modèle possède un ensemble d'états d'accueils ET ;

- puis nous prouvons, dans un deuxième temps, que le modèle est quasi-vivant à partir d'un état de ET, et enfin, qu'un des états de ET est accessible à partir du marquage initial Mo.

Nous définissons l'état du modèle par le marquage de l'ensemble de ses places, que nous regroupons en plusieurs sous-ensembles :

L'ensemble des places modélisant la partie émettrice du protocole Transport :

E = < Pe-compt ; Pe-inf ; Pe-sup ; Pe-fenêtre >

L'ensemble des places modélisant la partie réceptrice du protocole Transport :

R = < Pr-état ; Pr-inf ; Pr-sup ; Pr-attendu ; Pr-reçu >

L'ensemble des places modélisant le service de la couche Réseau, ce modèle a été étudié au cours du chapitre précédent, notamment nous avons prouvé qu'il est vivant et qu'il possède un ensemble propre d'état d'accueil ER.

Nous allons maintenant prouver que quelque soit l'état atteint par le modèle du protocole Transport, il est toujours possible de trouver une séquence de franchissements de transitions qui permette d'arriver à un des états de l'ensemble d'accueil ET. Nous allons utiliser à nouveau une norme D, qui permettra de mesurer la distance séparant un état quelconque M de l'ensemble d'accueil ET.

On sait que D est une norme pour l'état d'accueil ET si et seulement si , pour tout marquage M appartenant à l'ensemble des marquages accessibles A du modèle Transport :
- si le marquage M est élément de l'ensemble ET ,il faut alors que la norme de M soit nulle ($\forall M \in ET : D(M) = 0$) ;
- sinon il existe une séquence de franchissement qui permet de faire décroître la

norme ($\forall$ M $\notin$ ET : $\exists$ s $\in$ T* tel que M(s>M' et D(M) > D(M') .

Lemme 20 : Etat d'accueil du Réseau

Nous partons d'un état quelconque du modèle Transport, et nous plaçons le sous-modèle Réseau dans son propre ensemble d'état d'accueil ER que nous savons exister. Cela consiste à vider le réseau de l'ensemble des messages qui y circulent en franchissant les transitions formant l'interface Réseau/Transport. Le service Réseau assurant un transport bidirectionnel des messages, il faut que les transitions des deux parties émettrice et réceptrice du protocole Transport soient franchissables. En fait cette franchissabilité est assurée car l'union des prédicats des transitions de la partie émettrice forme une tautologie, il existe donc à tout moment au moins une transition franchissable. Il en est de même pour la partie réceptrice.

Marquage des places

Les assertions A1 et A13 indiquent clairement que le marquage de la place Pe-fenêtre et des places Pr-attendu et Pr-reçu sont déterminés ou dépendent respectivement du marquage de la place Pe-compt et des places Pr-inf et Pr-sup.

La construction du modèle et la sémantique issue du fonctionnement du protocole Transport assure que les places Pe-compt, Pe-inf, Pe-sup et Pr-inf, Pr-sup ne possèdent quelque soit l'état du modèle qu'une seule marque prenant valeur dans l'intervalle [0,N [.

La place Tr-erreur ne contient qu'une marque, modélisant l'état du protocole Transport, qui peut avoir une des deux valeurs CORRECT ou ERRONE.

Un état quelconque du modèle Transport peut donc être caractérisé par le marquage suivant :
Le modèle réseau est dans son état d'accueil ER , la partie émettrice dans l'état :

E = < e1 , e2 , e3 , [e1 ~-~ f , e1 [> ,

La partie réceptrice dans l'état : R = < r1 , r2 , r3 , R4 , R5 > ,avec e1,e2,e3,r2,r3 ∈ [0,N[et r1 ∈ {CORRECT,ERRONE} et R4 U R5 = [r2 , r3 [.

Nous déclenchons la transition Tr-acq de la partie réceptrice du protocole de Transport, ce qui a pour conséquence d'émettre une marque sur le Réseau qui a été préalablement vidé et synchronisé. Cette marque est la modélisation d'un message d'acquittement qui possède deux champs : le numéro d'acquittement et la valeur du crédit.

Le message est traité par le modèle Réseau comme tout autre message, le modèle étant vivant toutes ses transitions sont franchissables, le message d'acquittement peut donc est délivré à la partie émettrice de la couche Transport par le franchissement de la transition Te-acq, le prédicat étant vérifié par l'assertion A2.

En résumant les actions engendrées par cette étape, nous avons par le franchissement de la transition Tr-acq :
M (Tr-acq > M' avec M' = M + u et typmess(u) = ACQ ,
noacq(u)=M(Pr-inf) , crédit(u)= (M(Pr-sup) ~-~ M(Pr-inf)) ;
Par le franchissement de la transition Te-acq :
M' (Te-acq > M" avec M" = M' - u et typmess(u) = ACQ ,
M"(Pe-inf)=noacq(u), M"(Pe-sup) = crédit(u) ~-~ noacq(u)) ;
Nous obtenons les valeurs suivantes pour les places modifiées :
M"(Pe-inf) = M"(Pr-inf) et M"(Pe-sup) = M"(Pr-sup) .

