Etude de la resynchronisation d'un protocole de communication

Bernard Cousin, Pascal Estraillier

Titre :
Etude de la resynchronisation d'un protocole de communication

Auteurs :
Bernard COUSIN *, Pascal ESTRAILLIER *

* Laboratoire MASI, Université P. et M. Curie, 4, place Jussieu, 75252, Paris cedex 05

In T.S.I. — Technique et Science Informatiques 0752-4072/87/03/243-11/$ 3.10 @ AFCET—Bordas

Présentation : La modélisation des protocoles de communication constitue un domaine d'application privilégié pour toute théorie du parallélisme. Ceci est essentiellement dû à la façon dont sont traitées les données dans cette classe d'application. En effet, les données transitant dans un médium de communication subissent à la fois de fortes contraintes de synchronisation et de faibles transformations.

Depuis plusieurs années, des chercheurs tels que G. Berthelot, M. Diaz ou C. Girault nous ont démontré l'apport de l'utilisation de modèles formels tels que ceux fondés sur les réseaux de Petri. L'article de B. Cousin et P. Estraillier s'inscrit dans cette perspective et nous montre avec quelle concision les réseaux à prédicats permettent de décrire et d'analyser des situations complexes.

Si la modélisation d'un système reste du domaine de «l'art de l'ingénieur», ce type d'analyse devra, dans les années qui viennent, être assisté par ordinateur. Des projets importants se développent dans cette direction aujourd'hui, aux Etats- Unis ou en Europe, dans le cadre de Race ou d'Esprit. Dans leur conclusion, les auteurs citent les langages Estelle ou Lotos et par là, évoquent très clairement le projet Esprit SEDOS qui s'achèvera en cette année 1987, et dont nous espérons bien reparler plus amplement dans ces colonnes. Signé Gérard Memmi.

1. Introduction

De nombreux efforts de recherche sont dirigés vers la modélisation de protocoles de communications [Berthelot 83, Azéma 85, Estraillier 86]. La complexité des mécanismes de fonctionnement des systèmes parallèles rend la modélisation difficile et souvent laborieuse car elle repose sur une étude détaillée et complète des mécanismes mis en œuvre La modélisation doit intégrer l'ensemble (parfois très important) des fonctions réalisées. En effet, une analyse partielle de ces systèmes et l'introduction d'hypothèses simplificatrices ne sont susceptibles d'être satisfaisantes que lorsqu'on est assuré du bien—fondé de ces simplifications.

Nous proposons une démarche originale de modélisation écartant l'analyse exhaustive des mécanismes internes mis en œuvre au profit d'une construction méthodique d'abstractions permettant l'intégration des différentes fonctionnalités du système étudié. Cette méthode exhibe les propriétés du modèle et permet de valider fonctionnellement les propriétés du système. Elle est particulièrement adaptée à l'étude des

systèmes structurés en couches hiérarchiques car elle découle directement des travaux de normalisation des protocoles de télécommunications [Iso-7498 83]. Notre démarche vise à réaliser une étude locale de chacune des couches en ne prenant en compte, pour caractériser son milieu d'exécution, que les interfaces avec les couches adjacentes. Ce type de résultat est fondamental lorsqu'il s'agit d'étudier des systèmes complexes puisque l'environnement du système est représenté complètement (c.-à-d. sans hypothèse simplificatrice) et de manière concise.

Nous appliquons cette démarche à la modélisation, au moyen des réseaux de Petri à prédicats [Brams 82], des services offerts par la couche Réseau (niveau 3 de l'ISO) pendant la phase de transfert des données. Nous nous intéressons plus particulièrement au traitement des pannes des stations du réseau en décrivant le mécanisme de resynchronisation de la transmission. On modélise alors complètement la perte de paquets de données consécutive à la panne d'une station, la détection de cet événement et l'émission par la couche Réseau de paquets particuliers (de réinitialisation) permettant de prévenir la couche Transport d'une défaillance.

Après avoir détaillé les étapes de notre démarche, nous proposons le modèle des services offerts par la couche Réseau pour le transfert des données [Iso—8348 84]. Ce modèle est ensuite validé fonctionnellement dans la mesure où nous montrons que les propriétés qui lui correspondent sont en adéquation avec les propriétés qui définissent le service.

Un tel modèle peut alors être utilisé dans l'étude de la couche supérieure (Transport [Iso-8072 84, iso—8073 84]) qui intègre ainsi, sous une forme très réduite les services de la couche Réseau [Cousin 87]. Cette étude fournit ainsi une description précise et complète de protocoles en se dégageant des particularités de réalisation.

2. La démarche de validation fonctionnelle

Toute étude d'un système peut donner lieu à plusieurs modélisations différentes quant à leur forme, leur précision et leur fidélité par rapport au système modélisé. Pour valider le modèle proposé, il est donc nécessaire d'évaluer aussi sa conformité par rapport à la description du système. La démarche que nous proposons est divisée en quatre étapes (fig. 1) :

a) **La spécification des propriétés du système** que l'on veut modéliser repose sur l'étude de son fonctionnement.
Les propriétés externes, caractérisant son fonctionne—ment, sont dégagées.

b) **La modélisation des mécanismes externes** constituant le système permet d'obtenir comme modèle une abstraction des caractéristiques du système et non pas une description de ses mécanismes internes. Nous parvenons ainsi à nous dégager de toute réalisation particulière en proposant un modèle conçu seulement d'après les propriétés du système.
Pour modéliser, nous utilisons les réseaux à Prédicats qui constituent une abréviation des réseaux de Petri.
Cet outil est bien adapté à une description comportementale concise des systèmes parallèles complexes. On notera cependant que notre démarche

n’impose aucun type de modèle particulier, le choix devant être effectué en fonction des propriétés à valider.

c) **L’analyse du modèle** permet de vérifier qu’il possède l’ensemble des propriétés nécessaires.

Dans cet exemple, nous avons principalement utilisé la méthode par assertions pour vérifier les propriétés du modèle. Le principe de cette méthode consiste à vérifier qu’une assertion est vérifiée pour tout état du modèle. Dans cette étape encore, aucun procédé particulier de détermination des propriétés des modèles n’est imposé. Celles-ci sont choisies en fonction du type des résultats désirés et de la complexité du modèle.

d) **La validation fonctionnelle du modèle** permet de montrer l’adéquation entre les propriétés du modèle et celles du système étudié. Cette étape montre formellement que les propriétés obtenues dans l’étape d’analyse du modèle induisent les propriétés du système spécifiées dans la première étape. Le modèle résultant de la seconde étape constitue donc une abstraction équivalente, d’un point de vue fonctionnel, au système étudié.

Dans le cadre d’un système hiérarchisé, et en particulier d’un système de communications, notre démarche permet ainsi d’étudier localement le comportement de n’importe quelle couche et d’en proposer un modèle validé fonctionnellement et donc réutilisable dans une étude globale.

