Entreprise: ALCATEL CIT
Rue de Broglie
BP 344
22304 Lannion

Service: OML/GEP

Tuteurs: B. LOYER (96.04.73.17)
C. COLIN

Responsable: M. BOURI (ENSSAT)

VALADE Eric
LSI III

RAPPORT DE TUTORAT

SPECIFICATION LOTOS ET VERIFICATION DU SERVICE TELEPHONIE RNIS ET DE L'APPEL EN INSTANCE

ISDN telephony teleservice and call waiting
LOTOS specification and verification

Juillet 1994

Table des matières

Remerciements

Ce stage a laissé une place importante à la réflexion mais ce type de travail ne saurait s'envisager sans encadrement technique.
Je tiens à remercier vivement Monsieur Daniel COURTEL, chef du service OML, et Monsieur Daniel VIARD, chef du groupe GEP qui m'ont accueilli au sein de leur structure.

Ces remerciements ainsi que ma profonde reconnaissance s'adressent tout particulièrement à Madame Catherine COLIN et à Monsieur Bernard LOYER, pour m'avoir proposé ce tutorat et prodiguer de nombreux conseils tant au niveau de l'étude réalisée que de la rédaction de documents.

Je suis sincèrement reconnaissant à Madame Marie-Line Grone pour sa gentillesse et sa constante bonne humeur, ainsi qu'à l'ensemble du personnel du service OML Lannion pour son accueil.

Je remercie également Madame Mounia BOURI pour m'avoir suivi au cours de ce tutorat, ainsi que toutes les personnes à qui sera distribué ce document pour leur attention.

Mots clés / *Keywords* :

Systèmes distribués
Distributed systems

Protocoles et services de télécommunications
Communication protocoles and services

Réseau Numérique à Intégration de Services
Integrated Services Digital Network

Techniques de Description Formelle
Formal Description Techniques (FDT)

Langage de spécification à ordonnancement temporel
Language Of Temporal Ordering Specification (LOTOS)

Graphe de simulation
Simulation graph

Vérification de modèle
Model Checking

Introduction

Le tutorat de l'innovation s'est déroulé au sein de la société *Alcatel CIT* dans la branche Commutation publique, et plus précisément dans le groupe *OML/GEP* de Lannion.

Cet établissement consacre l'essentiel de son activité à la recherche, l'étude et la réalisation de systèmes de commutation tels que le produit Alcatel 1000 E10. Il contribue entre autres à des projets dans le domaine de la communication tel que l'implantation du RNIS (Réseau Numérique à Intégration de Services) en France et à l'étranger, ou le développement de brasseurs ATM (Asynchronous Transfert Mode).

Mon stage s'est effectué dans le service *OML* (Outillage et Méthodes de développement Logiciels), dont le rôle est de produire et former à de nouvelles méthodes destinées à faciliter la conception de projets de télécommunication.
Celui-ci se décompose en plusieurs groupes dont le *GEP* (Groupe d'Etudes et Prospective) au sein duquel je fus accueilli. Sa vocation est de réaliser une veille technologique, des études de faisabilité ou encore des maquettes d'applications à la télécommunication et d'en assurer le transfert vers les groupes de développement.

C'est dans cette perspective que s'est déroulé mon tutorat. Dans le cadre de la prospective, l'axe actuel d'étude de ce groupe concerne les techniques de description formelle en général et plus particulièrement deux standards: *SDL* (Specification Description Language) et *LOTOS* (Language Of Temporal Odering Specification).
Le thème de mon stage a porté sur la spécification formelle des services de téléphonie RNIS et d'appel en instance à l'aide du langage LOTOS et à son analyse à travers l'expérimentation d'outils supports dont la boite à outils caesar-aldebaran.

Aprés une présentation de l'entreprise et des caractéristiques de l'étude, ce rapport traite de la spécification formelle des services de télécommunication. Il aborde ensuite l'exploitation de cette matière à travers l'utilisation et la conception d'outils supports. Enfin, il propose une discussion-réflexion en guise de conclusion.

Les annexes sont fournies dans un document séparé et intégrent les spécifications LOTOS conçues, des documents techniques ainsi que des exemples d'analyse de description.

Abstract

This industrial placement took place at *Alcatel CIT* Commutation in Lannion in the *Prospective and Studies Group* (GEP) which is one sub-branch of the "*tools and software development methods*" department (OML).
The aim of this service is the experimentation of new software technologies, by making feasibility studies with communication domain, and to assure their transfer, with tools and methods adjuncts, in the development groups of the company. A prospective domain concerns the *Formal Description Techniques* (FDT) and among them two standards : *SDL* (Specification Description Language) and *LOTOS* (Language Of Temporal Ordering Specification).

In view, the work I was assigned with consists in the application of LOTOS description technique on two ISDN (Integrated Services Digital Network) services: the *telephony teleservice* and the *call waiting supplementary service*. ITU (International Telecommunication Union) recommendations have been used as references in this perspective.
This formal description was considered as support for an analysis by using part of development tools. In this way, the specification is gradually validated in parallel to its elaboration, with regard to requirements. some of tools used, *Hippo* and *Smile interactive simulators*, allow to trace some possible execution offered. But a better check requires the use of more powerful tools and the service has recently acquired new techniques as *caesar-aldebaran toolbox.*
So one target of my work was to get used and to experiment this kind of tools on LOTOS descriptions. I was more particularly attached on studying possibilities of *exhaustive analysis* in the case of *caesar*, *aldebaran* and the *open-caesar environment.*

This document is a final report of the training course structured as follows:

- the first chapter gives a *short presentation of the Alcatel CIT company* and describes gradually *caracteritics of the work* I was assigned to.
- the second chapter presents the *conception of the services specification.*
- the next chapter is dedicated to the *different form of analysis.*
- finally, the last one gives an *review of the experimentation.*

The appendices, at the end of this development, contain specifications realized, technicals documents, and some examples of description examination.

I . Cadre du Travail

I.1 L'entreprise

I.1.1 Le groupe ALCATEL NV

Alcatel NV est né du rapprochement, début 1987, des activités de communication d'*Alcatel Alsthom*, nommé CGE jusqu'au 1er janvier 1991, et d'*ITT*. Ce groupe est composé de 24 sociétés réparties dans le monde, et est structuré en 4 groupes de produits:

- *Systèmes de réseaux*: commutation publique, transmission sur câble et intégration de réseaux.
- *Radiocommunication*, *espace* et *défense*:
- *Systèmes d'entreprise*: communication d'entreprise et téléphonie privée.
- *Câble*: câble optiques et métalliques.

Alcatel NV (134 000 personnes, 110 milliards de francs de chiffre d'affaire) figure dans les toutes premières places au niveau mondial par ses domaines d'activité et la recherche y joue un rôle important.

I.1.2 La société Alcatel CIT

Alcatel CIT (Compagnie Industrielle des Télécommunications), dont l'origine remonte à 1879, est intégrée au groupe *systèmes de réseaux*. Choisie par *France Télécom* pour développer la commutation électronique temporelle, *Alcatel CIT* fournit les centraux temporels, éléments essentiels de la modernisation du réseau national. Ces systèmes sont largement exportés et cette société occupe aujourd'hui la première place mondiale dans le domaine de la commutation téléphonique numérique avec plus de 40 milliards de lignes en service ou commandées dans 80 pays.
Ses activités sont réparties en 4 secteurs:

- *La branche commutation publique*

  Au premier rang mondial en commutation numérique, cette activité emploie plus de 7000 personnes et représente 55% du chiffre d'affaire. Son produit de base, Alcatel E10, constitue le système numérique le plus répandu au monde.

- *Le département transmission sur câbles*

  Il représente 22% du chiffre d'affaire et est un des premiers fournisseurs mondiaux de sys tème de transmission numérique terrestre et sous-marine. C'est également le leader européen des réseaux de vidéocommunications par fibre optique.

- *Le département industries*

  Cet ensemble couvre deux grands domaines: la fabrication de composants et les technologies du vide.

- *Le département services et réseaux*

  Alcatel Data Networks développe des réseaux X.25 et Frame Relay avec son système 1100, et vise à étendre cette panoplie en réseaux multimédia avec le système ATM Alcatel 1000 HSS. Il fournit des services logiciels d'ingénierie et des conseils informatique aux autres branches.

L'*établissement de Lannion* regroupe les branches commutation publique et transmission depuis 1992. Parmi les faits marquants, celui-ci a connu la mise en service du premier système téléphonique à base de commutation temporelle, à Perros–Guirrec en 1969. Il participe depuis à de nombreux projets dont l'implantation du RNIS en France.

I.1.3 Le service Outillage et Méthodes de développement Logicielles (OML)

Depuis 1991, ce service est intégré à la Direction des Services Techniques Communs. Il se décompose en plusieurs groupes implantés à Vélizy, Lannion, Nantes et Villarceaux:

- *GDV, GDL, GDN, GDT*: groupe de développement d'outils supports à la réalisation de projets.
- *GMS*: Groupe Méthodes et Standards

- *GEP*: Groupe d'Etudes et Prospective.
- *GPQ*: Groupe de contrôle de gestion, d'étude et de qualité.
- *GSP*: Groupe Stratégie des Projets NV.
- *AGL*: groupe d'Atelier de Génie Logiciel

La mission du groupe GEP, *cadre de mon tutorat, est de réaliser des études de faisabilité, des maquettes ou prototypes sur les méthodes et outils supports au développement de logiciels. C' est dans cette perspective que s' est inscrit mon étude. La partie suivante aborde les caractéristiques de ce travail .*

I.2 Objet de l'étude

I.2.1 L'ingénierie des protocoles

Evolution des systèmes informatiques

Il est indubitable que cette fin de siècle est marquée par le renforcement de la technologie, notamment de l'informatique et des télécommunications. Les besoins de faire communiquer entre-elles des machines disparates et complexes, afin d'échanger de l'information, dont le spectre ne cesse de s'élargir, se font de plus en plus pressants. Cela se traduit notamment par l'émergence des réseaux et des services télématiques. Les systèmes ouverts sont donc de première nécessité pour répondre à des exigences croissantes en matière de fonctionnalités mais aussi de coûts.
Mais comme dans tout dialogue, il est nécessaire de diriger les échanges par un ensemble de règles. Ces règles ou protocoles sont dès lors l'objet d'une véritable étude du fait de leur rôle crucial dans l'exploitation efficace des réseaux.

Intérêt de la vérification des protocoles

Les protocoles peuvent être considérés comme des algorithmes distribués à travers les différentes entités intervenantes dans la communication : de manière très abstraite, appelant et appelé dans le cadre de la téléphonie par exemple. Ces règles de dialogue se font de plus en plus complexes et diversifiées, pour s'adapter aux nouvelles exigences.
Cette complexité est de nature diverse [*Gro89*]. Elle résulte, d'une part, d'une certaine difficulté de maîtrise intellectuelle du comportement dynamique de ces protocoles. En effet l'aspect parallélisme distribué implique que le système peut entrer dans un grand nombre d'états et il devient beaucoup plus difficile a appréhender que dans le cas de traitements purement séquentiels. De plus, la difficulté revient a mettre en rapport des machines hétérogènes et qui a priori n'ont pas été prévues toujours dans cette première optique.
Tout informaticien sait que l'aspect vérification fait partie intégrante de la phase d'élaboration d'un logiciel. Cela devient même primordial dès lors que les contraintes économiques ne sont pas négligeables. Il est alors impératif d'assurer la meilleure fiabilité possible, et une totale conformité par rapport au cahier des charges, et ce dès les premières phases de conception.
Du fait des principales complexités citées précédemment, il apparait que c'est encore plus vrai dans le domaine de la conception de protocoles, ou le seul esprit humain ne suffit pas a envisager l'ensemble des scénarios possibles. La vérification revêt donc ici tout son intérêt.