L'état du modèle Transport est caractérisé maintenant par le marquage suivant :
Le modèle réseau est dans son état d'accueil ER , la partie émettrice dans l'état E = < e1 , r2 , r3 , [e1 ~-~ f , e1 [> ,
La partie réceptrice dans l'état R = < r1 , r2 , r3 , R4 , R5 > avec
e1,r2,r3 ∈ [0,N[et r1 ∈ {CORRECT,ERRONE} et R4 U R5 = [r2 , r3 [.

Nous allons maintenant répéter une séquence de franchissement pour permettre d'établir une égalité de marquage des places Pr-inf et Pr-sup. Cela est possible par la séquence suivante :

Tant que M(Pr-sup) ≠ M(Pr-inf) faire

Franchir une des deux transitions Tr-dec-att ou Tr-dec-rec, ce qui est possible d'après l'assertion A13 qui assure qu'il existe dans une des deux places Pr-reçu ou Pr-attendu une marque u tel que nomess(u) = M(Pr-sup).

Le franchissement de la transition diminue le marquage de la transition Pr-sup : si M (Tr-dec-xx > M' alors M'(Pr-sup) = M(Pr-sup) ~-~ 1 .

Fin faire

L'assertion A14.2 décrivant le marquage de la place Pr-sup prouve que la décrémentation atteindra inévitablement l'égalité avec le marquage de la place Pr-inf (M(Pr-sup) = M(Pr-inf)) .

Nous aboutissons dans l'état du modèle Transport suivant :

Le modèle réseau est dans son état d'accueil ER ,

La partie émettrice dans l'état E = < e1 , n , n , [e1~-~f , e1[> ,

La partie réceptrice dans l'état R = < état , n , n , Ø , Ø > avec e1,n ∈ [0,N[et état ∈ {CORRECT,ERRONE}.

Le marquage de la place Pr-état est soit CORRECT soit ERRONE, dans le premier cas nous allons franchir la transition Tr-acq et émettre un message d'acquittement comme précédemment, dans le deuxième cas nous franchissons la transition Tr-rej pour placer le protocole dans un état correct et émettre un message de rejet. Du fait des bonnes propriétés du modèle du service Réseau (il est vivant) le message peut être délivré à l'autre extrémité par le franchissement soit de la transition Te-acq ou Te-rej. Ce qui a pour conséquence, comme pour l'étape précédente d'assurer l'égalité des marquages des quatre places :

m(Pe-inf)=M(Pe-sup)=M(Pr-inf)=M(Pr-sup) et M(Pr-état)=CORRECT .

Nous aboutissons dans l'état du modèle Transport suivant :
Le modèle réseau est dans son état d'accueil ER ,
La partie émettrice dans l'état E = < m , n , n , [m ~-~ f,m[> ,
La partie réceptrice dans l'état R = < CORRECT , n , n , Ø , Ø >
avec m,n ∈ [0,N[.

Nous allons maintenant exécuter une suite de franchissements de transitions pour permettre d'égaliser le marquage des places Pe-compt et Pe-inf. Le franchissement des transitions Tr-accroit, Tr-acq, Te-acq, Te-donnée, Tr-donnée, Tr-Session a pour conséquence d'incrémenter respectivement les marques des places Pr-sup, Pe-sup et Pe-inf, Pr-inf jusqu'à la valeur du compteur Pe-compt. L'assertion A5 nous assure la réussite de cette manoeuvre.

Tant que M(Pe-compt) ≠ M(Pe-inf) faire
- Franchir la transition Tr-accroit, qui provoque l'accroissement du marquage de la place Pr-sup, et place une marque dans la place Pr-attendu :
M(Tr-accroit>M' avec M'(Pr-sup)=M(Pr-sup)~+~1 (ou =n~+~1) ,
M'(Pr-attendu) = M(Pr-sup) (ou =n) .
- Franchir la transition Tr-acq, qui émet un message d'acquittement qui comporte la nouvelle valeur de la borne supérieure de la fenêtre en direction de l'autre extrémité du protocole Transport :
M(Tr-acq>M' avec M'(Réseau)=M(Réseau)+u et acq(u)=M(Pr-inf) (ou =n) , cred(u)=M(Pr-sup)~-~M(Pr-inf) (ou =1) et typmess(u)=ACQ.
- Ce message est véhiculé par le service réseau jusqu'à son destinataire.
- Franchir la transition Te-acq, qui réceptionne le message d'acquittement précédemment émis et met à jour la nouvelle valeur du crédit :

M(Tr-acq>M' avec M'(Réseau)=M(Réseau)-u ; le prédicat est vérifié typmess(u)=ACQ et M'(Pe-sup)=acq(u)~+~cred(u) (ou=n~+~1)

- Franchir la transition Te-donnée, qui émet un message. La fenêtre d'émission le permet, car un nouveau crédit vient d'être délivré et par définition de l'assertion A1, la place Pe-fenêtre contient le prochain message à émettre :

M(Te-donnée>M' avec M'(Réseau>=M(Réseau)+u , nomess(u)=n.