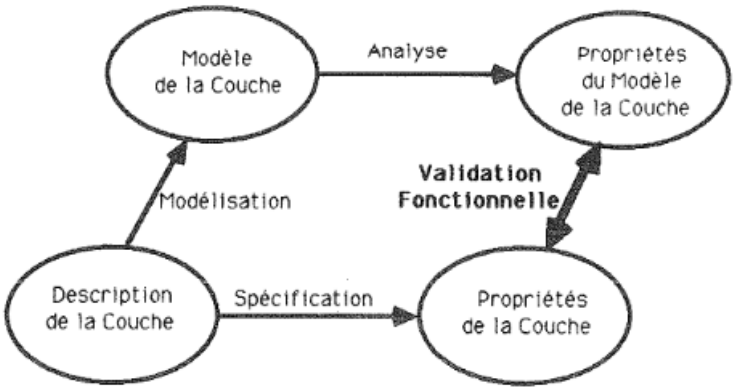

Figure 1. — Validation Fonctionnelle.

3. Le service Réseau

La normalisation en matière d’interconnexion d’ordinateurs repose sur une architecture en sept couches
[Zimmermann 80]. Elle décrit chaque couche en distinguant le *protocole*, qui explicite les mécanismes d’échange des messages, du *service* qui caractérise les fonctions offertes à la couche supérieure.

Après avoir décrit le service offert par la couche Réseau pour le traitement des pannes durant la phase de transfert des données, nous le caractérisons à l’aide de quatre propriétés. Notre étude repose sur la norme décrivant les réseaux de type TRANSPAC [Transpac 79].

3.1. PRÉSENTATION DU SERVICE RÉSEAU

En phase de transfert de données sur connexion Réseau (appelée Circuit virtuel), la couche Réseau établit un chemin bidirectionnel entre deux abonnés. Il autorise l’acheminement sur le circuit virtuel de données

structurées en paquets. Les deux voies du circuit virtuel sont indépendantes.

Le service Réseau, par opposition au protocole Réseau, n'est qu'une vue externe des fonctionnalités de la couche
Réseau. Par conséquent il n'inclue en aucun cas la description des mécanismes internes (routage, retransmission, gestion des fenêtres, ...).

Au niveau d'un circuit virtuel, on peut observer plusieurs types de situations :

a) Fonctionnement normal :

Sur chacune des voies, les paquets émis par un abonné sont pris en charge par le service Réseau et, après avoir progressé sur le circuit virtuel, ils sont délivrés à l'abonné destinataire.

b) Fonctionnement dans un environnement défaillant :

Au cours de la phase de transfert, un incident (panne de station, état incohérent du protocole, …) peut perturber le transfert des paquets (perte, détérioration, …).
On observe alors trois phases :
- **Désynchronisation** : Un élément du circuit virtuel est défaillant ; il est alors désynchronisé par rapport aux autres éléments. Les paquets dont il s'occupe peuvent alors contenir des informations incohérentes.
- **Perte** : Le mauvais fonctionnement de l'élément désynchronisé provoque la perte de paquets de données.
- **Resynchronisation** : Le service Réseau assure qu'après une phase de désynchronisation survient une phase de resynchronisation. Cette phase permet de prévenir les deux abonnés de l'occurrence d'une défaillance. Elle consiste à émettre un paquet de réinitialisation sur chacune des voies de communication. Ce type particulier de paquet est traité, au niveau des deux abonnés, par la couche supérieure (Transport) qui, ainsi prévenue de l'incident, met en œuvre un traitement de reprise.
On note cependant que les paquets de réinitialisation transitent au même titre que les autres et peuvent eux aussi être perdus à la suite d'une nouvelle désynchronisation. Cette perte n'est pas trop embarrassante dans la mesure où elle provoque ultérieurement la régénération d'un nouveau paquet de réinitialisation sur les deux voies de communication.

3.2. INTERFACE AVEC LA COUCHE TRANSPORT

Au niveau de chaque couche, des processus particuliers (appelés entités) prennent en charge l'ensemble des opérations permettant d'assurer le service offert par la couche. Ce paragraphe décrit les actions engendrées par ces entités pour la gestion d'un transfert de données.

On considère les deux abonnés situés aux extrémités du circuit virtuel. Chaque voie possède alors une extrémité émettrice et une extrémité réceptrice. Le transfert de données entre ces deux extrémités se déroule selon la manière suivante :
1) L'entité appartenant à la couche Transport de l'extrémité émettrice effectue une requête d'émission à l'entité Réseau locale.
2) L'entité Réseau de l'extrémité émettrice utilise un protocole Réseau pour transmettre la donnée (structurée en paquet) à l'entité Réseau de l'autre extrémité.

3) L’entité Réseau de l’extrémité réceptrice indique à l’entité Transport qu’un paquet a été délivré.

Les interfaces entre les couches Transport et Réseau se résument alors aux trois primitives suivantes :
- **N-SDU-Data—Requête** : La couche Transport effectue une requête d’émission de données à la couche Réseau.
- **N-SDU-Data-Indication** : La couche Réseau indique à la couche Transport la réception d’un paquet de données.
- **N-SDU-Reset-Indication** : La couche Réseau indique à la couche Transport la réception d’un paquet de réinitialisation.

Les autres primitives définies par la norme (notamment N—SDU—RESET—req) ne participent pas à la phase de transfert de données du service Réseau.

3.3. PROPRIÉTÉS DU SERVICE

On résume le service offert par la couche Réseau face aux désynchronisations durant le transfert de données par les quatre propriétés suivantes :
PS0 : Il est toujours possible de transférer des paquets (on n’observe jamais de blocage irrémédiable).
PS1 : Les paquets ne sont pas désordonnancés durant la phase de transfert (l’ordre d’émission des paquets correspond à l’ordre de réception).
PS2 : La phase de transfert ne produit pas de duplication de paquets (tout paquet de données est soit transmis, soit perdu).
PS3 : Pour le traitement des désynchronisations, le service Réseau garantit :
— Après une désynchronisation, un paquet de réinitialisation est transmis à chaque extrémité du circuit virtuel.
— Les paquets soumis au réseau avant la réception d’un paquet de réinitialisation sont soit délivrés à leur destinataire, soit détruits.
— Les paquets soumis au réseau après la réception d’un paquet de réinitialisation sont délivrés à l’extrémité réceptrice après que celle-ci ait reçu un paquet de réinitialisation.

Ces quatre propriétés du service Réseau devront selon notre démarche, pouvoir être induites par les propriétés résultant de l’analyse du modèle proposé dans le paragraphe suivant. Notre modèle sera alors fonctionnellement équivalent au modèle que l’on pourrait faire du protocole utilisé par la couche Réseau.

4. Le Modèle

Nous définissons les structures utilisées pour modéliser le service offert par la couche Réseau pour le transfert des données. Et, après un rappel sur les réseaux de Petri à prédicats, nous décrivons notre modèle.

4.1. STRUCTURES DE REPRÉSENTATION DU SERVICE

Nous devons, à partir des spécifications du service Réseau, construire un modèle dont les propriétés correspondent à celles du service à modéliser. Pour cela, nous proposons une structure sur laquelle repose une représentation fonctionnelle du service. Cette structure ne sert pas à décrire les opérations réellement effectuées par la couche Réseau (i.e. le protocole) mais permet la construction du modèle à valider.