Normalisation et architecture OSI

Pour solutionner une partie du problème et permettre à des systèmes hétérogènes ou non de communiquer, un travail de normalisation à été entrepris par deux organismes internationaux, l'ITU (International Telecomunication Union, anciennement CCITT) et l'ISO (International Standardization Organisation) afin d'établir un modèle de référence. Ceux–ci ont développé le concept d'architecture ouverte OSI (Open System Interconection) (*fig. 1*) .Cette structure permet, entre autre, d'optimiser les échanges entre machines ou terminaux, une certaine homogénéité entre couches fonctionnelles et une indépendance service / réseau [*Bou93, Zim85*].

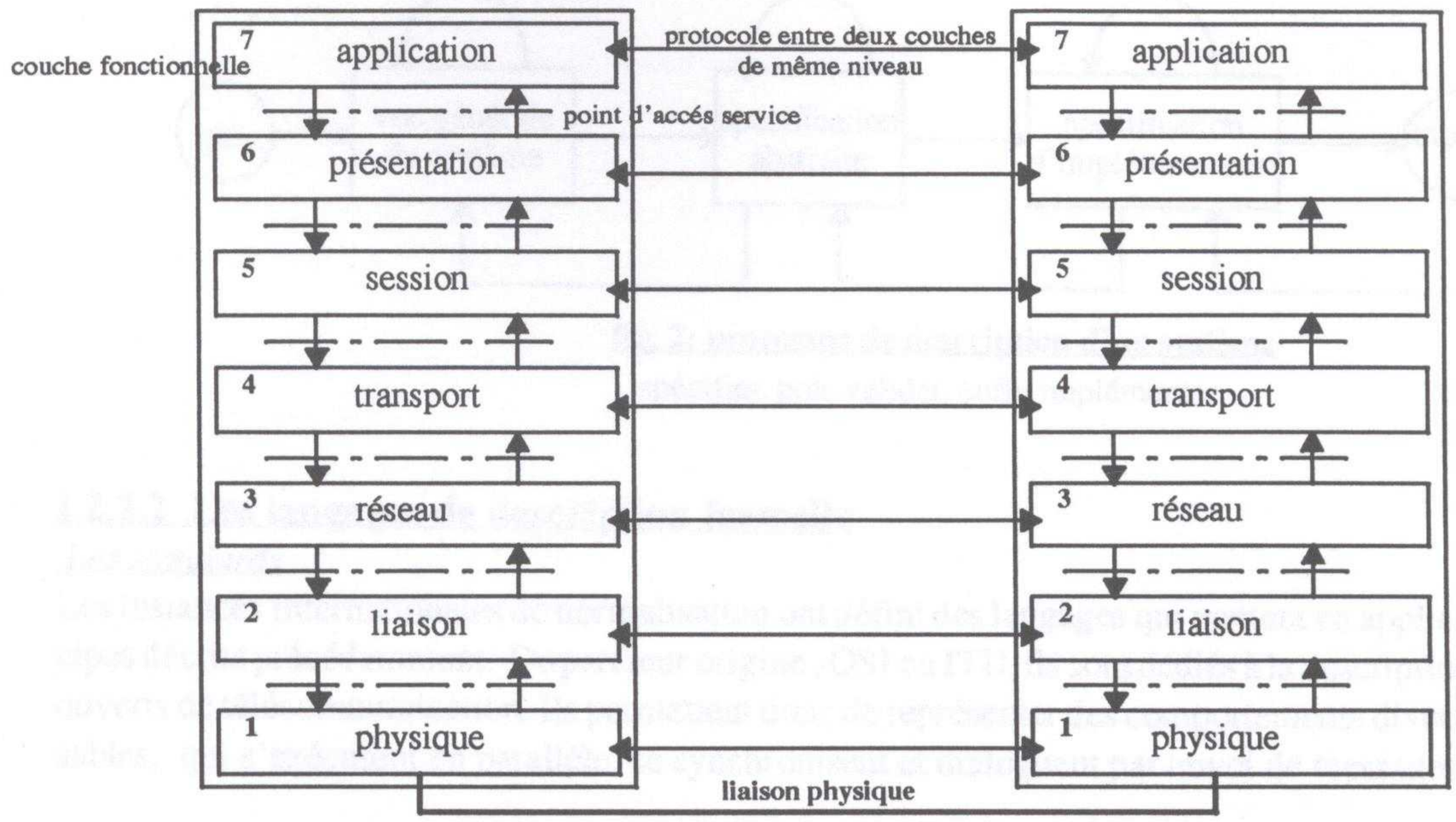


fig.1: modèle de référence OSI

Les comités de normalisation éditent également des descriptions de services et de protocoles à partir de cette structure, dans le but de constituer des références pour l'implémentation. Mais ces informations n'intègrent pas de véritable formalisme de part leur représentation sous forme de langage naturel ou de diagrammes de comportement. Elles entraînent donc des définitions volumineuses, sujettes aux différentes interprétations des utilisateurs et d'autant plus entachées d'erreurs qu'elles sont difficilement vérifiables.

Paradoxalement donc, la conception des protocoles reste encore artisanale dans beaucoup de cas. Alors que la rigueur est de mise, la plupart des réalisations sont effectuées sans supports véritablement formels. La qualité des réalisations s'en ressent fort dommageablement.

I.2.2 La spécification formelle

I.2.2.1 L'ingénierie des spécifications formelles

Afin de pallier ces inconvénients, de nouvelles techniques de descriptions plus rigoureuses ont été recherchées. Les organismes de normalisation que représentent l'ITU et l'ISO ont conçus des méthodes utilisant un plus grand formalisme : *les techniques de description formelles*, dèsignées par le terme générique FDT (Formal Description Technique) [*For90*].

Celles-ci permettent de spécifier des comportements de systèmes de télécommunication de manière rigoureuse et complète, tout en gardant un certain niveau d'abstraction. Cela aboutit à des descriptions hors-implémentation, destinées éventuellement à des cibles hétérogènes. A l'inverse du langage naturel humain, les spécifications résultantes se veulent concises, structurées et non-ambiguës, de part la syntaxe et la sémantique précise des FDT.

Elles servent alors de fondation pour une vérification dès la première phase de développement mais aussi de support d'implémentation (*fig. 2*).

Des constats établis à un niveau nécessitent parfois de revenir une ou plusieurs étapes en arrière. L'intérêt des FDT est bien entendu d'éviter tant que possible ce procédé. Deux approches sont possible pour la représentation de systèmes distribués : *l'approche modèle* et *l'approche linguistique* [*Ing93*]. La première d'entre-elles utilise des modèles mathématiques tels que les réseaux de Pétri. La seconde est basée sur un ensemble de construction plus vaste bien que s'appuyant sur un ou plusieurs modèle(s) mathématique(s). Dans le paragraphe suivant, nous nous intéresserons plus à ce second aspect.

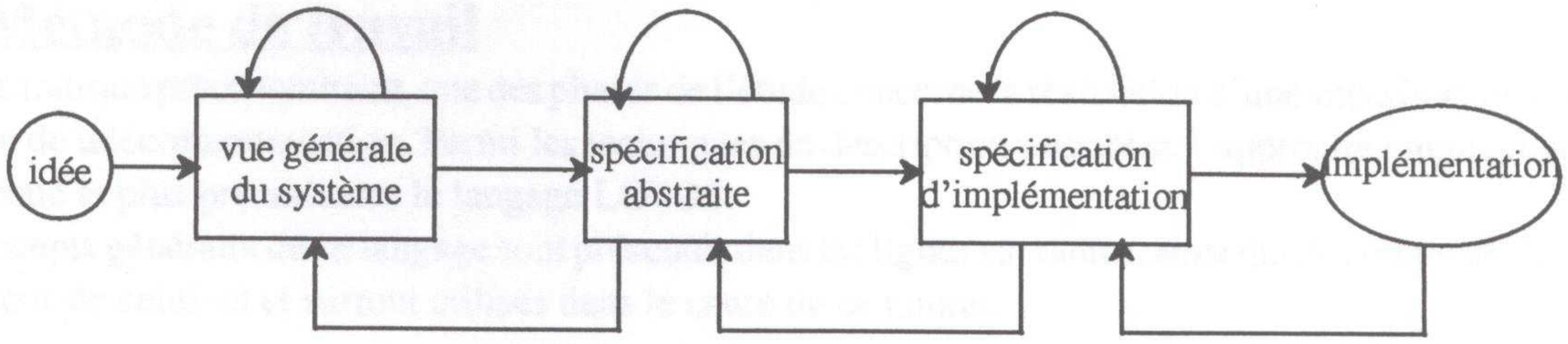


fig. 2: processus de description d'un système
spécifier, puis valider, enfin implémenter

I.2.2.2 Les langages de description formelle

Les standards

Les instances Internationales de normalisation ont défini des langages qui mettent en application les principes décrits précédemment. De part leur origine , OSI ou ITU, ils sont dédiés à la description de systèmes ouverts de télécommunication. Ils permettent donc de représenter des comportements divers et décomposables, qui s'exécutent en parallèle, se synchronisent et dialoguent par envoi de messages.

Trois standards ont été établis de part leur normalisation par l'un ou l'autre des organismes : Ainsi *SDL* (Specification and Description Language) fait l'objet de la norme ITU alors que *ESTELLE* et *LOTOS* (Language of Temporal Ordering Specification) sont liés à l'ISO [*Ing93*].

Les outils d' exploitation

Sans outils supports au développement, l'intérêt de ces langages ne s'en retrouverait que réduit. Le produit réalisé ne constituerait alors qu'une autre forme de représentation d'un système, certes plus concise et moins ambiguë que ces devancières (descriptions non formelles), mais sans véritable forme de vérification possible.
Néanmoins cela n'est pas le cas et le formalisme employé permet de concevoir des moyens d'exploitation de la spécification dans divers buts : assistance à la conception, ou analyse du comportement dynamique. Ces outils sont répertoriés selon la classification suivante [Gar89]:

- *les analyseurs* : éditeurs syntaxiques et/ou graphiques, analyseurs de syntaxe et de sémantique statique.
- *les interpréteurs* : ou simulateurs. ils permettent d'exécuter symboliquement une spécification en mode pas à pas.
- *les générateurs de code* : à partir d'une spécification, ils produisent un programme exécutable.
- *les vérificateurs* : permettent de mettre en évidence des propriétés spécifiques et s'appuient sur un modèle mathématique.

I.2.3 Cadre du travail à réaliser

C' est dans ce contexte que s' inscrit mon tutorat. Le sujet touche concerne plus particulièrement une étude de certains outils supports au développement, notamment dans la gamme des vérificateurs. Cette étude nécessite de la matière et elle fait donc suite à la conception d' une spécification d' un système de télécommunication, à l' aide d' une technique de description formelle.
Cette définition du sujet se veut volontairement abstraite pour ne pas noyer le lecteur dans les concepts. Chacune de ses caractéristiques est abordée, tour à tour, dans les prochains paragraphes à travers trois aspects : la méthode de travail soit la présentation du langage de description et des outils utilisés, le domaine d' étude à savoir l' objet de la spécification et enfin les supports de référence à la réalisation.

I.3 Méthode de travail

Comme indiqué précédemment, une des phases de l'étude concerne la réalisation d'une spécification d'un système de télécommunication. Parmi les techniques de description possibles, l'approche linguistique a été retenue et plus précisément le langage LOTOS.
Les concepts généraux de ce langage sont présentés dans les lignes suivantes, ainsi que les outils développés autour de celui-ci et surtout utilisés dans le cadre de ce tutorat.

I.3.1 Le langage LOTOS

I.3.1.1 Généralités

Le langage LOTOS [Lot87, Loy92] fait l'objet d'une normalisation de la part de l'organisme international ISO en 1988. De part cette origine, il est dédié à la représentation de systèmes ouverts.
LOTOS se caractérise par:

- *sa définition formelle,*
- *sa puissance d' expression,*
- *son degré d' abstraction,*
- *sa structure précise.*

LOTOS a pour principe la description de l'ordonnancement d'événements dans le temps. Le système décrit est donc vu de part les échanges de synchronisations entre les entités qui le composent.