- Ce message est véhiculé par le service Réseau jusqu'à son destinataire.

- Franchir la transition Tr-attendu, qui réceptionne le message de donnée si celui-ci est dans la fenêtre de réception. Cette transition est franchissable, car nous savons que :

M(Tr-attendu>M' avec M'(Réseau)=M(Réseau)-u et M(Pr-attendu)=nomess(u) (ou =n) et M'(Pr-reçu)=nomess(u) (ou=n).

- Franchir la transition Tr-Session avec la marque de la place Tr-reçu qui modélise le message de donnée qui vient d'arriver :

M(Tr-Session)>M' avec M'(Pr-inf)=M(Pr-reçu) (ou =n).

Fin faire

Nous aboutissons dans l'état du modèle Transport suivant :

Le modèle réseau est dans son état d'accueil ER ,

La partie émettrice dans l'état E = < n , n , n , [n ~-~ f,n[> ,

La partie réceptrice dans l'état R = < CORRECT , n , n , Ø , Ø >

avec n ∈ [0,N[.

Pour que les compteurs atteignent la valeur nulle, il faut réitérer la démarche précédente pour incrémenter modulo N l'ensemble des places Pe-compt, Pe-inf, Pe-sup, Pr-inf, Pr-sup, en franchissant successivement l'ensemble des transitions

définies précédemment plus la transition Te-Session. Les conditions de franchissement sont les mêmes qu'à l'étape précédente, c'est pourquoi nous ne les répétons pas.

Tant que M(Pe-compt) ≠ 0 Faire
- Franchir la transition Te-Session (M(Pe-compt) ~+~ 1);
- Franchir la transition Tr-accroit (M(Pr-sup) ~+~ 1);
- Franchir la transition Tr-acq ;
- Franchir la transition Te-acq (M(Pe-sup),M(Pe-inf) ~+~ 1) ;
- Franchir la transition Te-donnée ;
- Franchir la transition Tr-donnée ;
- Franchir la transition Tr-Session (M(Pr-inf) ~+~ 1) ;

Fin faire

Nous obtenons finalement l'état d'accueil du modèle Transport, qui se caractérise par les marquages suivants :
Le modèle réseau est dans son état d'accueil ER ,
La partie émettrice dans l'état E = < 0 , 0 , 0 , [0 ~-~ f , 0 [> ,
La partie réceptrice dans l'état R = < CORRECT , 0, 0 , Ø , Ø > .

Le **lemme 20** prouve donc que notre modèle du protocole de la couche Transport en phase de transfert de données possède un **état d'accueil ET**.

Nous venons de prouver qu'à partir d'un état quelconque du modèle, il est toujours possible d'atteindre l'état d'accueil ET. Nous prouvons maintenant, qu'à partir de l'état initial Mo, il est possible de gagner un état de l'ensemble d'accueil ET.

L'état initial M est caractérisé par :
Le réseau est dans un état synchrone et vide, le modèle du service Réseau est dans son état propre d'accueil ER;
Aucun message n'a encore été émis par le protocole Transport, seule l'attribution d'un crédit a pu être décidée par la phase de connexion, ce qui remplit la place

Pr-attendu de l'ensemble des messages attendus :
La partie émettrice est donc dans l'état E = <0,0,c,[~-~f,0[> et
la partie réceptrice est dans l'état R = <CORRECT,0,c,[0,c[, Ø > .

Pour atteindre l'état ET, il suffit de réduire le crédit alloué.

Tant que M(Pr-sup) ≠ 0 Faire
- Franchir la transition Tr-dec-att, ce qui est toujours possible tant que M(Pr-sup)>=M(Pr-inf) (ou ≠0) et par hypothèse sur le marquage de la place Pr-attendu, on obtient:
M(Tr-dec-att)M' avec M'(Pr-attendu)=M(Pr-attendu)-u et
M'(Pr-sup)=M(Pr-sup)~-~1 et u=M'(Pr-sup) .

Fin faire

Puis communiquer ce changement de crédit par l'émission d'un message de rejet en franchissant la transition Tr-rej.
M(Tr-rej>M' avec M'(Réseau)=M(Réseau)+u tel que noacq(u)=M(Pr-inf) (ou =0) et cred(u)=M(Pr-sup)~-~M(Pr-inf) (ou =0) .

Ce message circule sur le réseau pour aboutir à l'extrémité destinatrice, qui le reçoit en franchissant la transition Te-rej. Cette transition positionne alors la valeur des deux bornes de la fenêtre d'émission modélisées par les places Pe-inf et Pe-sup.
M(Te-rej>M' avec M'(Réseau)=M(Réseau)-u et M(Pe-inf)=noacq(u)=0
et M(Pe-sup)=noacq(u)-cred(u)=0.

Nous arrivons bien à l'état d'accueil ET.

Lemme 21 : Quasi-vivacité

Nous allons maintenant prouver que notre modèle est quasi-vivant à partir de son état d'accueil ET, toutes les transitions du modèle du protocole Transport peuvent être déclenchées à partir du marquage caractérisant l'état d'accueil.