Ce paragraphe décrit les structures permettant de supporter la modélisation du service Réseau. Nous présentons les informations gérées par la couche Transport, puis nous détaillons la solution choisie pour représenter le circuit virtuel.

4.l.l. Représentation de la couche Transport

Pour le service Réseau, la vision de la couche Transport est restreinte aux paquets qui sont transférés d'un abonné à l'autre. Les informations contenues dans les paquets étant transparentes à la couche Réseau, elles n'ont pas à être représentées dans notre étude. Toutefois, certaines propriétés du service Réseau (PS1 pour l'absence de désordonnancement et PS2 pour l'absence de duplication) exigent une certaine connaissance des informations. Pour valider notre modèle il a donc été nécessaire d'identifier les paquets émis par les entités Transport correspondant aux extrémités émettrices du circuit; nous avons choisi de les numéroter. Au niveau de chacune des voies de transmission, nous avons alors défini un compteur. Ces compteurs sont intégrés au modèle pour caractériser l'environnement de notre représentation du service Réseau Ils sont uniquement utilisés pour valider le modèle sans en perturber le fonctionnement.

Les compteurs sont initialisés avec la valeur zéro.
Après chaque émission sur une voie de transmission, le compteur correspondant est incrémenté.

Caractéristiques d'une donnée à transférer

Pour notre modèle, la donnée contenue dans un paquet est assimilée au numéro qui a été attribué au paquet. Une donnée émise au niveau de la couche Transport par une extrémité du circuit est alors caractérisée par le couple <donnée, sens> où :
. donnée : valeur du compteur d'émission
. sens : voie de transmission représentée.

4. 1 .2. Représentation du circuit virtuel

Le circuit virtuel est constitué de plusieurs éléments (stations, organes de liaisons, …) qui participent à la circulation des paquets. Fonctionnellement, il n'est pas important de distinguer ces différents éléments. On peut alors voir chaque voie du circuit virtuel comme un ensemble d'emplacements, dont chacun correspond à la zone de mémorisation d'un paquet. Un emplacement peut donc être vide ou contenir un paquet.

Pour représenter la chronologie des arrivées des paquets sur une voie de communication, cet ensemble doit être organisé sous forme d'une file. La taille de la file (TAILLEFILE) correspond au nombre maximum de paquets pouvant circuler simultanément sur la connexion.
Le premier emplacement (numéro 1) est alors associé à l'extrémité émettrice et le dernier (numéro TAILLEFILE) à l'extrémité réceptrice. Les emplacements intermédiaires correspondent aux paquets en cours de transfert.

La figure 2 décrit les deux files permettant de représenter les deux voies du circuit virtuel. Un élément du circuit est représenté à l'aide d'un couple d'emplacements, chacun appartenant à une file différente. Sur chaque voie, les emplacements sont numérotés par ordre croissant à partir

de l'extrémité émettrice. Ainsi les numéros correspondant aux emplacements d'un couple sont complémentaires.

Gestion des emplacements

Les opérations effectuées sur les emplacements permettent de représenter les actions opérées par le service Réseau sur les paquets. La nature de ces opérations dépend de l'état du circuit. Nous allons donc reprendre les situations présentées dans le paragraphe 3.1 et donner leurs interprétations pour la gestion des emplacements :
a) Fonctionnement normal
Les deux files sont gérées indépendamment. La gestion d'un emplacement revient alors à organiser le transit des paquets en assurant leur progression dans la file.
Transit : Pour qu'un paquet puisse être transféré dans un emplacement, il faut que celui-ci soit vide. Par construction de la file, le paquet à mémoriser provient de l'emplacement dont le numéro d'ordre est directement inférieur.
b) Fonctionnement dans un environnement défaillant
Pour intégrer dans le modèle la situation de défaillance, il est nécessaire de gérer le couple d'emplacements représentant un élément du circuit virtuel.
Désynchronisation : les emplacements appartenant au couple associé à l'élément défaillant deviennent désynchronisés. Les paquets contenus dans ces emplacements ne sont pas atteints.
Perte : Le paquet contenu dans un emplacement désynchronisé est perdu. L'emplacement redevient alors vide. Au niveau d'un couple d'emplacements, la perte est observée indépendamment.
Resynchronisation : Les emplacements appartenant à un couple impliqué dans le traitement de la défaillance redeviennent synchronisés. Le traitement consiste à générer dans ces emplacements des paquets de réinitialisation.

Après ce traitement, les emplacements sont donc à nouveau dans un état normal (synchronisé). Le paquet de synchronisation progresse alors dans la file au même titre qu'un paquet de données.

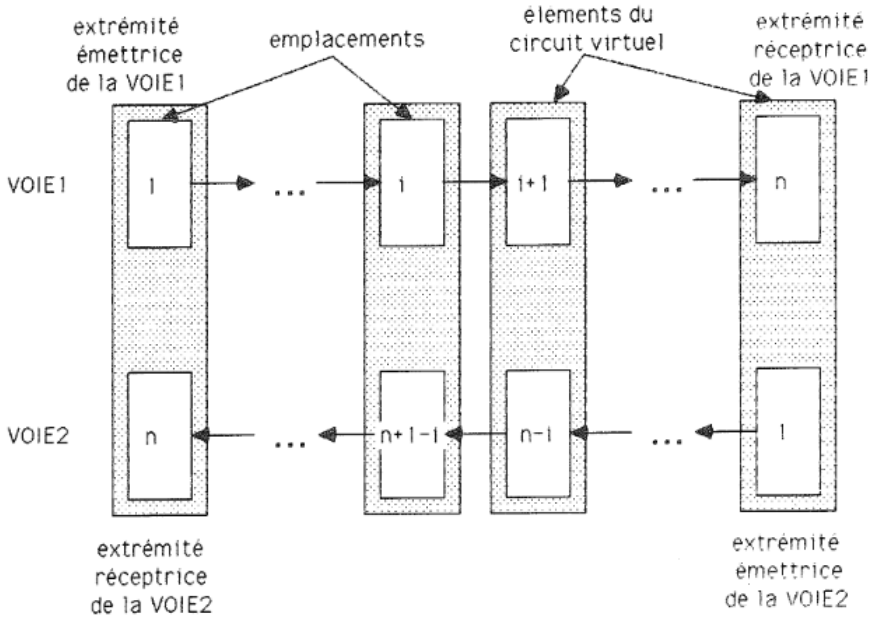

Figure 2. — Représentation du circuit virtuel.

Caractéristiques d'un emplacement

Conformément à l'ensemble des opérations agissant sur un emplacement, on peut définir un emplacement par le 4–uplet <no, info, état, sens> tel que :

. no : numéro de l'emplacement (numéro d'ordre dans la file);
. info : information contenue dans l'emplacement (rien ou un paquet de données ou de réinitialisation);
. état : état de l'emplacement (synchronisé ou désynchronisé);
. sens : voie de transmission représentée.