I.3.1.2 Les spécifications LOTOS

Il existe en LOTOS une nette séparation entre données et contrôle. La partie données repose sur les types abstraits algébriques alors la partie contrôle met en jeu une algèbre de processus utilisant ces mêmes données. Les possibilités offertes par LOTOS sont riches et le but des lignes suivantes n'est pas de procéder à une présentation exhaustive, mais de présenter les concepts de bases de ce langage.

La partie données

Dans un soucis de description hors-implémentation et de réduction de la complexité, les données sont définies de manière abstraite sous la forme de types algébriques. Cette technique s'articule autour du concept de *type* qu'il est possible de rapporter à la notion d'objet dans les autres langages.
Dans cette optique, seules les propriétés essentielles sur les types sont définies : *les opérations*. Les aspects liés à la représentation interne et à la manipulation en mémoire de ceux-ci sont laissés de côté. Un type est donc décrit autour des opérations, dont les résultats sont des occurrences valides de celui-ci. Dans cet objectif, des constructeurs permettant d'obtenir des occurrences de base sont fournis dans un premier temps. Puis les opérations associées à ces constructeurs pour en engendrer de nouvelles sont définies à leur tour.

Les manipulations offertes autour de ce concept de type sont importantes :

- *Une librairie de types standards prédéfinis* est disponible pour les types de bases tels que NATURAL NUMBER, BOOLEAN,SET,BIT,...
- *Des opérations sont possibles sur les types* telles que les combinaisons, le renommage ou l'actualisation pour les types paramètrés (décrits partiellement).
- *Les équations conditionnelles sont autorisées.*
- La surcharge d'opérateurs est permise : opérateurs de même nom pour des profils différents.
- *LOTOS possède la notion d' héritage multiple* : importation dans un type des sortes, opérations et équations définies dans d'autres types.

La partie contrôle

La partie comportementale de LOTOS est construite autour d'une algèbre de processus : chaque processus décrit un comportement à l'aide d'expressions mathématiques, et d'opérateurs de contrôle fournis par ce langage. Celui-ci peut être décomposé en sous-processus, et ainsi de suite, pour aboutir à une hiérarchie.

Leur exécution peut se dérouler en parallélisme asynchrone ou synchrone à divers niveaux; les échanges s'effectuant par rendez–vous.

Portes et offres

Un comportement élémentaire correspond à une interaction entre deux comportements parallèles synchronisés. Celle–ci est définie autour d'une *porte* et d'une *liste d' offres*. La porte permet la synchronisation entre plusieurs processus et les offres d'échanger des informations (données issues des types abstraits) et donc de simuler (nuance importante) l'envoi ou la réception.

Par exemple, l'interaction suivante OUTPUT !TRUE signifie un rendez–vous par le biais la porte OUTPUT sur l'offre TRUE. OUTPUT ?X:BOOL dénote par contre un rendez–vous à travers la porte OUTPUT sur une valeur de sorte BOOL (TRUE ou FALSE par exemple). Mais harmonisation et passages de valeurs peuvent être combinés comme suit : OUTPUT !1 ?X:BOOL ?Y:BOOL ...
Enfin, trois interactions spécifiques sont prédéfinies :

- *i : action interne non observable,*
- *stop : inaction,*
- *exit : terminaison avec succès.*

Opérateurs

L'un des grands intérêts de LOTOS provient de la richesse de ses opérateurs de combinaison d'expressions comportementales. Ils permettent de traduire simplement des comportements complexes de systèmes de télécommunication tel que le parallélisme synchrone, ainsi que parvenir à une spécification plus claire et concise.

Ces opérateurs sont les suivants :

- *opérateur de séquentialité* : a;B
  a est une interaction et B une expression comportementale séquentielle à a.
- *opérateur de choix* : B1[]B2
  B1 et B2 sont des expressions comportementales.
- *opérateur de parallélisme* :
  - asynchrone : B1|||B2
    B1 et B2 s'exécutent en pure parallélisme.
  - synchrone partiellement : B1|[g1,...,gn]|B2
    B1 et B2 sont des expressions synchronisées sur les portes g1,...,gn.
  - totalement synchrone B1||B2
    B1 et B2 sont synchronisées sur l'ensemble de leur porte communes.
- *opérateur de composition séquentielle :* B1 >> B2
  si B1 termine avec succès alors B2 est déroulé.
- *opérateur de préemption :* B1 [> B2
  interruption possible du fonctionnement de B1 par le premier évènement de B2 sans possibilité de reprise
- *opérateur de masquage hide :* hide [g1,...,gn] in B
  g1,...,gn sont des portes de B invisibles au niveau hiérarchique supérieur.

I.3.2 Les outils supports

Les outils développés autour du langage LOTOS sont issus de projets de recherche universitaire et sont encore à l'état de prototype. Néanmoins plusieurs d'entre eux sont disponibles et leur nécessité n'est pas à sous–estimer puisque n'oublions pas que le fondement des techniques formelles est, à la base, le besoin de vérifier les protocoles et que sans ces outils, il n'existe point de vérification.
Ces outils ont un rôle central dans ce tutorat, puisque l'un des objectifs est d'évaluer le panel de fonction-

nalités qu'ils offrent à l'utilisateur, à des fins d'exploitation de la spécification.
Les outils utilisés peuvent être classés selon leur fonctionnalités : *les interpréteurs*, *les compilateurs* et *les vérificateurs*.

I.3.2.1 Les interpréteurs ou simulateurs

Ils permettent d'exécuter symboliquement la spécification et donc d'observer le déroulement du comportement en mode pas à pas. A tout moment l'utilisateur peut s'informer du scénarioso déroulé depuis le début de la simulation. Il peut également franchir des séquences d'actions, inspecter l'état des variables ou des processus décrivant le contrôle.
A noter qu' une évaluation des types est possible et que cela n'est pas négligeable puisque ceux-ci peuvent être la source d'erreurs tout autant que la partie contrôle.
Deux outils de cette catégorie sont utilisés : *hippo* et *smile*.

Hippo 2.1 [*Hip87*]
Il s'agit d'un simulateur interactif qui permet uniquement le mode pas à pas.

Smile 3.1 (environnement LTB 1.0)[*Smi93*]
Il s'agit là encore d'un simulateur interactif. Mais plus convivial qu'hippo (environnement multi-fenêtrage), il ne restreint pas la simulation au mode pas à pas : des points d'arrêts sont marquables. D'autre part, alors qu'avec hippo les variables issues des offres sont à instancier dès leur rencontre, elle peuvent être retardées voire même non réalisées si non nécessaire.
Smile est intégré à une boite à outil configurable : LTB qui permet à l'utilisateur de faire également appel à un éditeur (textuel ou graphique), un analyseur (syntaxique et sémantique statique), ou encore un compilateur.

I.3.2.2 Les compilateurs

Ils permettent de générer du code exécutable dans un soucis d'implémentation indirecte via les techniques formelles suite à une vérification. Seul topo a été abordé dans cette ensemble.

Topo 1R7 [*Top91*]
Ce compilateur LOTOS vers C ou ADA peut être aussi utilisé à des fins de simulation, et donc de vérification. Mais son utilisation est trés contraignante puisqu'elle nécéssite l'intégration d'annotations C au niveau de la spécification : Pour l'implémentation des types et des opérations en ce qui concerne la partie données; pour diriger la compilation ou la simulation en ce qui concerne le contrôle. De plus, l'utilisateur se doit d'écrire un module d'interface C. Ces ajouts étant jugés trop contraignants dans le seul but de la simulation, cet outil fût peu utilisé.

I.3.2.3 Les vérificateurs

Ces outils offrent les moyens de passer du stade de la simple vérification par simulation interactive, à celui de la validation de la spécification et du système décrit. La nuance est importante puisqu'elle fait apparaître l'aspect complétude de l'analyse de la description.
Les vérificateurs servent entre autre à rechercher certaines propriétés à travers la spécification, et à détecter des erreurs de conception (telles que la non adéquation d'un protocole à un ensemble des services).
Parmi les approches possibles, seule la génération de modèle est utilisée par l'intermédiaire de la boite à outils caesar/aldebaran [*Boi94*] (autre: preuve automatique).

La boite à outils caesar/aldebaran
La spécificité de cette approche est de traduire, lorsque c'est possible, la spécification à valider vers un modèle graphe qui caractérise sa sémantique dynamique. dès lors toute évolution possible du système est accéssible à travers ce modèle. La présente boite à outils intègre deux types de fonctionnalités : *la compila-*

tion de la spécification et *sa vérification*
La *compilation* n'engendre pas du code directement exécutable comme c'est le cas dans la classe des compilateurs décrite précédemment. Celle–ci consiste plutôt à traduire la description vers un modèle dans le but d'une vérification via une simulation.Cette traduction s'opère à deux niveaux : les données et le contrôle. Deux compilateurs sont donc fournis, à titre différent, pour traduire la spécification : *caesar.adt* et *caesar*. La *vérification* repose sur une analyse du modèle graphe engendré par caesar. Deux types d'outils sont utilisés dans cette optique et à des étapes différentes : *aldebaran* et o*pen/caesar.*

Caesar.adt 4.2
L'implémentation des types abstraits algébriques consiste à produire une bibliothèque en langage C, contenant pour chaque sorte (resp. opération) un type concret (resp. une fonction) qui lui est associé. Cette étape peut être effectuée de deux manières : l'utilisateur a la possibilité de décrire ses propres implémentations ou d'en utiliser des existantes; mais la génération peut être également automatique à l'aide de l'outil caesar.adt. Les deux aspects peuvent être combinés.
D'autre part, une vérification des types abstraits peut être réalisée là encore automatiquement à l'aide de ce dernier(mais plus sur la forme que sur le fond).

Caesar 4.8
Caesar peut être considéré comme le "noyau" de la boite à outil caesar/aldebaran. Il a pour tâche la traduction de la partie contrôle vers le modèle graphe sous des formats différents pour interfacer un certain nombre d'outils avals existants [*Cae94*].
Deux modèles sont présentés en sortie de traitement : un modèle réseau de Pétri interprété et un modèle graphe. Celui–ci peut également être utilisé en tant qu'analyseur syntaxique et sémantique, plus contraignant que dans le cas d'hippo ou smile, du fait de la génération de modèles.

Aldebaran 5.6
Ce vérificateur est utilisé en aval de caesar, une fois le graphe complet généré. Il a deux deux fonctions: *la réduction de graphe* et *la comparaison de deux graphes modulo certaines relations d' équivalence.*

L' environnement open/caesar 1.6
Afin d'offrir la possibilité à l'utilisateur d'intervenir au niveau de la génération de graphe, caesar a été étendu et son environnement "ouvert".
Par le biais de cet outil, celui–ci a les moyens de suivre la progression dans la conception du modèle (et éventuellement de s'en servir comme support de génération), voire même d'y inférer (restrictions sur le développement, contraintes sur le parcours...).

De part les intérêts qu' elle présente à priori dans l' objectif d' une vérification, une large place a été faite à cette boite à outils lors l' analyse de la spécification. Son large panel de fonctionnalités ainsi que les caractéristiques, avantages et inconvénients de chacune des approches décrites précédemment font l' objet de paragraphes dans la suite de ce présent rapport.

I.4 Le domaine d'étude

Le paragraphe précédent fait état de la technique formelle utilisée pour réaliser la spécification. Cette spécification nécessite également de la matière : un système de télécommunication dont on souhaite décrire et étudier le comportement.
En se référant à l'architecture multi–couches OSI, il apparait que deux approches sont possibles dans le traitement l'interconnexion de systèmes ouverts : *l' aspect protocole* et *l' aspect service* pour chacune des sept couches.
La spécification n'est pas vue comme une fin en soi (le but n'est pas la vérification du système ni la réalisation d'une spécification de référence) dans ce tutorat mais plutôt comme une étape nécessaire afin de jau-

ger certains outils supports. L'aspect protocole jugé inutilement complexe, l'approche service a donc été retenue. De plus, la séparation de l'information et de la signalisation (par canal sémaphore), implique une vision des protocoles sous deux formes pour chaque couche : le protocole de transfert d'information et le protocole de signalisation. Ceci pour témoigner de la complexité qui serait celle de la spécification résultante.
Le domaine d' étude choisi à pour cadre le RNIS (Réseau Numérique à Intégration de Services).