Par hypothèse, plaçons-nous dans l'état d'accueil :
Le modèle Réseau est dans son état propre d'accueil ER,
La partie émettrice du protocole Transport dans l'état E=<0,0,0,[~-~f,0[>, Et la partie réceptrice dans l'état R=<CORRECT,0,0, Ø, Ø > .

Nous franchissons la transition **Tr-accroit**, qui provoque l'accroissement du marquage de la place Pr-sup, et place une marque dans la place Pr-attendu :
M(Tr-accroit>M' avec M'(Pr-sup)=M(Pr-sup)~+~1 (ou =1) ,
M'(Pr-attendu) = M(Pr-sup) (ou =0) .

Nous pouvons alors franchir la transition **Tr-acq**, qui émet un message d'acquittement qui comporte la nouvelle valeur de la borne supérieure de la fenêtre en direction de l'autre extrémité du protocole Transport :
M(Tr-acq>M' avec M'(Réseau)=M(Réseau)+u et acq(u)=M(Pr-inf) (ou=n) ,
cred(u)=M(Pr-sup)~-~M(Pr-inf) (ou =1) et typmess(u)=ACQ.

Ce message est véhiculé par le service réseau jusqu'à son destinataire.

Nous pouvons maintenant franchir la transition **Te-acq,** qui réceptionne le message d'acquittement précédemment émis et met à jour la nouvelle valeur du crédit :
M(Tr-acq>M' avec M'(Réseau)=M(Réseau)-u ; le prédicat est vérifié
typmess(u)=ACQ et M'(Pe-sup)=acq(u)~+~cred(u) (ou =1)

Nous franchissons la transition **Te-Session**, qui place un nouveau message à émettre dans le tampon d'émission modélisé par la place Pe-fenêtre :
M(Te-Session>M' avec M'(Pe-fenêtre)=M(Pe-fenêtre)-u1+u2 et
nomess(u1)=M(Pe-compt)~-~f (ou =~-~f) ; nomess(u2)=M(Pe-compt) (ou =0) ; M'(Pe-compt)=M(Pe-compt)~+~1 (ou =1) .

Il faut franchir la transition **Te-donnée** pour émettre un message.La fenêtre d'émission le permet car un nouveau crédit vient d'être délivré, et par l'assertion A1,

la place Pe-fenêtre contient l'ensemble des prochains messages à émettre :
M(Te-donnée>M' avec M'(Réseau>=M(Réseau)+u , nomess(u)=1.

Ce message est véhiculé par le service Réseau jusqu'à son destinataire.

Le franchissement de la transition **Tr-attendu** réceptionne le message de donnée si celui-ci est dans la fenêtre de réception. Cette transition est franchissable, car nous savons que :
M(Tr-attendu>M' avec M'(Réseau)=M(Réseau)-u et
M(Pr-attendu)=nomess(u) (ou =n) et M'(Pr-reçu)=nomess(u) (ou =1).

Puis nous pouvons franchir la transition **Tr-Session** avec la marque de la place Tr-reçu qui modélise le message de donnée qui était précédemment arrivé :
M(Tr-Session)>M' avec M'(Pr-inf)=M(Pr-reçu) (ou =1).

Nous savons par l'étude du service Réseau au cours de la partie B, que ce modèle peut à tout moment se désynchroniser puis se resynchroniser en émettent sur chacune des deux voies un paquet de réinitialisation. Ce paquet arrivant à la partie émettrice du protocole Transport permettra de franchir la transition **Te-erreur**. Le paquet symétrique arrivant à la partie réceptrice franchira la transition Tr-erreur, qui place le protocole dans l'état ERRONE.

Le franchissement de la transition **Tr-rej**, permet au protocole de revenir dans un état CORRECT en émettant un message de rejet à destination de l'autre entité.

Ce message transporté par le réseau franchit la transition **Te-rej** qui prévient la partie émettrice de cet événement.

Le modèle proposé est bien **quasi-vivant** à partir de l'état d'accueil, le **lemme 21** est prouvé.

La conjugaison du lemme 20 et du lemme 21 permet d'obtenir la preuve de la vivacité du modèle, par l'application de la propriété 1.

Notre modèle possède donc la **propriété T2** de la phase de spécification de notre méthodologie, ce qui établit définitivement **l'adéquation du service.**

5. CONCLUSION

Après avoir étudié tous les comportements du protocole de la couche Transport, nous avons expliqué les raisons et la possibilité de restreindre le modèle aux fonctionnalités importantes. Nous venons de construire un modèle de la phase de transfert du protocole de la couche Transport classe 3, en nous intéressant tout particulièrement à la gestion de la fenêtre avec un crédit variable et susceptible de diminution.

Ce modèle décrit à l'aide des Réseaux de Petri à prédicats, utilise le modèle précédent décrivant le service rendu par la couche Réseau.

Après une analyse du modèle, en application à notre méthode, nous avons établi : la concordance du modèle, qui prouve que notre modèle correspond bien aux spécifications établies (preuve des propriétés E0, E1, E2, E3, E4, E5 et R1, R2, R3, R4, R5, R6); et son adéquation avec le service, qui établit que le protocole de la couche Transport permet bien d'obtenir le service nécessaire à la couche Session (preuve des propriétés T1 et T2).