4.2. Les RÉSEAUX DE PETRI À PREDICATS

Les réseaux de Petri à Prédicats sont une abréviation des réseaux de Petri classiques. Ils permettent la modélisation de systèmes complexes sous une forme particulièrement concise. Notre définition est conforme à celle proposée dans [Genrich 79].

Un réseau de Petri à prédicats est un 6-uplet <P, T, C, V, A, E> tel que :

- P est un ensemble fini de places.
- T est un ensemble fini de transitions.
- C est un ensemble fini de constantes.
- V est un ensemble fini de variables à valeur dans C.
- A est un ensemble d'arcs qui forme un graphe biparti entre P et T; les arcs sont étiquetés par des sommes formelles de n-uplets formés de variables de V.
- E est un ensemble d'étiquettes qui sont des expressions logiques sur les variables de V.

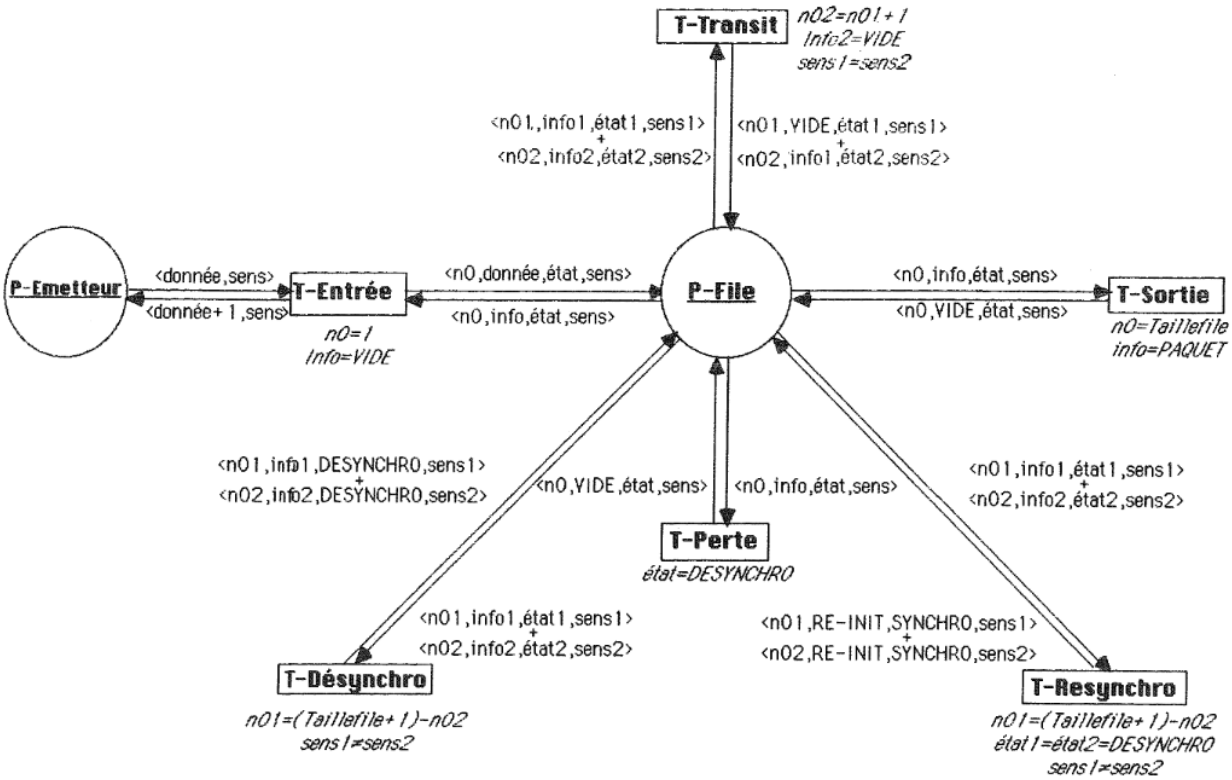

Figure 3. – Le modèle du service Réseau

Pour franchir une transition du réseau, ou doit substituer à chaque occurrence d'une variable de V située sur l'un des arcs reliés à la transition une même valeur de C.
- La transition est franchissable si l'expression logique associée à la transition est vérifiée après la substitution et si les places d'entrée contiennent suffisamment d'occurrences de constantes comme l'exigent les valuations des arcs d'entrée de la transition.
- Le franchissement de la transition enlève dans les places d'entrée de la transition et ajoute dans les places de sortie de la transition autant d'occurrences de constantes qu'il en apparait sur les valuations des arcs de sortie de la transition.

4.3. DESCRIPTION DU MODÈLE

La puissance de représentation des réseaux de Petri à Prédicats permet de proposer un modèle (fig. 3) particulièrement simple puisque l'ensemble des éléments à modéliser ont été repliés au niveau de la représentation des voies de transmission, de l'état des éléments du circuit et des emplacements.

La structure du modèle sépare distinctement les éléments appartenant à la couche Réseau, ceux qui sont rattachés à la couche Transport et ceux qui sont situés à l'interface.

4.3.1. Éléments modélisant la couche Transport

De la couche Transport, il est seulement utile de ne représenter que la génération des données par les extrémités émettrices de chaque voie de transmission.

Place P—Émetteur : Cette place permet de modéliser les extrémités émettrices du circuit virtuel. Elle contient deux marques définies par le couple <donnée, sens> tel que :

. donnée $\in$ N
. sens $\in$ {VOIE1, VOIE2}

4.3.2. Éléments modélisant la couche Réseau

Nous devons représenter les emplacements appartenant aux deux files en présentant les actions qui les modifient.

Place P—File : Cette place modélise les deux files qui représentent le circuit virtuel. Elle contient donc autant de marques qu'il existe d'emplacements dans les files (soit TAILLEFILE * 2).

Chaque marque est définie par le 4-uplet (no, info, état, sens> introduit dans le paragraphe précédent. Le domaine de chaque composante est le suivant :

. no $\in$ {l .. TAILLEFILE}
. info $\in$ {VIDE, Paquet} tel que
  Paquet $\in$ {RE-INIT, Donnée} et Donnée $\in$ N
. état $\in$ {SYNCHRO, DESYNCHRO}
. sens $\in$ {VOIE1, VOIE2}

Transition T-Transit : Cette transition modélise la progression des paquets. Elle nécessite deux marques u1 = <no1,info1,état1,sens1> et u2 : <no2,info2,état2,sens2> de la place P-File telles que :

En entrée : no1 = no1 + 1 (i.e. u2 représente l'emplacement suivant u,)
. info2 = VIDE (i.e. u2 ne contient pas d'information)
. sens1 = sens2 (i.e. u1 et u2 correspondent à la même voie)
En sortie : info1 := VIDE (i.e. u1 ne contient plus d'information)
. info2 := info1 (i.e. l'information contenue dans u1 a été transférée dans u2)

Transition T-Désynchro : Cette transition modélise la désynchronisation d'un couple d'emplacements. Elle nécessite deux marques u1 = _ <nol,infol,étatl,sens1 > et u2 = (no2,info2,état2,sens2> de la place P-File telles que:
En entrée : u1 = (TAILLEFILE + 1) - no2 (i.e. u1 et u2 constituent un couple)
. sens1 ≠ sens2 (i.e. u1 et u2 ne correspondent pas à la même voie)
En sortie : état1 := état2 := DESYNCHRO (i.e. u1 et u2 sont désynchronisés)