I.4.1 Les services du RNIS

Le RNIS, par le biais de la numérisation, permet le transport d'un large spectre d'information, sous forme de données, son ou image, et donc de répondre de façon adaptée à des besoins de plus en plus différenciés. Les fonctionnalités offertes aux souscripteurs sont répertoriés selon trois classes [*Rni88*]:

- *les services supports,*
- *les téléservices,*
- *les services supplémentaires.*

les deux premières catégories peuvent être offertes seules (services de base) alors qu'un service supplémentaire accompagne obligatoirement un des deux précédents.

I.4.1.1 Les services supports

Ces services sont relatifs aux couches basses de l'architecture OSI (physique,liaison et réseau) et sont offerts à l'interface ente l'usager et le réseau. Ils permettent d'établir une connexion entre deux systèmes ouverts.
Leur caractérisation se fait par l'intermédiaire d'attributs concernant entre autre le transfert d'information ou encore l'accès au réseau: *mode, débit ,capacité de transfert, structure, établissement, configuration, symétrie, canaux, ou encore protocoles de signalisation et d' information dans D pour chacune des trois premières couches.*

I.4.1.2 Les téléservices

Les téléservices sont offerts à l'usager en aval du terminal et comprennent les fonctions relatives à l'ensemble des couches basses et hautes (1 à 7). La caractérisation de ces services se fait en adjoignant à l'ensemble des attributs précédents, un attribut de type d'information (son, parole, texte, vidéo,....) ainsi que les identifications des protocoles pour les niveaux supérieurs (4 à 7).

Les téléservices standardisés offerts à l'abonné RNIS incluent:

- *le téléphonie 300–3400Hz,*
- *le télétex : service courrier électronique pour documents dactylographiés,*
- *la télécopie groupes III et IV : échange de documents graphiques,*
- *le mode mixte : télécopie + télétex,*
- *l' audiographie : échange vocal + document manuscrit (tablette d' écriture),*
- *l' audiovidéotex : renforcement du service vidéotex actuel (minitel).*

P*armi cet ensemble, le cadre de la spécification concerne la téléphonie RNIS*. Ses caractéristiques sont abordées dans le sous-paragraphe suivant.

La téléphonie rnis

Ce service est fondamentalement équivalent dans son objectif premier de communication vocale interpersonnelle au traditionnel service délivré par les réseaux analogiques. Par contre, il se différencie de son "ancêtre" de part les possibilités accrues. Celles–ci ont pour origine quatres révolutions en terme :

- *d' interfaces usager–réseau,*
- *de structure des installations terminales d' abonnés,*
- *de protocole de communication,*

– *de services additionnels proposés.*

Ces services additionnels sont les véritables témoins de la formidable évolution actuelle en matière de téléphonie. Ils font l'objet du prochain paragraphe.

I.4.1.3 Les services supplémentaires

Ces services sont vus comme des facilités associées à l'appel de base afin d'en améliorer le confort d'utilisation et la souplesse d'exploitation.

Les compléments propres à la téléphonie sont répartis en cinq groupes selon leur rôle:

– *la simplification de l'usage du terminal en cours de communication,*
ex: portabilité, rappel du dernier numéro...

– *l'amélioration des situations d'attente,*
ex: appel en instance, présentation systématique ..

– *la gestion des situations d'absence,*
ex: renvois d'appel, terminal en répondeur ...

– *la présentation d'informations additionnelles,*
ex: identification d'appel, taxation, ...

– *autres.*
ex: conférence, secret d'identité,...

Le système de télécommunication à spécifier intègre, en plus de l'appel téléphonique de base, le service supplémentaire d'appel en instance. Ses principes sont entrevus par la suite.

L'appel en instance

L'objectif de ce service est d'assouplir le traitement d'un appel dans le cas d'un impossibilité de prise en charge au niveau de l'installation terminale appelée. Son principe est d'informer un abonné de la présence d'un appel le concernant mais qu'il ne peut pas prendre en ligne dans l'immédiat. L'appel est alors mis en instance.

Deux types d'indisponibilités peuvent apparaître et engendrer l'invocation de ce service :

– *une indisponibilité du terminal téléphonique appelé.*
le terminal appelé est déjà en communication ou ses possibilités d'accès font l'objet d'une réservation (maintien d'appel invoqué par exemple)

– *une indisponibilité des canaux d'informations* au niveau de l'interface usager–réseau

Lorsqu'un appel fait face à une telle situation, le service est invoqué et l'ensemble des usagers en est averti. Ceux–ci peuvent alors rejeter le service ou patienter dans le cas de l'appelant que le terminal appelé puisse le prendre en ligne, suite à une libération de la communication au niveau de ce dernier ou à une libération de canaux au niveau de son installation.

I.4.2 Objet de la spécification

Le système de télécommunication, faisant l'objet d'une spécification formelle, s'inscrit donc dans le contexte des fonctionnalités du RNIS. Il a pour objet le téléservice de la téléphonie RNIS et le service supplémentaire d'appel en instance.

Le système décrit permet la gestion de liaisons téléphoniques de base simultannées, concurrentes ou non. D'autre part, en cas d'impossibilité de prise en charge d'un appel, celui–ci est mis en instance et le terminal appelé est averti de sa présence pour un traitement ultérieur éventuel.

I.5 Les supports de référence

Face au risque de divergence incontrôlée provoquée par l'évolution et la diversité de systèmes hétérogènes, un certain nombre d'organismes internationaux établissent des normalisations sur l'ensemble des aspects protocoles, services et interfaces de télécommunication.

I.5.1 Les recommandations de l'ITU sur le RNIS

Parmi les comités crées, l'ITU (International Telecomonication Union) a entrepris de grands travaux de normalisation depuis 1984 et la multiplication des assemblées a aboutit à la parution du livre rouge, puis du livre bleu dans un second temps.
Ces ouvrages renferment un large spectre de définitions sur les aspects des télécommunication et notamment le RNIS. Les recommandations concernant le RNIS sont organisées autour de la *série I* de la commission d'études XVIII.
La spécification sur la téléphonie de base RNIS et le service supplémentaire d'appel en instance ne saurait être réalisée sans l'aide de supports de référence nécessaire à la garantie de la conformité de la description vis à vis du comportement réel du système.
Les services sont abordès au niveau de la *série I.200*. Celle–ci est elle–même décomposée en sous–séries de la façon suivante :

- *guide d'utilisation de la série,*
- *définition du concept de service,*
- *aspects commun des services,*
- *vue générale sur les services supports, de base et supplémentaires,*
- *description individuelle des services pour chacune des trois catégories.*

Les informations contenues dans ces documents regroupent des définitions générales ainsi que la description statique et dynamique des services du RNIS. Ces données sont présentées sous la forme de prose pour ce qui est des définitions et des descriptions statiques, alors que les comportement dynamiques prennent la forme de diagrammes SDL, véritables références pour l'élaboration de la spécification formelle.
Les recommandations sur les services de téléphonie et d'appel en instance sont répertoriées en annexe A.

I.6 Définition complète du sujet

Après que la nécessité de la vérification de protocole ait été mise en évidence dans un premier temps, le paragraphe I.2.3 *présente de façon générale le cadre du travail à réaliser : spécification formelle d'un système de télécommunication à l'aide langage de description, et utilisation d'outils, supports au développement, associés à celui–ci.*
Par la suite la tâche à réaliser à été détaillée à travers trois de ses composantes :

la méthode : le langage de description formelle LOTOS,
et les outils hippo, smile, topo et la boite à outils caesar/aldebaran.
le domaine : le téléservice de la téléphonie RNIS,
et le service supplémentaire d'appel en instance.
les supports de référence : les recommandations de l'ITU sur le RNIS.

L'objet de l'étude intègre donc la réalisation d'une spécification formelle des services de téléphonie RNIS et d'appel en instance, en LOTOS, en conformité avec les considérations sur ces deux fonctionnalités telles qu'elles sont décrites dans les recommandations de l'ITU.
Cette description n'a pas de finalité en soi (spécification formelle de référence pour d'autres développements par exemple) mais sert de matière à une étude de faisabilité sur certains outils conçus autour de LOTOS ou plus généralement des langages de description formelle. Ceux–ci ont pour noms hippo, smile, topo, caesar, aldebaran ou l'environnement open/caesar et offrent un panel très diversifié de possibilité d'exploitation de spécification, notamment dans le cas de caesar (de la simple utilisation d'outil à la conception de modules d'exploration de graphes).

II . Conception de la spécification

II.1 Fonctionnalités du système à spécifier

II.1.1 Définitions élémentaires et hypothèses

Le présent paragraphe définit des terminologies spécifiques au domaine de la télécommunication. Ces concepts ne présentent aucune difficulté quant à leur compréhension mais leur présentation à priori semble nécessaire du fait de leur caractère incontournable.

Equipement terminal: ET

Un équipement terminal (ou terminal de communication) est une interface directe entre l'usager et le réseau auquel il est connecté, et qui lui permet d'accéder au large spectre de services auxquels il a souscrit. Dans notre cas, la téléphonie RNIS, cette terminologie ne se limite pas au "téléphone" mais s'élargit à tout terminal, numérique ou non (via une adaptation), pouvant invoquer ce service : terminal téléphonique numérique ou analogique, terminal multimédia (station de travail bureautique par exemple).

Installation Terminale d'Abonné: ITA

une ITA caractérise l'architecture et les constituants de l'installation d'abonné. Elles se différencient selon plusieurs critères :

- *l'interface d'accès: le nombre et la nature des liaisons de raccordement au réseau,*
- *la présence ou non d'organes assurant des fonctions de commutation privée,*
- *la topologie de l'installation,*
- *le nombre et la nature des terminaux raccordés*

Dans notre cas, la spécification réalisée fait appel à un certain degrés d'abstraction puisque l'un de ses intérêt est de décrire le système hors-implémentation. Cette abstraction se retrouve dans la quasi-totalité de ces niveaux : topologie, terminaux, et commutation. Ce n'est par contre pas le cas de l'interface d'accès, qui doit être connue pour chacune des ITA intervenantes dans le système, de manière à pouvoir gérer sa capacité à traiter un appel.
Ce caractère évolutif permet donc d'adapter les installations aux différents styles d'appels retenus afin de parvenir à une analyse plus complète des services (et donc une vérification plus exhaustive).
Il est bien entendu évident que chaque installation à l'origine d'une communication de type téléphonique ou donnant réponse à la présentation de l'appel, se doit de posséder au moins un terminal offrant la possibilité de traiter ce service!

Appel ciblé et appel diffusé

Deux types de requêtes téléphoniques sont offertes à l'usager appelant : *appel ciblé* et *appel diffusé*.
Dans le premier cas, l'usager désire obtenir une liaison avec un poste bien précis d'une installation terminale : il adjoint donc à sa requête d'appel l'adresse réseau de l'ITA voulue ainsi que la sous-adresse du terminal ciblé.
Dans l'autre cas, l'usager ne pose pas de contraintes quant à l'identification du terminal appelé et seule l'adresse de l'ITA ciblée est indiquée. Dés lors, tout terminal ayant les capacités à traiter ce type de service représente un "interlocuteur potentiel", sous réserve de configurations particulières de la part du correspondant, comme l'utilisation de la fonction opératrice (acceptation d'appel que sur un seul service) par exemple.
D'autres cas de diffusion sont possibles, ayant notamment pour origine l'ITA appelante où dans ce cas, selon les services supplémentaires retenus par l'abonné appelé, un appel initialement ciblé peut être diffusé vers un sous-ensemble des terminaux. Ce type de transformation n'est pas considéré dans un premier temps dans notre cas, mais comme tout service supplémentaire, il peut être incorporé par la suite.