Notamment, nous prouvons que le phénomène particulier de réquisition de crédit (en dehors de toute tentative de mesure de performance), géré comme la norme le spécifie, n'entraîne ni blocage ni disfonctionnement du protocole Transport, et permet de rendre le niveau de service nécessaire à la couche Session.

Nous avons employé de nombreuses techniques pour établir nos preuves, la plupart de ces techniques étant déjà connues, cependant nous les avons employées intensivement et sur un modèle décrit par RdP à prédicats:

La technique assertionnelle prouve la conservation de l'assertion dans tous les états accessibles du modèle, l'assertion devient un invariant du modèle. Cette technique est pratique , à défaut d'obtention automatique (comme pour les invariants linéaires), si l'on a une bonne connaissance de la sémantique du modèle (c'était le

cas ici, nous avons conçu le modèle) et un petit nombre de transitions (dans notre cas, une petite dizaine). La preuve est rendue plus aisée, si les prédicats des transitions sont contraignants et si l'on dispose d'autres invariants.

Nous avons employé des propriétés et des résultats démontrés par ailleurs, notamment pour établir la vivacité : Extension de la notion d'état d'accueil pour les RdPàP (ensemble d'accueil); Application d'une norme pour atteindre l'ensemble d'accueil.

Nous avons utilisé une technique plus directe ou mathématique, pour démontrer la conservation de la séquentialité. La combinaison de l'ensemble des invariants précédemment établis, les propriétés de fonctionnement du modèle Réseau, et les propriétés des modulos ont permis d'obtenir cette dernière propriété.

Nous avons, aussi, employé notre propre résultat (montrant l'équivalence de comportement de deux réseaux issus l'un de l'autre par modification de la structure interne d'une classe d'uplet) pour utiliser le modèle Réseau comme médium pour le protocole Transport.

Nous n'avons pas employé : De méthodes automatiques (il en existe peu pour les RdPàP, et nos invariants ne sont pas uniquement linéaires) ; Ni l'étude du graphe des marquages atteints (notre modèle a un graphe infini à cause des compteurs Pe-Session et Pr-Session); Ni d'outils de réduction (nous avons déjà réduit les places et transitions à la création du modèle, et il n'existe que peu de résultats pour les RdPàP).

Dans une démarche cumulative, les résultats (modèles et propriétés) pourront ultérieurement servir à l'établissement du protocole de couche supérieure (niveau Session et plus).

0.5 CONCLUSION

Après avoir situé les étapes de modélisation et de validation dans la stratégie globale de conception des systèmes informatiques, nous avons montré leurs intérêts pour établir la fiabilité des logiciels.

Pour illustrer notre approche, notre choix s'est porté sur le protocole Transport des télécommunications. Ce choix s'explique par de nombreuses raisons, que l'on peut synthétiser par les mots originalité et complexité. Les problèmes abordés recouvrent notamment, les processus concourants que forment les différents sites qui participent à la communication; les synchronisations rendues nécessaires par la coopération de plusieurs processus; les événements indéterministes comme les pertes de messages pendant le transfert; et des fonctionnalités sophistiquées et complexes telles que la gestion des phases de synchronisation du service Réseau ou la gestion du contrôle de flux à crédit variable du protocole de Transport.

Nous avons utilisé les réseaux de Petri à prédicats, qui ont prouvé et prouvent ici, leur puissance et leur adaptation à la modélisation et la validation des systèmes. Nous avons formalisé leur définition, ce qui nous a permis d'établir une preuve d'équivalence de comportement entre deux réseaux. Un réseau dérivant de l'autre par adjonction d'un champ à une classe d'uplets.

Nous avons appliqué notre méthode au service de la couche Réseau et au protocole de la couche Transport, ce qui a mis en évidence les étapes de concordance de modèle et d'adéquation de service. Ces principes sont directement issus du modèle de référence pour l'interconnexion des systèmes ouverts, appliqués à la modélisation et à la validation. Nous disposons ainsi de trois modèles et de leur propriétés associées : le modèle du service de la couche Réseau, le modèle du protocole de la couche Transport, et le modèle du service de la couche Transport.

Les réseaux de Petri sont un modèle qui permet d'obtenir une description précise

et des preuves formelles, de nombreux outils permettent d'automatiser ces preuves. Mais malheureusement, les modèles, de jour en jour, accroissent leur taille et leur complexité. Et malgré l'apparition d'abréviations et d'extensions pour les réseaux de Petri, le dessin du modèle s'accroît dans le même sens, ce qui diminue le lisibilité. Pour sortir de la contrainte graphique, une issue possible est de disposer d'un langage de description de ces mêmes réseaux de Petri, mettant en oeuvre l'ensemble des facilités d'abstraction des langages. Ce langage de description permettrait de lier les différents réseaux de Petri, notamment ceux temporels et stochastiques et les réseaux colorés, ce qui permettrait sur la même description, d'obtenir une validation du comportement et des résultats sur les performances du système.