Transition T—Perte : Cette transition modélise la perte des paquets. Elle nécessite une marque u = <no,info,état,sens> de la place P—File telle que :
En entrée : état = DESYNCHRO (i.e. u est désynchronisé)
En sortie : info := VIDE (i.e. u ne contient plus d'information)

Transition T—Resynchro : Cette transition modélise la resynchronisation d'un couple d'emplacements. Elle implique deux marques u, = <no1,info1,état1,sens1> et u2 = <no2,info2,état2,sens2> de la place P-File telles que :
En entrée : no1 = (TAILLEFILE + 1) - no2 (i.e. u1 et u2 constituent un couple)
. sensl ≠ sens2 (i.e. u, et M2 ne correspondent pas à la même voie)
. état1 = état2 = DESYNCHRO (i.e. u1 et u2 sont désynchronisés)
En sortie : état1 := état2 := SYNCHRO (i.e. u1 et u2 sont resynchronisés)
. info1 := inf02 := RE-INIT (i.e. un paquet de réinitialisation est généré dans chaque emplacement)

4.3.3. Eléments modélisant l'interface

L'interface entre les couches Réseau et Transport est modélisé à l'aide de transitions représentant les primitives utilisées.

Transition T—Entrée : Cette transition modélise la requête d'émission de données sur une voie de transmission Elle implique d'une part une marque u = <no,info,état,sens> contenue dans P-File et d'autre part, une marque v = <donnée,sens> contenue dans la place P—Émetteur.

L'égalité, au niveau des marques, de la composante *sens* détermine la voie de transmission.
En entrée : no = 1 (i.e. le dépôt est effectué dans le ler emplacement)
. info : VIDE (i.e. l'emplacement est disponible)
En sortie : info := donnée (i.e. l'emplacement contient la donnée)
. donnée := donnée + 1 (i.e. le compteur d'émission est incrémenté)

Transition T—Sortie : Cette transition modélise les indications de réception d'un paquet de données ou de réinitialisation. Elle implique une marque u = <no,info,état,sens> contenue dans P—File.
En entrée : no = TAILLEFILE (i.e. on considère le dernier emplacement de la file, celui qui correspond à l'extrémité réceptrice)

. Info = paquet (i.e. l'emplacement n'est pas vide)
En sortie : info := VIDE (i.e. l'emplacement contient la donnée)

4. 4. MARQUAGE INITIAL

Il est commode d'introduire une notation permettant de manipuler aisément les uplets du modèle. Cette notation sera utilisée pour définir le marquage initial et dans l'évaluation du modèle. On définit un ensemble de projections élémentaires permettant d'atteindre les composantes d'un uplet à l'aide de fonctions simples.

Soit l'uplet ui tel que : ui = < noi,infoi,étati,sensi>.
. noempl(ui) : rend le numéro d'emplacement (noi) associé à ui.
. état(ui) : rend l'état (étati) associé à ui.
. sens(ui) : rend la voie de transmission (sensi) associée à ui.

Pour manipuler la composante infoi, on définit deux fonctions complexes :
. typpaquet(ui): rend le type du paquet associé à l'uplet ui
typpaquet(ui) := si infoi = VIDE alors VIDE
sinon si infoi = RE-INIT alors RE-INIT
sinon DONNÉE; _
. nodonnée(ui) : rend le numéro du paquet associé à l'uplet ui.
nodonnée(ui):= si typaquet(ui) = DONNEE alors infoi
sinon INDETERMINE

Le marquage initial correspond à l'état du système après l'établissement du circuit virtuel. On note MO(P) l'ensemble des marques contenues dans la place P à l'état initial.

Place P-EMETTEUR : A l'état initial, le compteur associé à chaque voie de transmission contient la valeur zéro.
Mo(P—Emetteur) = { <0,VOIEl>; (0,VOIE2> }

Place P—File : A l'état initial, tous les emplacements sont vides et synchronisés. Leur numéro d'emplacement et leur voie permettent de les distinguer.
. les emplacements sont vides :
$\forall$ ui $\in$ Mo(P-File); typaquet(ui) = VIDE
. les emplacements sont synchronisés :
$\forall$ ui $\in$ Mo(P—File) ; état(ui) = SYNCHRO
. les emplacements sont distinctement identifiés :
{noempl(ui)/ui $\in$ Mo(P—File) et sens(ui) = VOIE1} = [1,TAILLEFILE]
{noempl(ui)/ui $\in$ Mo(P-File) et sens(ui) = VOIE2} = [1,TAILLEFILE]

5. Evaluation du modèle

Notre modèle du service de la couche Réseau doit être validé, c'est-à-dire qu'il faut montrer qu'il respecte les propriétés du service introduites dans le paragraphe 3.3.

Nous donnons les preuves des théorèmes établissant ces propriétés. Ensuite, nous évaluons la conformité du modèle avec le service en montrant l'adéquation entre les propriétés du modèle et celles du service. On peut alors conclure à une équivalence fonctionnelle entre notre modèle du service Réseau et un modèle du protocole Réseau.

5.1. PROPRIÉTÉS DU MODÈLE

Pour valider fonctionnellement notre modèle, il est nécessaire de déterminer les propriétés qu'il doit vérifier.

PMO : Le modèle est vivant (il n'y a pas de blocage). Cette propriété correspond à la notion usuelle de vivacité.

PM1 : Le modèle respecte, dans la gestion des uplets, la relation d'ordre définie sur les numéros de paquets (il conserve la séquentialité des paquets).

PM2 : Le modèle ne génère pas plusieurs uplets qui référencent le même paquet (il ne duplique pas les paquets).

PM3 : Toute séquence de franchissement décrivant la perte d'un paquet est prolongé par une séquence représentant une resynchronisation.

5.2. VÉRIFICATION DES PROPRIETES DU MODÈLE

Les propriétés du modèle sont démontrées à l'aide de trois théorèmes. Ce paragraphe présente la démarche de preuve utilisée; les preuves complètes sont proposées dans [Cousin 85].

Pour les preuves, il est nécessaire d'introduire les ensembles suivants :

. A : Ensemble des marquages accessibles à partir du marquage initial
. T : Ensemble des transitions du modèle.

Il est aussi nécessaire d'introduire des fonctions permettant la manipulation des uplets.

Soit P un ensemble d'uplets contenus dans la place P-File.

Suivant(ui,P) : La fonction rend l'uplet (appartenant à l'ensemble P) qui correspond au premier emplacement non vide dont le numéro est inférieur à l'emplacement désigné par ui.

Soit ui $\in$ P, soit uk $\in$ P; Suivant (ui,P) = uk, ssi :
. typaquet(uk) != VIDE (i.e. uk contient un paquet)
. noempl(uk) <= noempl(ui) (i.e. uk est plus proche de l'émetteur)
. $\forall$ un $\in$ P tel que typaquet(un) != VIDE et noempl(un) <= noempl(ui) noempl(un) <= noempl(uk) (i.e. uk est entre un et ui)

Premier(P) : La fonction rend l'uplet qui correspond à l'emplacement (de l'ensemble désigné) contenant le premier paquet de données pris en compte. Cet emplacement est alors plus proche de l'extrémité réceptrice que tous les autres emplacements non vides de l'ensemble.