Interface usager–réseau

l'interface usager–réseau incarne l'une des richesses du RNIS. Elle est organisée en canaux [*Rni88*]. Un canal représente une portion définie et stable de la possibilité de transport d'information d'une interface. On en distingue principalement deux types:

– *le canal B* : canal à 64 kbit/s, commuté en mode circuit et utilisé pour transporter l'information.
– *le canal D :* canal à 16 ou 64 kbit/s, commuté en mode paquet, apte à transporter la signalisation.

La structure d'interface représente l'organisation en canaux de l'interface usager–réseau et elle détermine la capacité d'accès d'une ITA.
On distingue principalement deux type d'accès :

– *l'accès de base :*
débit util de 144 kbit/s pour une interface en 2B+D (deux canaux d'information à 64 kbit/s et un canal D à 16kbit/s)
– *l'accès à débit primaire:*
débit utile de 2048 kbit/s pour une interface en 30B+D (30 canaux d'information à 64 kbit/s et un canal de signalisation à 64 kbit/s) ou de 1544 kbit/s pour une interface en 23B+D avec les même caractéristiques de canaux.

Dans le cas de la spécification réalisée, la gestion de la disponibilité des canaux d'information, au niveau de chacune d'une ITA, est une nécessité pour déterminer la capacité de cette installation à prendre en charge l'appel dont elle fait l'objet.
Dans un soucis de simplification de gestion de cette ressource et dans la mesure où la diversité n'apporterait pas un plus, l'ensemble des interfaces est supposé structuré en 2B+D. Dans le même sens, il est supposé qu'un seul canal B est attribué à chaque appel, et que chaque ITA ne dispose que d'une seule interface (à débit de base).

II.2 L'orientation des recommandations

Les recommandations de l'ITU, utilisées comme support de référence à l'établissement de la spécification, servent à orienter la structuration. Du point de vue de l'aspect dynamique des services, ces documents présentent le comportement global du système à travers un ordonnancement temporel de primitives d'échanges et de tâches. ils identifient ainsi toutes les actions du système perceptibles du point de vue de l'abonné au niveau de l'interface avec le réseau (interactions usager–réseau).

II.2.1 Raccordement des usagers au RNIS

Le raccordement des usagers au RNIS est représenté autour de deux concepts : les points de références et les entités fonctionnelles (*fig. 3*) [*Com87*].

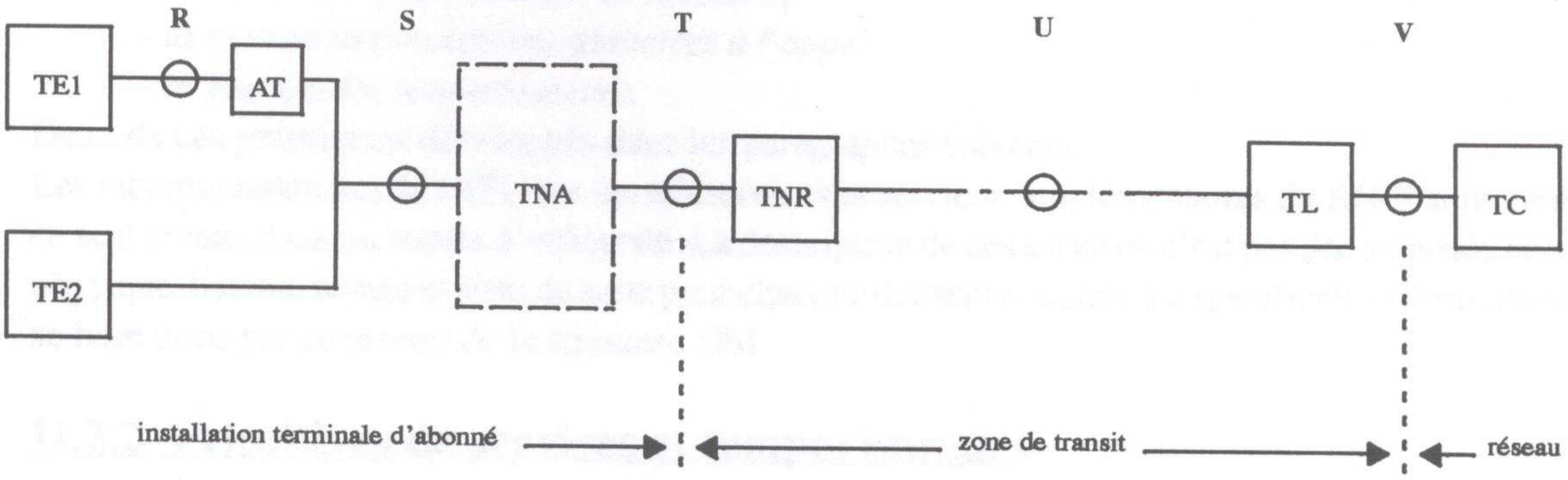
Les points de référence

Un point de référence est un point théorique séparant deux groupements fonctionnels: selon la réalisation, il correspond à une interface physique ou non.
L'interface usager–réseau du RNIS est l'interface physique correspondant au niveau S/T. Les points S et T sont regroupés de cette sorte du fait de l'aspect non obligatoire de l'entité TNA.
Le point de référence R est l'interface physique entre un équipement terminal non numérique et l'adaptateur qu'il requiert. Enfin les points U et V représentent successivement la ligne de transmission et l'interface avec le commutateur public.

Les groupements fonctionnels

Il s'agit d'ensemble de fonctions regroupées à un même endroit .
On répertorie :

– *le terminal,*



TE: terminal
AT : adaptateur de terminal
TNA: terminaison numérique d'abonné
TNR: terminaison numérique de réseau
TL: terminal de ligne
TC: terminal de commutation
○ point de référence
□ entité fonctionnelle

fig. 3: raccordement d'usager au RNIS: configuration de référence

- *l'adaptateur de terminal,*
- *la terminaison numérique d'abonné,*
- *la terminaison numérique de réseau,*
- *le terminal de ligne,*
- *le terminal de commutation.*

Les recommandations de l'ITU sur les services du RNIS décrivent le flux d'information au niveau des points de référence S/T (*fig. 4*), par le biais de primitives et de tâche, et de manière globale les tâches internes aux groupements fonctionnels.

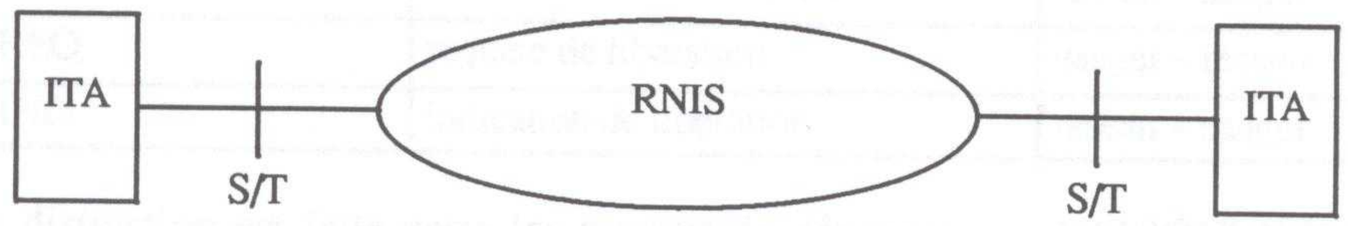


fig. 4: services RNIS et points de référence

La spécification formelle réalisée représente ainsi les interactions entre usagers et réseau au niveau des deux points de références S/T.
Le comportement décrit au niveau de l'interface correspondant à l'installation appelante est caractérisée par la terminologie "appel sortant". Le symétrique au niveau de l'ITA appelée est caractérisée par la terminologie "appel entrant".

Le protocole D

La signalisation mise en oeuvre entre l'usager et le réseau permet l'établissement des connexions nécessaires au transfert des infos entre usagers, et à la réalisation des services supplémentaires.
Pour ce faire, le protocole D est structuré en couches:

- *le niveau physique* (couche1): défini aux interfaces S/T des accès de bases ou à débit primaires.
- *le niveau liaison (couche2)* : assure la transmission sécurisée de trames.
- *le niveau réseau* (couche3) : commande et supervise les appels

Son objectif est d'offrir tous les services définis par le RNIS aux différents équipements d'abonnés.
Utilisé pour la commande des appels, le niveau 3 réalise entre autre:

- *le traitement des messages de niveau 3,*
- *la gestion des ressources associées à l'appel,*
- *la gestion des temporisateurs.*

Deux de ces points sont développés dans les paragraphes suivants.
Les recommandations de l'ITU sur les téléservices et services supplémentaires du RNIS sont basées sur ce seul niveau dans un soucis d'efficacité. La description de ces services n'est pas décomposée en service physique, liaison, réseau et ainsi de suite pour chacune des sept couches. La spécification formelle réalisée se base donc sur ce niveau de la structure OSI.

II.2.2 Primitives de services et temporisareurs

Les primitives de services

Un certain nombre de primitives ont été définies pour symboliser les différentes phases de la communication téléphonique entre usagers au niveau de leur interface S/T [*rec88*]. Le service est dit confirmé. La requête d'appel s'accompagne donc d'une confirmation de la part de l'appelé.
Les primitives principales sont les suivantes :

primitive	signification	sens
SETUP_REQ	requête de service	usager appelant – réseau
SETUP_IND	présentation de la requête	réseau – usager appelé
SETUP_RESP	prise en charge d'un appel	usager appelé – réseau
SETUP_CONF	présentation de la prise en charge à l'appelant	réseau – usager appelant
REPORT_REQ	requête d'alerte	usager appelé – réseau
REPORT_IND	présentation de la requête	réseau – usager appelant
DISCONNECT_REQ	requête de déconnexion	usager – réseau
DISCONNECT_IND	indication de déconnexion	réseau – usager
RELEASE_REQ	requête de libération	usager – réseau
RELEASE_IND	indication de libération	réseau – usager

A noter que distinction est faîte entre les notions de déconnexion (DISCONNECT) et de libération (RELEASE). A l'issu du dialogue par exemple, chacun des interlocuteurs peut établir une requête de déconnexion. Celui–ci est alors dégagé de l'appel mais les ressources réservées (le circuit à travers les canaux) par cet appel ne sont libérées que par le biais de l'événement RELEASE.
D'autre part, un grand nombre de primitives, propres à l'architecture, ont été rajoutées. Elles correspondent le plus souvent à des interactions nécessaires à la validation d'événements par rapport aux ressources gérées(canaux,...). Certaines de ces primitives annexes ne sont pas des interactions, mais des tâches internes à chacune des entités intervenantes pour la gestion de la validité de la requête, du routage, ou encore des temporisateurs par exemple.

Les temporisateurs

Le niveau rattaché à la description du service implique également la gestion de temporisateurs. Les trois temporisateurs utilisés sont les suivants:

- *Temporisateur Timer1*

  Celui–ci est activé dés la présentation de l'appel à l'ITA appelé.
  Il est désactivé dés la réception d'une alerte, d'une déconnexion de part ou d'autre,ou encore d'une indication de prise en charge de l'appel par un terminal de l'ITA appelée.
  S'il est déclenché, la phase de déconnexion est générée ainsi que la libération des ressources allouées au niveau de l'appelant.

- *Temporisateur Timer2*

Il est activé suite à l'éventuelle réception d'une première alerte (après indication, un terminal peut très bien prendre l'appel en ligne sans émettre d'alerte).
Il est désactivé lors d'une requête de prise en charge d'appel ou de déconnexion.
S'il est déclenché, tout comme pour son prédécesseur, déconnexion et libération s'en suivent auto matiquement.

- *Temporisateur Timer3*

Il est activé dans le cas d'une phase de déconnexion avec rétention d'appel après indication de cel le-ci.
S'il est déclenché, la libération de ressource ne s'est pas effectuée au niveau de l'un des ITA durant le temps accordé, et la rétention est dans ce cas automatiquement générée.