Notre méthode, pour rendre pleinement son but, nécessite le développement d'un langage de spécification qui permet : De disposer d'une référence dépourvue d'ambiguïté; De fournir la concordance de modèle automatiquement par la génération du modèle à partir du langage de spécification; D'offrir un ensemble d'outils de validation facilitant l'adéquation de service. Toutes ces solutions existent déjà dans le monde de la recherche, mais il s'agit de les regrouper et de les adapter aux modèles de réseaux de Petri.

Nous pouvons aussi étendre la méthode en la prolongeant vers le test et l'implantation, tout en conservant l'homogénéisation des outils : Le test par génération automatique de scénario, issu de la spécification, exécutable sur une machine abstraite ; L'implantation par l'adjonction de modules pré-programmés adaptés aux protocoles de télécommunications.

Nous voyons, donc, après avoir prouvé l'intérêt et la possibilité d'application de notre méthode, que l'avenir offre encore un grand choix de développements potentiels.

Methodologie de validation des systèmes :

0.6 BIBLIOGRAPHIE

[ADA 83] Reference Manual for the ADA programming language, ANSI/MIL standard,1983 .

[André 81] C.André, "Systèmes à évolutions parallèles : modélisation par réseaux de Pétri à capacité et analyse par abstraction ", Thèse d'état, Université de Nice, 1981 .

[Ansart 82] J.P.Ansart, "A protocol independant system for testing Protocol implementation", 2ème workshop on protocol specification testing and validation, North-holland, 1982.

[Ansart 83] J.P.Ansart, V.Chari, M.Neyer, O.Rafiq, D.Simon, "Description, Simulation and Implementation of communication Protocol using PDIL", ACM 83 Communication Architectures and Protocoles, Austin - 1983.

[Ansart 86] Ansart, Amer, Chari, Lenotre, Lumbroso, Mariani, Mattere, "Software Tools for Estelle", On protocol Specification,Testing and Verification, Montreal - 1986.

[Architel 83] Architel, "Specifications Techniques d'utilisation et de raccordement", CNET, 1983.

[ARSAC 77] J.Arsac, " La construction de programmes structurés", Dunod - 1977.

[Ayache 85] J.M.Ayache, J.P.Courtiat, M.Diaz, G.Juanole, "Utilisation des Réseaux de Pétri pour la modélisation et la validation de protocoles", T.S.I vol 4 No 1 - 1985.

[Baer 80] J.L.Baer, G.Gardarin, C.Girault, G.Roucairol, "The two-step commitment protocol: modeling, specification and proof methodology", 5th International conference on Sofware Engineering, SanDiego, 1981.

[Beaudoin 84] M.Baudoin-Lafon , C.Bresse, "Caty: un environnement de programmation pour la construction graphique et interactive de programmes" ,T.S.I N°4 (spécial Génie Logiciel), 1984 .

[Behm 85] P.Behm, "A tool for analysing parallel systems in the L environnement", 6ème workshop on applications and theory of Petri Nets, Espoo - FINLAND ,1985 .

[Berthelot 81a] G.Berthelot, R.Terrat, "Utilisation de Réseaux de Pétri à prédicats pour la modélisation et la preuve de protocoles de transmission de type HDLC", AFCET,1981 .

[Berthelot 81b] G.Berthelot, R.Terrat, "Modélisation et validation de protocoles de Transport par Réseaux de Pétri à predicats", Conception des systèmes télématiques, NICE, 1981.

[Berthelot 82] G.Berthelot, R.Terrat, "Petri Nets theory for correctness of protocols", IEEE Trans. on Communications Vol COM 30, n°12 , 1982 .

[Berthelot 83] G.Berthelot, "Transformation et analyse de Réseaux de Pétri, application aux protocoles", Thèse d'état - Université PARIS VI , Juin 1983.

[Berthomieu 81] B.Berthomieu, "Technique algèbrique pour la spécification de protocole de communication ", AFCET, Paris, 1981.

[Blanchard 79] M.Blanchard, "Comprendre, maitriser et appliquer le Grafcet", édition CEPADUES, Toulouse, 1979.

[Bochman 80] G.V.Bochman, C.Sunshine, "Formal methods in communication protocol design", IEEE trans on communications-28 No4, 1980 .

[Bochmann 77] C.V.Bochmann, R.J.Chung, "A formalized description of HDLC classe of procedure", I.E.E.E nat. Telecom. , Los Angeles , 1977.

[Bouillier 84] P.Bouillier, "Contribution à la construction automatique d'analyseurs lexicographiques et syntaxiques", Thèse d'état, Univ. Orléans , 1984 .

[Bourguet 86] A.Bourguet, "A Petri net tool for service validation in protocol", 6th Interational W. on protocol specification, testing and verification p 8-17, Montréal, 1986.

[BRAMS 83] G.W.BRAMS, "Réseaux de Pétri : Théorie et Pratique" Tome I et II, MASSON, PARIS, 1983 .

[Chrétienne 83] P.Chrétienne, "Réseaux de Pétri temporisés", Thèse d'état , Universite PARIS VI , 1983 .

[Cousin 87] B.Cousin, P.Estraillier, "Etude de la resynchronisation d'un protocole de communication", TSI vol 6 n°3, 1987.