Soit uk $\in$ P, Premier(P) = uk ssi :
. typaquet(uk) $\neq$ VIDE
. $\forall$ un $\in$ P tel que typaquet(un) $\neq$ VIDE alors
noempl(uk) >= noempl(un) (i.e. uk est le plus proche de l'extrémité réceptrice)

Dernier(P) : La fonction rend l'uplet qui correspond à l'emplacement (de l'ensemble désigné) contenant le dernier paquet de données pris en compte.

Soit uk $\in$ P, Dernier(P) = uk ssi :
. typaquet(uk) = DONNEE (i.e. uk contient un paquet de donnée)
. $\forall$ un $\in$ P tel que typaquet(un) = DONNEE alors
noempl(uk) <= noempl(un) (i.e. uk est le plus proche de l'extrémité émettrice)

Voiel(P) : Cette fonction rend l'ensemble des uplets appartenant à la voie de transmission VOIE1 de l'ensemble P.

Voiel(P) = {ui ∈ P tel que sens(ui) = VOIE1}

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

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

Compteur(P) : Cette fonction rend la valeur de la composante <donnée> d'une marque contenue dans la place P-Emetteur. La marque est déterminée en fonction de la voie associée aux uplets de l'ensemble P.

5 .2. l. Preuve de la propriété PMO

Pour prouver la propriété PMU, nous utilisons le théorème suivant :

Théorème TO : le modèle est vivant (toute transition du modèle peut toujours être rendue franchissable}.

∀ M ∈ A, ∀ t ∈ T, ∃ S ∈ T* tel que M(St>.

La vivacité du modèle est vérifiée en deux étapes. La première montre que le modèle possède un ensemble E d'états d'accueil. La seconde étape prouve que dans un état quelconque de l'ensemble E, le modèle est quasi vivant.

Étape 1 : Le modèle possède un ensemble E d'états d'accueil.

Nous définissons l'ensemble E par le marquage suivant :

. les uplets de P-FILE sont vides de contenu, leur état est synchrone

∀ M ∈ E, ∀ u ∈ M(P-FILE,

typaquet(u) = VIDE, état(u) = SYNCHRO

. les uplets de P—Émetteur peuvent être quelconques.

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 l'un des états de l'ensemble E. D'après [Keller 76} un modèle possède un état d'accueil E, si en appliquant une norme B qui affecte à chaque marquage un poids dans N :

- La norme est nulle pour l'état d'accueil (B(E) = O) ;
- Pour tout marquage accessible non nul, il existe une séquence de franchissement qui fasse décroître la norme (V M e A si B(M) != 0 alors

  ∃ S ∈ T* tel que M(S>M' et B(M) > B(M')).

Nous définissons notre norme ainsi :

∀ M ∈ A, ∀ u ∈ M(P—Emetteur) B(u) = 0;

∀ u ∈ M(P—File),

B(u) = B(noempl(u))*(B{typaqnet(u)) + B(état(u)) avec

B(no) = no; B(PAQUET) = l; B(VIDE) = O; B(DESYNCHRO) = 2; B(SYNCHRO) : 0.

A l'état initial, nous vérifions que la norme est nulle.

La suite de la démonstration est effectuée en deux sous-étapes : on montre que le modèle permet, d'une part la resynchronisation de tous les emplacements, et d'autre part la progression des paquets de données et de réinitialisation.

Sous–étape 1 : Il est possible de resynchroniser les emplacements désynchronisés de la place P–File.

L'idée consiste à supprimer les emplacements désynchronisés de la place P-File. On veut donc :

$\forall$ u $\in$ M(P–File); état(u) = SYNCHRO.

Pour la démonstration, il est nécessaire d'introduire le lemme suivant :
LEMME L0 : Les uplets couplés pour la représentation d'un élément du circuit virtuel sont toujours dans le même état.

$\forall$ M $\in$ A, $\forall$ ui $\in$ Voie1(M(P–FILE)),$\forall$ uk $\in$ Voie2(M(P-FILE)),
si noempl(ui) = (TAILLEFILE + l) * noempl(uk) alors
état(ui) = état(uk).

D'après ce lemme, on peut considérer séparément les uplets associés à chacune des deux files puisqu'on est assuré qu'ils possèdent toujours le même état.

Soit Voie l'ensemble des uplets de P-File appartenant à l'une quelconque des voies de transmission.
Tant que $\exists$ u $\in$ Voie; état(u) $\neq$ SYNCHRO (P–File contient au moins un paquet)
Déclencher T-Resynchro (un paquet RE–INIT est alors généré dans l'emplacement qui retourne à l'état SYNCHRO. Le réseau comporte alors un emplacement désynchronisé de moins).
Fin

Sous–étape 2 : il est possible de délivrer au récepteur l'ensemble des paquets présents dans la place P–File (les emplacements deviennent alors vides).
L'idée consiste à vider les emplacements de la place P–File. On veut donc que :

$\forall$ u $\in$ M(P-FILE); typaquet(u) = VIDE

Pour cela, ou peut considérer séparément les uplets associés à chacune des files. En effet, toutes les transitions qu'il est nécessaire de franchir (T-Transit et T-Sortie) n'agissent que sur une seule voie de transmission. On effectue alors les actions suivantes :
Tant que $\exists$ p $\in$ M(P-FILE); p = Premier(M(P-FILE) (P–File contient au moins un paquet)
Tant que noempl(p)!= TAILLEFILE (p ne correspond pas à l'extrémité réceptrice)
. Déclencher T-Transit (toujours possible puisque, par construction, les emplacements entre noempl(p) et TAILLEFILE sont vides. Le paquet progresse d'un emplacement)
. p = Premier(M(P–FILE)) (Le paquet conserve cependant son statut de Premier)
Fin
Déclencher T-Sortie (désormais possible puisque les conditions sont satisfaites. Un paquet de moins est alors présent dans le réseau)
Fin

Nous avons vérifié que les actions effectuées conduisaient à l'obtention d'uplets caractérisant des emplacements vides et synchronisés, ce qui montre que l'ensemble E est bien un ensemble d'accueil. Il est désormais nécessaire de prouver, que les actions décrites peuvent toujours être exécutées en montrant que le modèle est quasi-vivant.

Étape 2 : Dans un état quelconque de l'ensemble E d'accueil, le modèle est quasi-vivant.

Pour prouver cette propriété, nous devons franchir successivement toutes les transitions du modèle.

Dans un état quelconque de l'ensemble des états d'accueil, la place P—File ne contient que des emplacements vides.