II.2.3 Exemple de procédure de communication

Cet exemple (*fig. 5*) présente un cas sommaire de traitement de requête d'appel à travers le séquencement d'événements, au niveau de l'interface usager–réseau, entre le réseau et les installations terminales d'abonnés en jeu. Il n'intègre que des interactions de bases présentées dans le tableau ci–dessus. Les autres événements, interactions ou tâches internes, dépendant de l'architecture choisie, seront présentées par la suite.

L'appel ci–joint peut se décomposer en trois phases:

- *requête et prise en charge d' appel diffusé avec alerte,*
- *phase active de l' appel (dialogue),*
- *déconnexion de la part de l' initiateur de l' appel*

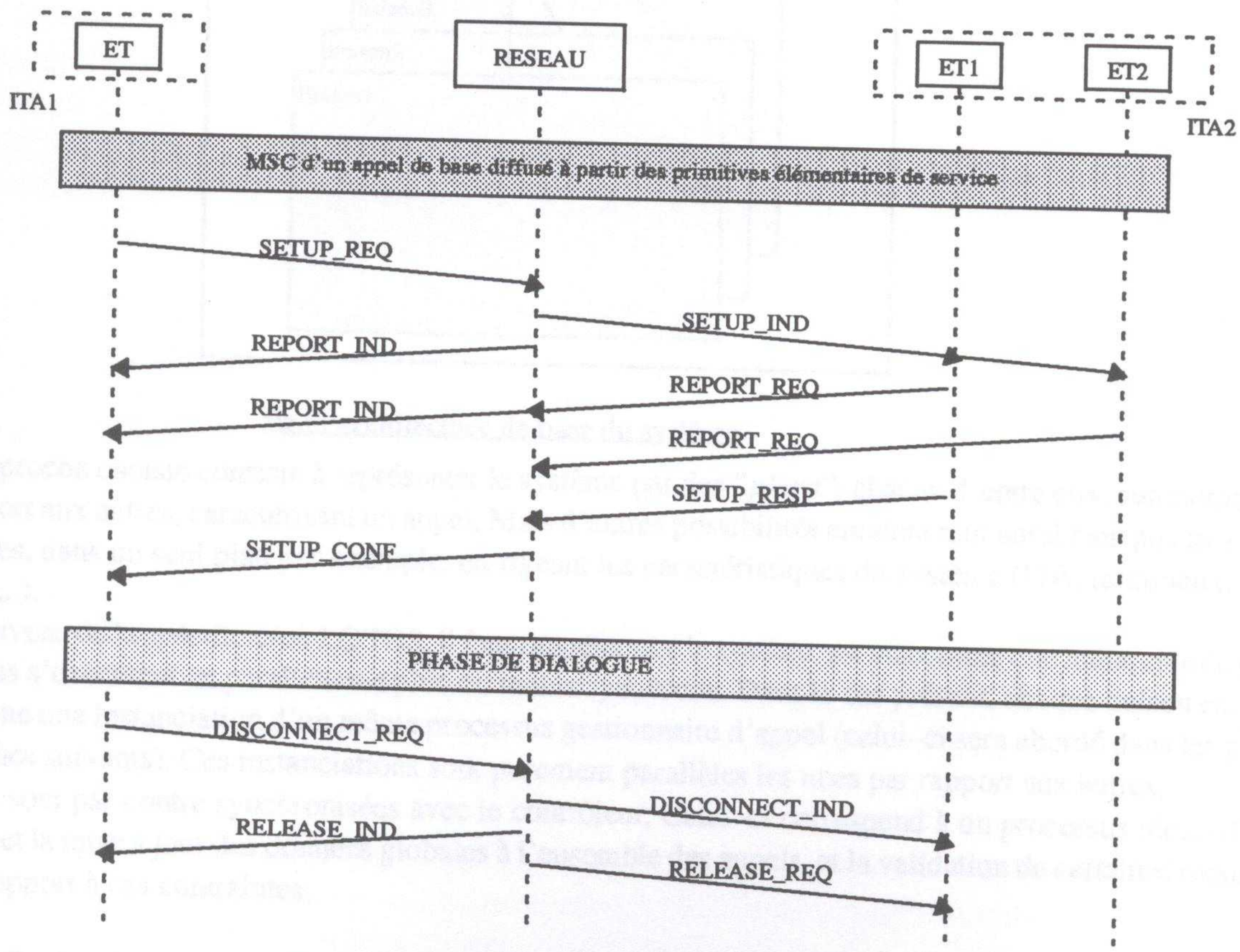


fig.5: procédure d'appel de base et primitives de services

II.3 Architecture orientée "interface usager–réseau"

De part les services qu'il intègre, le système décrit doit permettre le traitement de plusieurs appels simultanés et éventuellement concurrents. Mais cela ne peut se faire sans la gestion des ressources associées aux

appels et la validation contextuelle des requêtes.

II.3.1 Les ressources

Les ressources gérées, au sens matériel, concernent la disponibilité des canaux d'informations au niveau de chaque installation terminale, qui prend part à une liaison téléphonique dans le système. A ces ressources viennent s'en ajouter d'autres, liées également à la validation des requêtes. Les principales données correspondent à la gestion des connexions réalisées et de la disponibilité des terminaux. L'ensemble de ces contraintes sont globales au système et sont regroupées sous la terminologie "ressources" de la spécification formelle.

II.3.2 Structure de base du système

La structure de base du système permet de mettre en oeuvre un ensemble d'appels, la mise à jour des données gérées ainsi que la validation des requêtes émises, par le biais de ces mêmes informations. Par exemple une requête d'établissement d'appel ne sera validée que si l'initiateur n'est pas déjà en communication. Si la requête est valide alors la ressource "disponibilité des terminaux" est mise à jour.
Ces ressources (vis_à_vis de la spécification formelle) sont gérées par un contrôleur global. Le schéma suivant (*fig.6*) présente l'architecture de base du système:

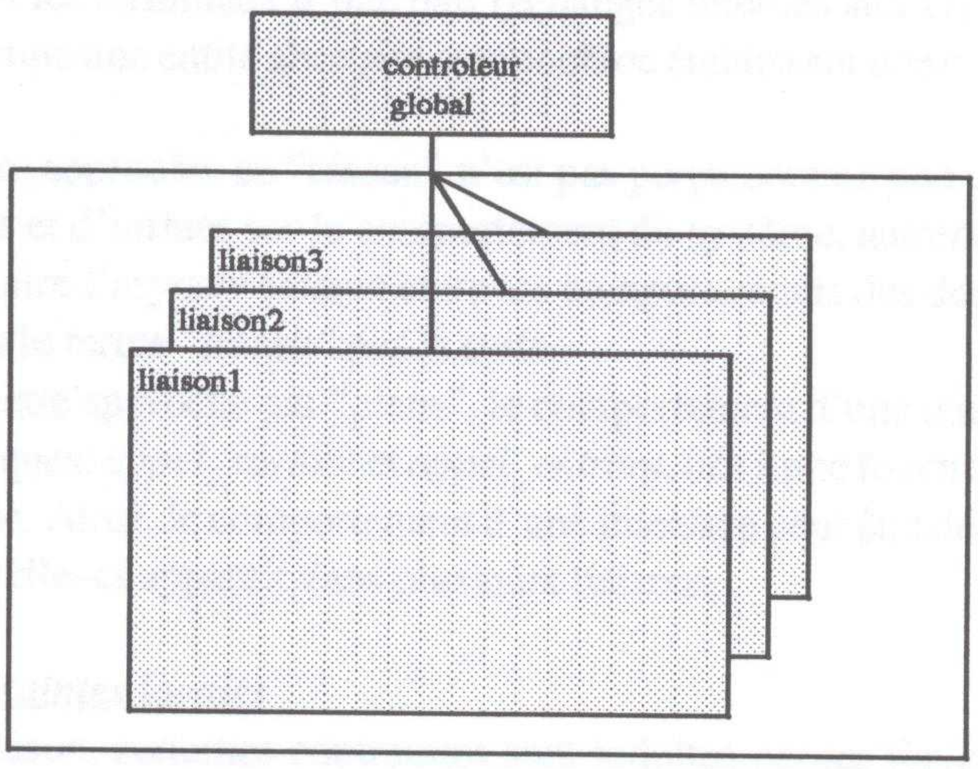


fig.6: architecture de base du système

L'approche choisie consiste à représenter le système par des "*plans*"; chacun d'entre eux, autonome par rapport aux autres, caractérisant un appel. Mais d'autres possibilités auraient tout aussi bien pu être envisagées, dans un seul plan par exemple, en figeant les caractéristiques du système (ITA, terminaux, liaisons,..).
Au niveau de la spécification LOTOS, il faut rappeler que le système est vu comme une hiérarchie de processus s'exécutant en parallélisme plus ou moins synchrone. Dans le cas présent, chaque liaison est vue comme une instanciation d'un même processus gestionnaire d'appel (celui-ci sera abordé dans les paragraphes suivants). Ces instanciations sont purement parallèles les unes par rapport aux autres.
Elles sont par contre synchronisées avec le contrôleur. Celui-ci correspond à un processus récursif qui permet la mise à jour des données globales à l'ensemble des appels, et la validation de certaines requêtes par rapport à ces contraintes.

II.3.3 Architecture liée à un appel

A ce niveau, les possibilités sont nombreuses. Et le choix réalisé est important puisqu'il influt sur les possibilités d'exploitation futures de la spécification (analyse graduelle,...).
De part l'orientation des recommandations, la première approche adoptée peut être caractérisée par la terminologie "*approche orientée interface usager-réseau*".

L'architecture de spécification choisie pour représenter une liaison téléphonique, regroupe ainsi trois composantes principales (*fig.7*) :

- *l'interface usager–réseau au niveau de l'appelant,*
- *l'interface usager–réseau au niveau de l'appelé,*
- *une zone de transit.*

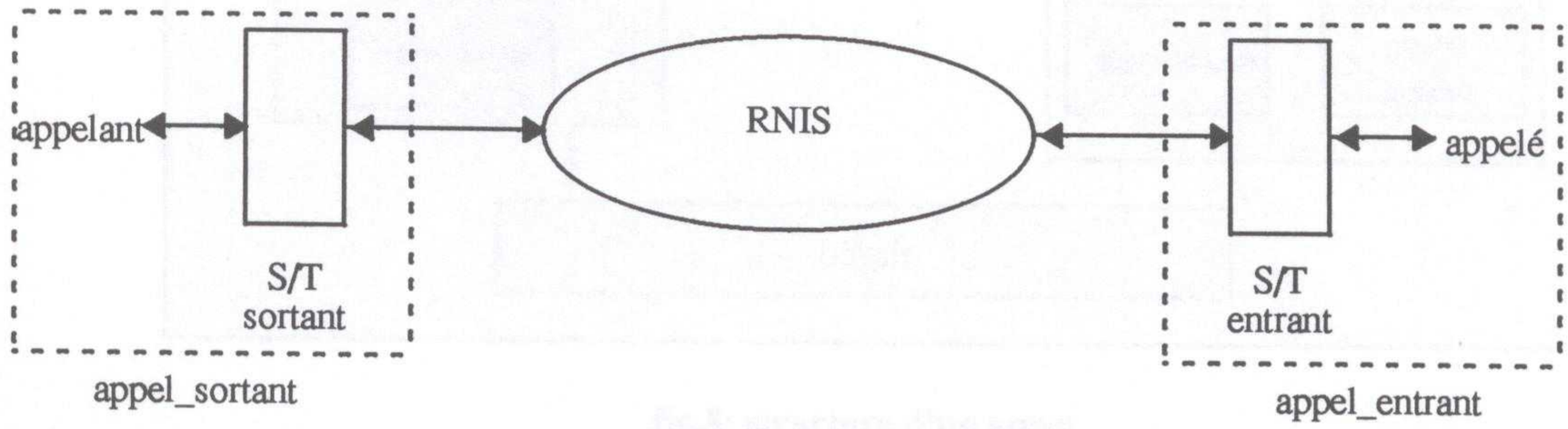


fig.7: entités de communication dans la spécification

De ce fait, la spécification présente le système par le biais de l'ensemble des interactions, entre les interfaces usager–réseau et les terminaux d'une part (échanges internes aux ITA), et le réseau d'autre part. Ce dernier étant vu comme une entité abstraite et monobloc établissant une connexion entre les deux interfaces.
En réalité, dans cette approche, ce "réseau" n'est pas perçu comme une unité "intelligente", capable de gérer des ressources et d'influer sur le comportement du système, autrement dit, de faire du contrôle de flux. Il sert à introduire l'asynchronisme entre les comportements des deux interfaces Pour cette raison, il sera référencé par le terme "transit" par la suite.
D'autre part, dans cette approche par "plans", le comportement d'une interface est décomposée en deux aspects non symétriques: *appel_sortant* et *appel_entrant*, liés à une fonction spécifique: initiative d'appel ou installation ciblée. Ainsi, le comportement d'une interface peut être décomposé à travers divers plans dans la mesure où celle–ci apparait dans plusieurs liaisons.