[Diaz 82] M.Diaz, "Modelling on analysis of communications and cooperations using Petri Nets", Computer networks, 1982 .

[Divito 81] B.L.Divito, "A mechanical verification of the Alternative Bit protocol", ICSCA-CMP-21 , AUSTIN , 1981.

[Dufau 84] J.Dufau, " Un outil pour la validation des protocoles décrits par RdP", thèse docteur-ingenieur, Toulouse - 1984 .

[ECMA 14] ECMA TR 14, "Local area Networks - Layers 1 to 4 - Architecture and Protocol ".

[ECMA 72] ECMA, "Standard ECMA Transport Protocol", 3ème édition, ECMA/TC24/72 , 1985 .

[ECMA 88] ECMA, "Basic virtual terminal: service description et protocol definition", 1983 .

[ESTELLE 85] ISO/TC97/SC21/W616-1 N422, "A formal description technique based on an extended state transition model ESTELLE", CCITT and ISO meeting, Paris - 1985.

[Estraillier 86] P.Estraillier, "Conception de protocoles d'interconnexion robustes: Application à la gestion d'un anneau virtuel", doctorat de l'université Paris VI, 1986 .

[Estubier 84] J.Estublier, S.Ghoul, "Un système automatique de gestion des gros logiciels : la base de programme Adele", T.S.I vol 3 n°4 (spécial Génie Logiciel) - 1984.

[Favreau 86] J.P.Favreau, R.J.Linn, "Automatic generation of test scenario skeleton from protocol specification written in Estelle", On protocol Specification, Testing and Verification, Montréal - 1986 .

[Florin 78] G.Florin, S.Natkin, "Evaluation des performances d'un protocole de communication à l'aide des réseaux de Pétri et des processus stochastiques", AFCET-CNRS : multi-ordinateurs, multi-processeurs en temps réel, Paris,1978.

[Florin 85] G.Florin, S.Natkin, "Les réseaux de Pétri stochastiques" , TSI Vol 4 n°1, 1985.

[Floyd 67] Floyd, "Assigning meanings to Programs", symposium in Appl. Math. vol 9 , Providence - 1967 .

[Foisseau 85] J.Foisseau, R.Jacquart, M.Lemaitre, M.Lemoine, G.Zanon, "SPARC : expression et gestion des spécifications d'algorithme et de représentation" , TSI vol 4 n° 4, 1985 .

[Gelenbe 82] E.Gelenbe, G.Pugolle, "Introduction aux réseaux de files d'attente", Eyrolle -1982 .

[Genrich 79] H.J.Genrich, K.Lautenbach, "The analysis of distributed systems by means of predicate/transitions nets", Semantics and concurrent computation, Lecture notes in Computer Sciences n° 70, Springer Verlag , 1979 .

[Grafcet 77] GRAFCET, "Normalisation de la représentation du cahier des charges d'un automatisme logique" , rapport final AFCET - 1977 .

[Gressier 85] E.Gresseier, "A stochastic Petri Nets model for Ethernet" , IEEE congres sur les Reseaux de Petri, TURIN - 1985 .

[Hack 75] M.Hack, "Petri Nets languages", Computer Structure Gr. Memo. 124, Project HAC, MIT, Cambridge Mass., 1975 .

[Haddad 86] S.Haddad, "Les réseaux de Pétri réguliers, validation par le logiciel ARP", 3ème colloque du génie logiciel, AFCET , 1986 .

[Hernandez 82] J.Hernandez, E.Horlait, R.Joly, G.Pujolle, "Escalibur : une structure d'interconnexion à haut débit", SEIR 2, Santiago de Compostella - 1982.

[Hoare 74] C.A.RHoare, "Monitors, an operating structured concept", CACM 17.10, p549, 1974.

[Hoare 78] C.A.R.Hoare, "Communicating Sequential Processes", CACM-21-8, 1978.

[ISO HDLC] ISO - High Data Link Connection (balanced and unbalanced) (3309, 4335, 6159, 6256), 1979 à 1981.

[ISO Test] O.S.I , "Conformance Testing" , ISO TC97/SC16/WG1 N909 , November 1985.

[ISO 7498] "Basic reference model of Open Systems Interconnection" , (DIS 7498) , 1983.

[ISO 8072] "O.S.I - Transport Service Definition ", (DIS 80729 , 1984.

[ISO 8073] " O.S.I - Transport Protocol Specification", (DIS 8073) , 1984.

[ISO 8208] "OSI - X25 Packet level Protocol" , (DIS 8208) , 1984.

[ISO 8326] "O.S.I - Basic connection oriented Session Service Definition", (DIS 8326), 1984.

[ISO 8327] "O.S.I - Basic connection oriented Session Protocol Specification", (DIS 8327), 1984.

[ISO 8348] "Network Service Definition ", (DIS 8348) , 1984.

[ISO 8650] "Specification of protocol for common Application Service element - Basic kernel ", (DP 8650) , 1985.

[ISO 8807] ISO : "LOTOS : a formal description technique based on the Temporal Ordering of Observational Behaviour" , TC97/SC21- N423 , DP 8807 - 1985 .