. La transition T-Entrée est franchissable et génère un paquet de donnée dans le premier emplacement d'une file.
. La transition T-Désynchro est toujours franchissable.
Nous la franchissons pour le premier emplacement de la file précédente. Cet emplacement devient donc désynchronisé (ainsi que l'emplacement qui lui est associé dans l'autre file).
. La transition T-Perte peut alors être franchie et le paquet de donnée contenu dans cet emplacement désynchronisé est perdu. L'emplacement redevient vide.
. La transition T-Resynchro est franchissable, elle génère un paquet de réinitialisation dans chaque emplacement du couple désynchronisé.
. La transition T-Transit permet alors la progression du paquet de réinitialisation à travers le réseau, jusqu'à l'extrémité réceptrice.
. La transition T-Sortie devient alors franchissable

5.2.2. Preuve des propriétés PM1 et PM2

Les propriétés PMI et PM2 concernent la séquentialité et l'absence de duplication des paquets.

Pour prouver les propriétés, nous utilisons le théorème suivant :

THÉORÈME Tl : le modèle ne duplique pas les paquets et conserve leur séquentialité.

Soient deux emplacements distincts, d'une même voie et contenant des paquets de données :
. Les paquets sont différents.
. Le paquet le plus près du récepteur (i.e. contenu dans l'emplacement dont le numéro est le plus grand) est le paquet ayant été émis le plus tôt (le numéro de paquet est le plus petit).

$\forall M \in A, \forall Voie \in \{ Voie(M(P\text{-}FILE)) , Voie2(M(P—FILE)) \}$
$\forall ui \in Voie; \forall uk \in Voie - \{ui\}$
tels que typaguet(ui) : typaquet(uk) == DONNEE
si noempl(ui) > noempl(uk)
alors nodonnée(ui) < nodonnée(uk)

La preuve est réalisée en deux étapes : nous montrons d'abord qu'il est possible de restreindre le modèle et de l'assimiler le modèle à une simple file FIFO, ce qui nous donne les propriétés recherchées. Nous montrons ensuite que ces propriétés sont préservées dans la gestion de la Désynchronisation.

Étape 1 : Assimilation du modèle à celui d'une file FIFO.

Dans les transitions T—Entrée, T-Transit et T—Sortie, les composantes <état> et <sens> ne sont pas référencées dans les conditions de déclenchement, ni affectées dans les valuations des arcs de sortie. On peut alors projeter les uplets du modèle, restreint à ces transitions,

sur le uplet <no, info>. On obtient ainsi le modèle classique d'une file FIFO sans perte [Berthelot 81]. Il a été montré que ce modèle possédait les propriétés de séquentialité (absence de désordonnancement des informations véhiculées) et de non-duplication. Notre modèle, restreint à ces transitions possède donc lui aussi ces propriétés.

Étape 2 : Préservation des propriétés dans la gestion de la désynchronisation

Les propriétés de séquentialité et de non—duplication sont exprimées par les composantes <no> et <info> des uplets.
. La transition T-Désynchro ne modifie que la composante (état) des uplets; les autres composantes ne sont donc pas affectées et les propriétés sont conservées.
. La transition T—Perte efface le contenu du paquet mais ne modifie pas le numéro d'emplacement Le paquet devient donc vide et n'intervient plus dans le théorème.
. La transition T-resynchro remplace l'information contenue par un paquet de réinitialisation sans modifier le numéro d'emplacement. Là encore, le paquet n'intervient plus dans le théorème.

5.2.3. Preuve de la propriété PM3

La propriété PM3 concerne le traitement des désynchronisations. Pour la prouver, nous utilisons le théorème T2.

THÉORÈME T2 : Toute désynchronisation est correctement détectée.
Soit un emplacement contenant un paquet de données.
Si sur la même voie il existe un emplacement :
- plus près de l'émetteur et
- contenant un paquet de données dont le numéro n'est pas consécutif (il y a eu perte entre ces deux paquets)

Alors :
. soit il existe un paquet de réinitialisation dans un des emplacements de numéro inférieur (plus proche de l'émetteur}
. soit un des emplacements de numéro inférieur est désynchronisè.

$\forall$ M $\in$ A, $\forall$ Voie e { Voie1(M(P—FILE)) , Voie2(M(P-FILE)) }
$\forall$ ui $\in$ Voie tel que typaquet(ui) = DONNEE
si $\exists$ uk $\in$ Voie — {ui} tel que Suivant(ui, Voie) = uk et
tvpaquet(un) = DONNEE et nodonnée(ui) > nodonnée(uk,) + 1
alors :
. T21: $\exists$ un $\in$ Voie — {ui} tel que
noempl(un) < noempl(ui) et typaquet(un) = RE — INIT
. T22 : $\exists$ un $\in$ Voie – {ui} tel que :
noempl(un) (noempl(ui) et état(un) = DESYNCHRO

Pour prouver que ce théorème est un invariant du modèle, nous prouvons qu'il est vérifié à l'état initial M0 (trivial), puis que tout franchissement de transitions conserve le théorème.

Pour la démonstration, il est nécessaire d'introduire le lemme suivant :

LEMME L1 : Toute désynchronisation affectant le dernier paquet émis est détectée.

Soit le paquet le plus près de l'émetteur, s'il ne vient pas d'être émis (son numéro ne correspond pas à la valeur du compteur d'émission décrémentée de 1) alors :
. 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.

∀ M ∈ A, ∀ Voie ∈ {Voie1(M(P—FILE)) , Voie2(M(P-FILE))}
si ∃ ui ∈ Voie tel que Dernier(Voie) = ui alors
nodonnée(ui)< Compteur(Voie) — 1 et
soit L11 : ∃ un ∈ {Voie — ui} tel que :
noempl(un) < noempl(ui) et typaquet(un) = RE — lN1T
soit L12 : ∃ un ∈ {Voie - ui} tel que :
noempl(un) < noempl(ui) et état(un) = DESYNCHRO

Pour prouver le théorème, nous étudions le comportement de chaque transition :
. Avant le franchissement de la transition T-Entrée, sur n'importe quelle voie, il existe un dernier paquet émis. D'après le lemme 1, toute désynchronisation le concernant est détectée. En cas de désynchronisation, un emplacement situé entre lui et l'émetteur contient donc un paquet de réinitialisation ou est dans l'état désynchronisé.

La transition T-Entrée génère un paquet dont la donnée correspond à la valeur du compteur d'émission.
Après le franchissement, le théorème T2 est donc vérifié. Ce paquet devient alors le dernier paquet émis, sa désynchronisation est détectée et on observe la situation précédente.

La transition T—Transit provoque la progression des paquets en les transférant dans des emplacements vides. Etant donné que :
- T—Transit ne modifie ni les informations contenues dans les paquets qui transitent, ni l'état des emplacements;
- Les emplacements vides ne sont pas pris en compte par le théorème T2.
- Chaque numéro d'emplacement identifie de manière unique son uplet porteur.