La gestion des contraintes locales
Au niveau d'une liaison, certaines contraintes sont induites par les règles de dialogue entre entités (un terminal ayant opéré une alerte, ne peut être à l'initiative d'une déconnexion avant qu'il n'ait pris l'appel en charge, par exemple). Les ressources gérées par le contrôleur global ont déjà été présentées ci–dessus. D'autres sont à gérer au niveau local (plan de liaison) afin de valider certains comportements en fonction du contexte.
Dans un soucis de modularité, les données sont rattachées à une entité dédiée à ce traitement au niveau de chacune des interfaces. De même chacun des services est décrit dans une entité différente (dans sa fonction appel entrant), de manière à pouvoir réduire le comportement dans le cas d'une simulation ciblée (par rapport à certains scénarios) mais aussi faciliter les évolutions et les modifications (ajout ou suppression de services par exemple).Dans la configuration actuelle, chaque "entité terminale de plans" est donc décomposée en deux ou trois blocs synchronisés à divers niveaux.

Bien sur, il est supposé que chacune des ITA a souscrit le service de la téléphonie RNIS, et éventuellement celui de l'appel en instance. Dans le schéma suivant (*fig. 8*), l'architecture du système est restreinte à un appel synchronisé avec le contrôleur global.

Les synchronisations
Les interactions entre les différents blocs (processus LOTOS) opérants dans la spécification se font par l'intermédiaire de synchronisations. Ainsi, certaines actions sont synchronisées entre le processus "contrôleur global" et les processus suivants (les portes de synchronisation sont également indiquées):

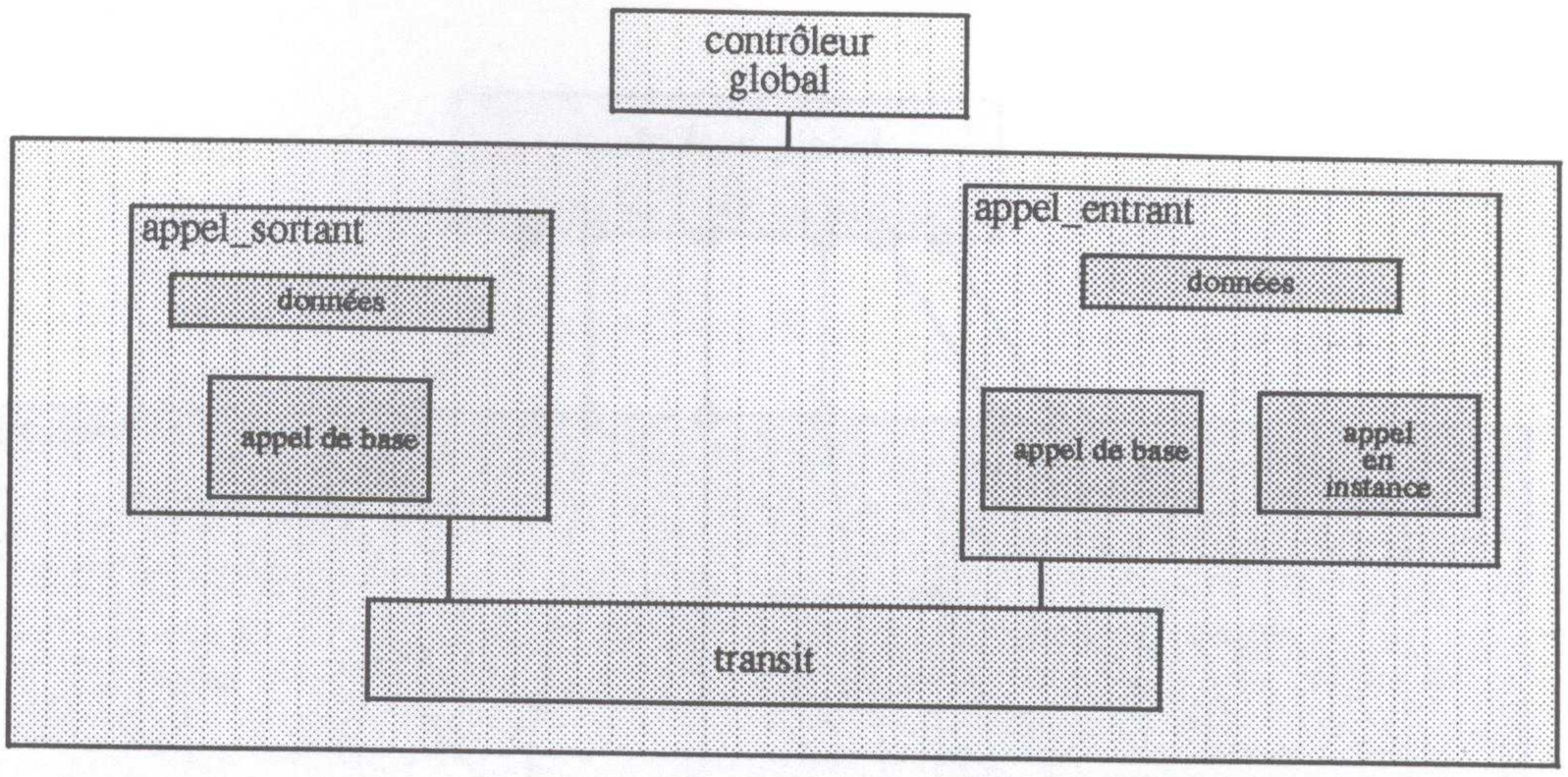


fig.8: structure d'un appel

- appel sortant et transit: *Out_Net*,
- appel entrant et transit: *In_Net*,
- appel sortant, transit et appel entrant: *Dial* (-> Dialogue),
- appel entrant seul: *UserB*, *In_Ctrl*,
- appel sortant seul: *UserA*.

Ces portes sont présentées ici avant même la description du comportement de chacune des "boites", mais en fait leur apparition provient de constats lors de cette phase. La vue générale du système par rapport au réseau de transit et aux interfaces S/T avec les principales synchronisations UserA, UserB, Out_Net et In_Net est fournie par la *figure 9* . La *figure 10* représente la traduction au niveau de chaque plan "liaison" de la spécification LOTOS.

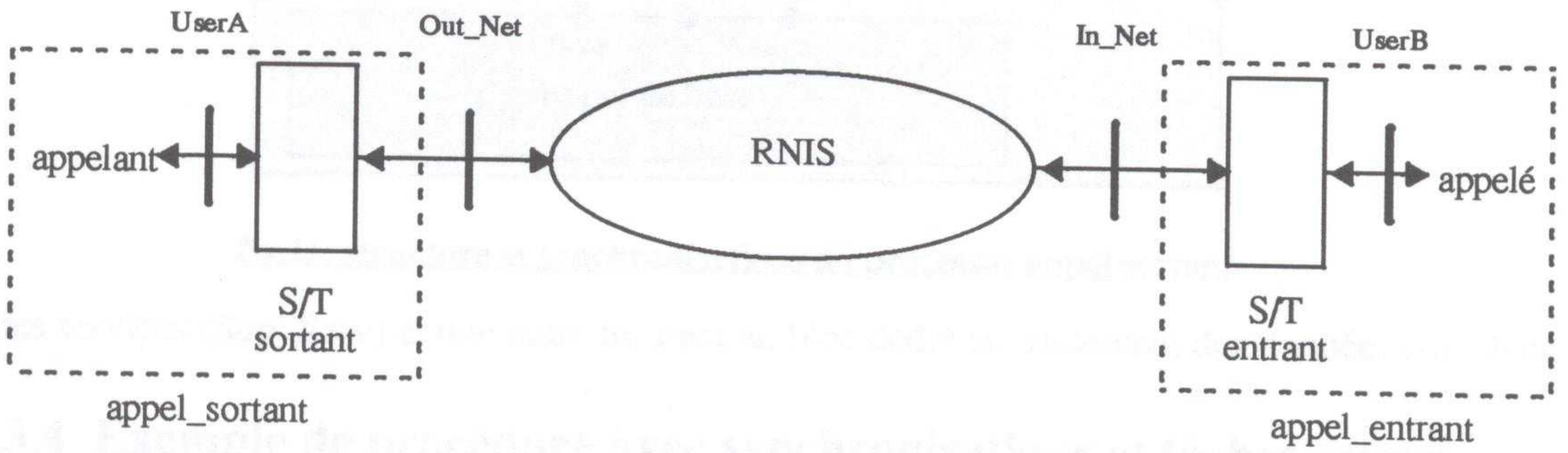


fig.9: synchronisations dans les entités de la spécification

Une fois la connexion établie, les deux entités "*appel_sortant*" et "*appel_entrant*" dialoguent de manière synchronisée sur "Dial". Au niveau de chacune de ces "entités terminales", des synchronisations existent aussi entre les différents processus.

Les synchronisations au niveau des processus appel sortant et appel entrant

Concernant l'interface usager-réseau côté appelant (*fig.11*), des synchronisations lient les processus de gestion de données, du comportement de base (côté appelant) et les portes de synchronisations décrites ci-dessus (UserA,Dial et Out_Net) .

Au niveau de l'interface usager-réseau coté appelé (*fig. 12*), outre des synchronisations entre certaines portes décrites au niveau supérieur (UserB, Dial,In_Ctrl et In_Net) et les gestionnaires de données, d'appel de base et d'appel en instance, une dépendance interne est établie entre chacun des processus descrip-