[ISO 8822] "O.S.I - Connection oriented Presentation Service definition", (DIS 8822), 1984.

[ISO 8823] "O.S.I - Connection oriented Presentation Protocol Specification", (DIS 8823), 1984.

[ISO 9074] ISO : "A Formal Description Technique based on Extended State

Transition Model", DP 9074 - 1985 .

[Jard 85] C.Jard, J.F.Monin, R.Groz, "VEDA, a software simulator for the validation of protocol specification", Comnet 85, Budapest - 1985 .

[Jensen 81] K.Jensen, "Coulored Petri Nets and the invariant Method", Theorical Computer Science n°14 ,p 317 , 1981 .

[Jensen 83] K.Jensen, "High-Level Petri Nets" , Application and Theory of Petri Nets , Springer-Verlag - 1983 .

[Keller 76] R.M.Keller, "Formal Verification of Parallel Programs" , Communication of ACM vol 19 N0 7 , p371-384 , 1976.

[Kotov 78] V.E.Kotov, " An algebra for parallelism based on Petri nets" , MFCS 78 , Lecture notes in Computer Science, n°64, Springer Verlag p39, 1978.

[Lamport 80] L.Lamport, "Sometimes is sometimes not never", A.C.M on principes of programming language , p174 , 1980 .

[Laprie 85] J.C.Laprie , "Sûreté de fonctionnement des systèmes informatiques et tolérance de fautes : concept de base" , T.S.I n°5 (spécial Parallélisme) - 1985 .

[Lautenbach 85] K.Lautenbach, A.Pagnoni, "Invariance and duality in Predicate / Transition Nets and in Coloured Nets", GMD n°132, 1985.

[LOTOS 85] ISO/TC97/SC21/W616-1 N299, "A formal description technique LOTOS", 1985.

[Memmi 83] G.Memmi, "Méthodes d'Analyse des RDP, réseaux à files et applications aux systèmes temps réels" ,Thèse d'état , Univ PARIS VI , 1983.

[Milner 80] R.Milner, "a Calculus of Communicating Systems", Lect Notes,Springer Verlag, 1980 .

[OVIDE 83] OVIDE, Petri net validation tool, Société SYSECA , 1983.

[Pétri 62] C.A. Petri, "Kommunication mit Automaten", Schriften des I.I.M. , n°2 , BONN - 1962

[Pnuelli 79] A.Pnuelli, "The temporal semantics of concurent programs" semantics of concurent computation, Evian - 1979 .

[Pujolle 83] G.Pujolle, "Les réseaux de l'entreprise", édition Eyrolles, 1983.

[Pradin 79] B.Pradin, "Un outil graphique interactif pour la vérification des systèmes à évolutions parallèles décrits par réseaux de Pétri", Thèse de Docteur-Ingénieur, Université Toulouse, 1979.

[Rafiq 83] O.Rafiq, "Etudes sur les techniques de description des protocoles de communication et applications ", Thèse d'état, Université de Bordeaux, 1983.

[Reisig 83] W.Reisig , "Petri Nets with individual tokens" , Application and Theory of Petri Nets , Springler-Verlag - 1983 .

[Roucairol 74] G.Roucairol, "Transformation de programmes séquentiels en programmes parallèles", 1er colloque sur la programmation, Springer Verlag, Paris - 1974 .

[Schwartz 81] R.L.Schwartz, P.M.Mellian-smith, "Temporal logic specification of distributed systems", 2eme conference on distributed computed systems, Paris - 1981 .

[Stenning 76] N.V Stenning, "A data transfer protocol", Computer Network 1, 99-110, 1976 .

[STUR 80] "Transpac caractéristique technique d'utilisation des Services", 1980 .

[Sunshine 82] C.A Sunshine "Specification and Verification of communication Protocol" , I.E.E.E Transactions , 1982 .

[Symons 80] F.J.W Symons "Introduction to numerical Petri Nets : a general graphical model of concurent processing systems", A.T.R - Vol 14 n°1,1980 .

[TELECOM 81] CNET, "Télécommunications spatiales", tome I et II, Masson editeur, 1981.

[Valet 83] I.Valet, "Description générale du protocole multipoint et des services Réseau et Transport", Nadir internal report, INRIA, 1983.

[Vautherin 85] J.Vautherin, "Un modèle algébrique basé sur le réseau de Pétri pour l'étude des systèmes parallèles", Thèse de docteur-ingénieur, université de Paris

sud, 1985.

[Véran 84] M.Véran , D.Potier , "QNAP2: a portable environnement for Queui System Modelling" , Conf on modelling Technics and Tools for Preformance Analysi INRIA -1984 .

[Vissers 84] C.A.Vissers , L.Logrippo , "The importance of the service concept the design of data communication protocol" , 5ème IFIP , Protocol Spécification Testi and Verification, Toulouse, 1984.

[Zenié 85] A. Zenjé "Colored Stochastic Petri Nets" International Workshop Timed Petri Nets , TURIN , 1985 .

[Zimmerman 80] M.Zimmermann "The ISO reference model of architecture Open Systems Interconnection", I.E.E.E Trans on Comm, vol 28 N°4, p425 ,1980 .