On peut conclure que la transition T—Transît conserve le théorème T2.
. Les transitions T-Sortie et T-Perte suppriment les paquets en les échangeant par des emplacements vides de même numéro. Les emplacements vides n'étant pas pris en compte par le théorème T2, il conserve sa véracité.
. Le franchissement de la transition T-Désynchro fait passer un emplacement de chaque voie de transmission dans l'état désynchronisé. De ce fait, cet emplacement vérifie, par construction, la propriété T22 du théorème T2.
. Le franchissement de la transition T-Resynchro remplace, pour chaque voie de transmission, un emplacement en état désynchronisé par un emplacement synchronisé contenant un paquet de réinitialisation.
Avant le déclenchement de la transition, la propriété T22 et, après le franchissement, la propriété T21 étaient alors vérifiées.

Le théorème T2 est donc vérifié pour chaque transition du modèle.

5.3. VALIDATION FONCTIONNELLE

La validation fonctionnelle du modèle consiste à vérifier que les propriétés du modèle peuvent induire celles du service modélisé.

Naturellement, la complexité de cette démarche dépend de la qualité des propriétés que l'on est parvenu à tirer du modèle. Dans notre exemple, la validation fonctionnelle est facile pour les trois premières propriétés du service, car nous avions déterminé les propriétés du modèle en fonction de cela.

Ainsi, on montre trivialement les adéquations suivantes :
. la propriété du modèle PMO (vivacité du modèle) correspond directement à la propriété du service PSO (absence de blocage irrémédiable).
. la propriété du modèle PM1 (conservation de la séquentialité des paquets) induit la propriété du service PS1 (absence de désordonnancement des paquets).
. la propriété du modèle PM2 (absence de duplication de paquets) correspond précisément à la propriété du service PS2.

La preuve de l'adéquation entre la propriété PM3 (traitement des désynchronisations) et la propriété du service PS3 nécessite l'utilisation de lemmes supplémentaires.

LEMME 2 : Toute perte de paquet est déterminée par la présence d'un emplacement désynchronisé :
$\forall$ M $\in$ A, si M(T-Perte) alors $\exists$ ui $\in$ M(P—File)
tel que état(ui) = DESYNCHRO

LEMME 3 : Toute phase de resynchronisation se concrétise par l'émission, au niveau de chaque voie de transmission, d'un paquet de resynchronisation :
$\forall$ M $\in$ A, si M(T-Resynchro> alors
$\exists$ u1 $\in$ Voie1(M(P—File)) tel que typaquet(u1) = RE-INIT
et $\exists$ u2 $\in$ Voie2(M(P—File))tel que typaquet(u2) = RE-INIT

La propriété PS3 caractérise le comportement du service Réseau face aux désynchronisations. Nous prouvons, à l'aide des lemmes 2 et 3 et en utilisant les théorèmes T1 et T2 que le modèle est conforme à sa spécification :
. D'après le lemme 2, la transition T-Perte n'est franchissable que si l'un des emplacements est désynchronisé.
. D'après le lemme 3, la phase de resynchronisation (transition T-Resynchro) insère un paquet dans chaque voie de transmission.
. Le théorème T2 affirme que ces insertions sont effectuées après toute désynchronisation.
. L'ordre relatif des paquets est conservé jusqu'à leur livraison à l'extrémité réceptrice, et ce, sans duplication (théorème Tl).

L'ensemble de ces points permet de constater que la propriété PS3 est vérifiée à partir des propriétés du modèle.

6. Conclusion

Nous avons proposé dans cet article un modèle des services de resynchronisation dans la phase de transfert de données de la couche Réseau. Le modèle obtenu est minimal car il regroupe sous une forme très réduite les propriétés exigées. Il est fonctionnellement équivalent à un modèle des protocoles décrivant l'ensemble des mécanismes internes.

Notre modèle peut alors être intégré à la modélisation de la couche Transport. En effet, les assertions relatives au modèle de la couche Réseau sont conservées dans le modèle de la couche Transport. Ces

assertions peuvent ainsi être utilisées dans la démonstration des propriétés de la couche Transport.

La démarche de validation fonctionnelle d'un modèle peut être généralisée à la caractérisation de tout autre milieu d'exécution hiérarchisé. Le développement con—joint de spécification et des modèles de validation permettront certainement dans un proche avenir de disposer d'outils pour spécifier, modéliser et valider de manière rigoureuse et aisée.

BIBLIOGRAPHIE

[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 ; Technique et Science Informatiques, vol. 4, n° 1, janvier-février 1985.

[Azéma 85] P. AZÉMA : Protocol Analysis by using Petri Nets ; IFIP 1985 — Workshop on Protocol Specification Testing and Verification, 1985.

[Berthelot 81] G. BERTHELOT, R. TERRAT : Petri Nets theory for correctness of protocols ; IEEE Trans. on Comm. vol. COM 30, n" 12, 1981.

[Berthelot 81] G. BERTHELOT : Transformation et analyse de R.d.P.. Application aux protocoles; Thèse d'état, Université de Paris VI, juin 1983.

[Brams 82] G. W. BRAMS : Réseau de Pétri : Théorie et Pratique ; Tome I et II, Masson, 1982.

[Cousin 85] B. COUSIN, P. ESTRAILLIER : Validation fonctionnelle de réseaux et prédicats : Application au modèle d'un protocole de communications ; publication MASI, Paris, 1985.

[Cousin 87] B. COUSIN : Modélisation et Validation de systèmes structurés en couches ; Thèse de l'Université de Paris 6, avril 87.

[Estraillier 86] P. ESTRAILLIER Conception de protocoles d'interconnexion robustes ; Thèse de doctorat de l'Université de Paris 6, 1986.

[Genrich 79] H. J. GENRICH, K. LAUTENBACH : The analysis of distributed systems by means of Predicate/Transitions Nets ; Lectures notes in Computer Sciences 110 70, Springer Verlag, 1979.

[ISO_7498 83] NORME ISO : Basic Reference Model on Open Systems Interconnection ; ISO/Dis 7498, 1983.

[ISO_8072 84] NORME ISO : OSI- Transport Service Definition ; ISO/Dis 8072, 1984.

[ISO_8073 84] NORME ISO : OSI - Transport Protocol Specification; ISO—Dis 8073, 1984.

[ISO_8348 84] NORME ISO : OSI — Network Service Definition ;ISO/Dis 8348, 1984.

[Transpac 78] TRANSPAC : Transpac caractéristiques techniques d'utilisation des services - STUR ; 1979.

[Zimmermann 80] M. ZIMMERMANN : OSI reference model - the ISO model of architecture for Open Systems Interconnection ; IEEE Trans. on Comm., vol. COM 28,1980.

Article reçu le 20 mai 1986. Version révisée en septembre 1986.

Bernard Cousin, docteur de l’Université de Paris 6, est assistant à l’Institut de Programmation de l'université Paris 6. Au sein de l‘équipe «modélisation et parallélisme» du laboratoire MASI, il travaille sur l’intégration des phases de spécification, modélisation et validation des systèmes distribués.

Pascal Estraillier, docteur de l‘Université Paris 6, est assistant à l‘Institut de Programmation. Il est membre de l'équipe « modélisation et parallélisme » du laboratoire MASI où il mène des recherches sur la conception, la modélisation et la validation de protocoles de communication robuste.

Université P. et M. Curie, laboratoire MASI,
Tour 65—66, 4, place Jussieu, 75252 Paris cedex 05.