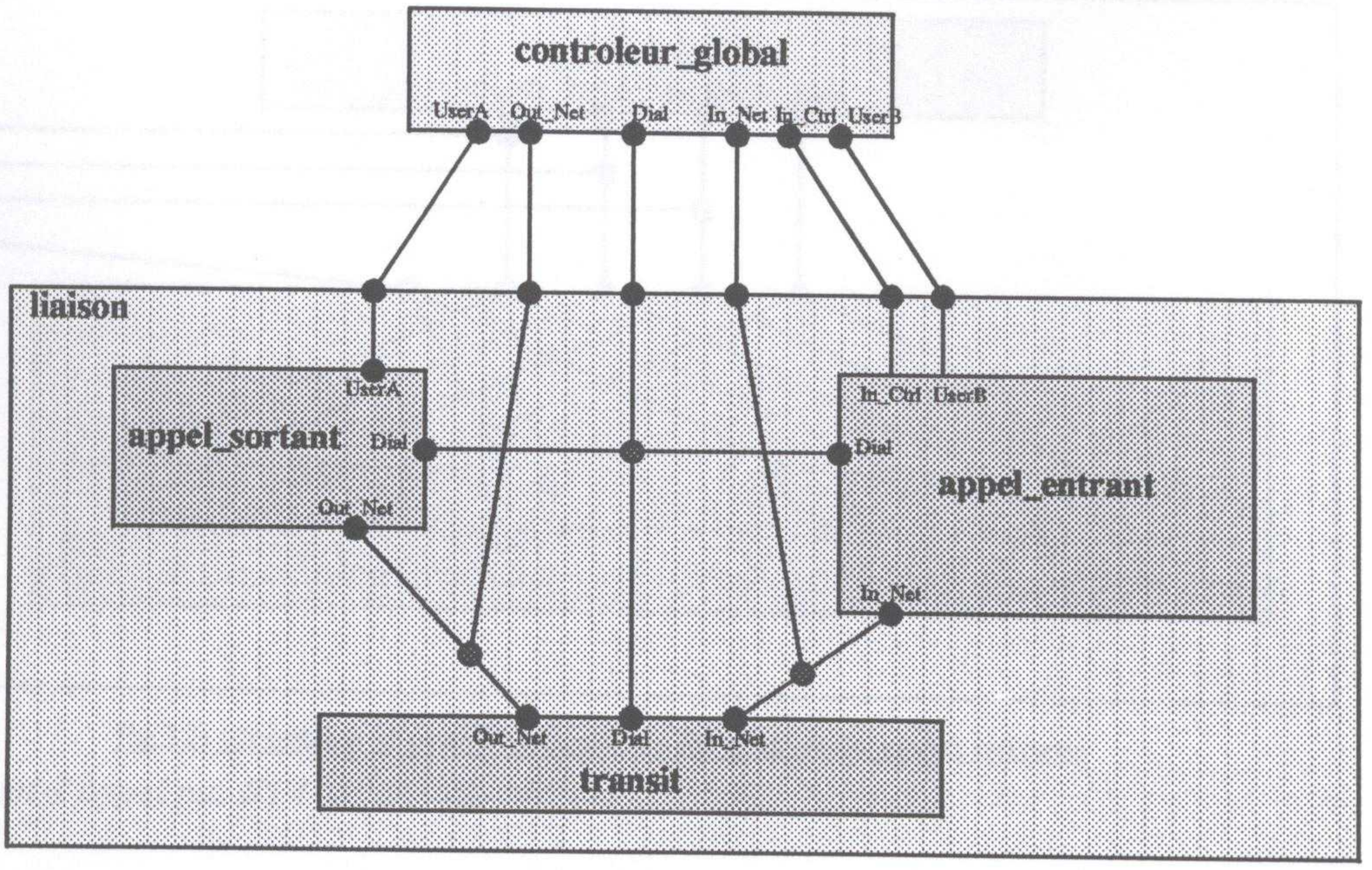


fig.10: structure de contrôle et synchronisation

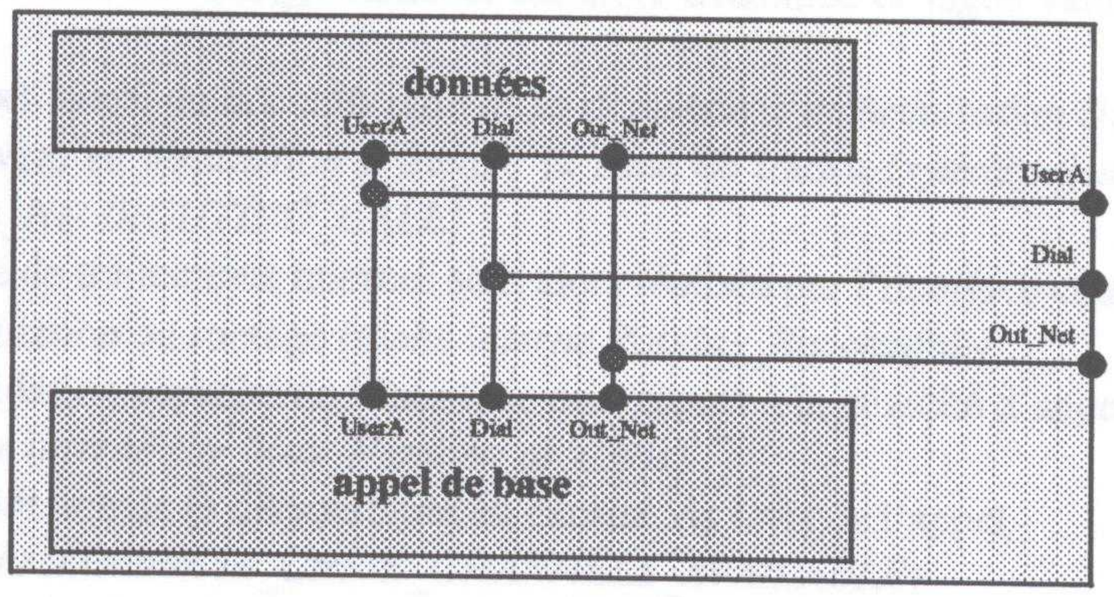


fig.11: structure et synchronisations du processus appel sortant

tifs des services (Sup_Serv) et une autre les lient au bloc dédié au traitement des données (In_Data).

III.3.4 Exemple de procédure avec synchronisations et tâches internes

L'exemple suivant (*fig. 13*) présente un exemple élémentaire de procédure d'appel intégrant les deux services spécifiés sous la forme de MSC. Les communications entre entités y sont représentées par des primitives de services associées aux synchronisations décrites précédemment. Les tâches internes à chacune des ces entités sont également inclues (gestion de temporisateur par exemple) ainsi que les synchronisations internes. Ces entités sont les suivantes :

- *L'ITA à l'initiative de l'appel: ITA_Ori,*
- *le réseau de transit,*
- *l'ITA ciblée: ITA_CiB*

Interprétation du MSC

L'exemple abordé peut se décomposer en plusieurs phases:

- *requête d'appel à l'initiative du terminal ET2 et prise en charge par ET4*
- *requête d'appel émanant du terminal ET1 et mise en instance*
- *libération de communication par ET2 et prise en ligne de l'appel en instance par ET4*

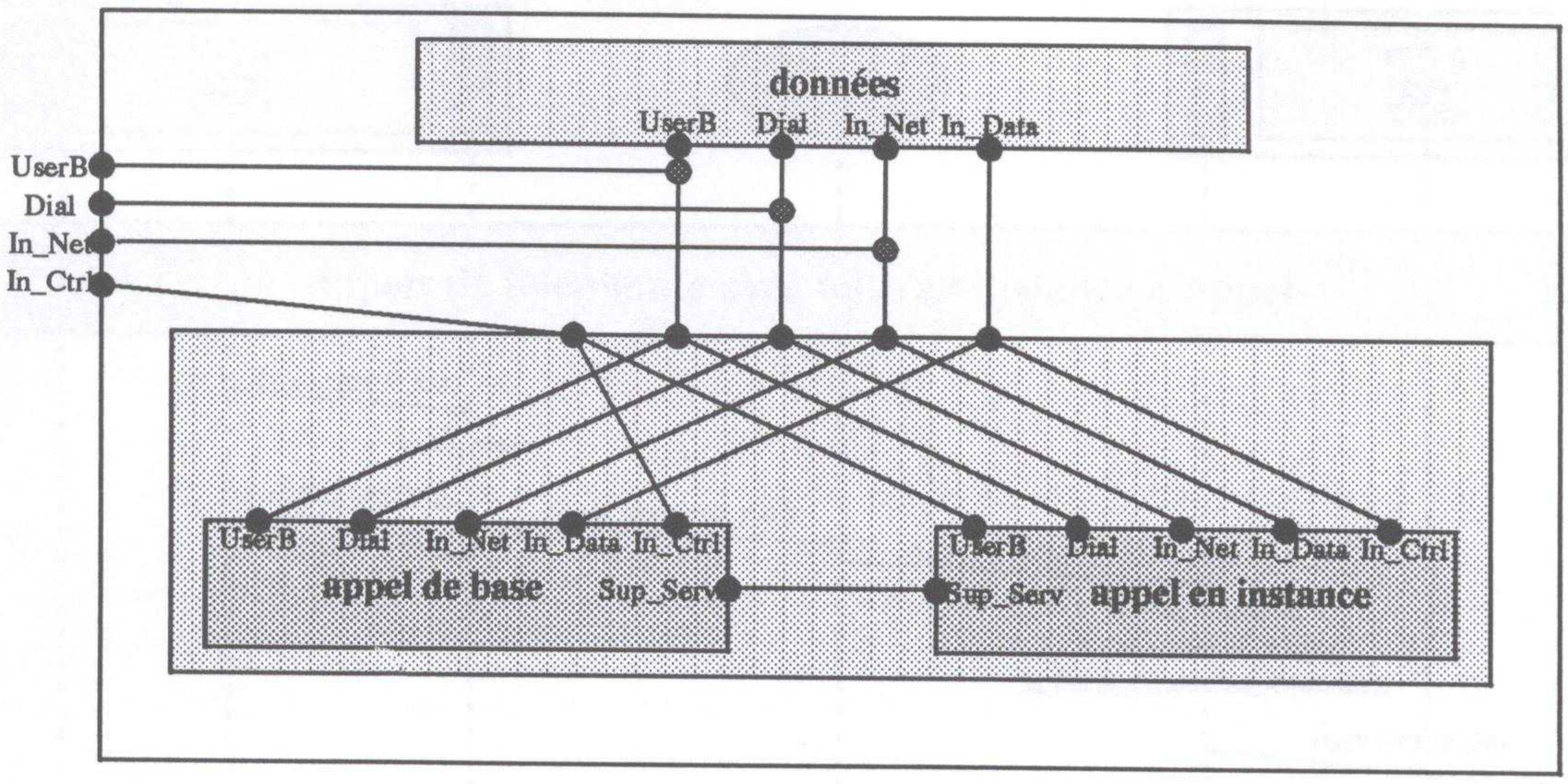


fig.12: structure et synchronisations du processus appel entrant

– libération d' appel par ET4

La première d'entre-elles est détaillée dans les lignes suivantes:
Le terminal ET2 de l'ITA *ITA_Ori* émet une requête d'appel diffusé vers son interface usager–réseau *(synchronisaton UserA !SETUP_REQ)*. Celle–ci est alors examinée et jugée valide *(tâche interne Valide !Request)*.
L'initiateur est alors averti de la conformité de sa demande *(synchronisation UserA !REPORT_IND)* et l'appel est routé vers sa destination à travers la zone de transit *(synchronisations Out_Net !ROUTING_CALL et In_Net !ROUTING_CALL)*.
Un certain nombre de traitements internes sont alors réalisés par l'interface côté appelé:

– test et validation du routage *(tâche interne Routing_Successfull)*
– test et validation capacité de l'interface par rapport à la quantité d'appels
(tâche interne No_Max_Number_of_Calls)
– test et validation de la disponibilité en canaux vis à vis de l'appel
(synchronisation In_Ctrl !B_CHANNELS_AVAILABLE)

A noter que l'ensemble des terminaux faisant figures d'interlocuteurs potentiels se voient notifier l'appel *(synchronisation UserB !SETUP_IND)*.
Le temporisateur Timer1 est activé *(synchronisation In_Data !SET_TIMER1)*, puis le terminal ET4 décide de prendre l'appel en ligne *(synchronisation UserB !SETUP_RESP)*. L'interface établit alors la connexion point à point *(tâche Connect_Circuit)* et désactive le temporisateur *(synchronisation In_Ctrl !Reset_Timer1)*
Le terminal ET2, initiateur de l'appel, est alors averti de la prise en charge *(synchronisations In_Net !SETUP_CONF, Out_Net !SETUP_CONF, UserA !SETUP_CONF)* alors que ET2 se voit indiquer l'établissement de la connexion *(UserB !CONNECTED_IND)*.
L'appel rentre dès lors dans sa phase "active" *(synchronisation Dial !DIalogue)*

Evolutions
L'exemple étudié ne reprend qu'un sous–ensemble des possibilités inhérentes au traitement d'un appel. Les tests ont toujours été validés mais ils peuvent très bien être rejetés et dans ce cas, la phase de libération est activée avec rétention ou non de l'appel (attention: ne pas confondre rétention et mise en instance. L'appel est en rétention lors d'une phase de déconnexion tant que l'interlocuteur n'a pas raccroché).
D'autre part, le premier appel a été pris en charge dés sa présentation. Mais des alertes ou des indications de rejet pourraient très bien être émises en prémisse de la part de terminaux, potentiels interlocuteurs.

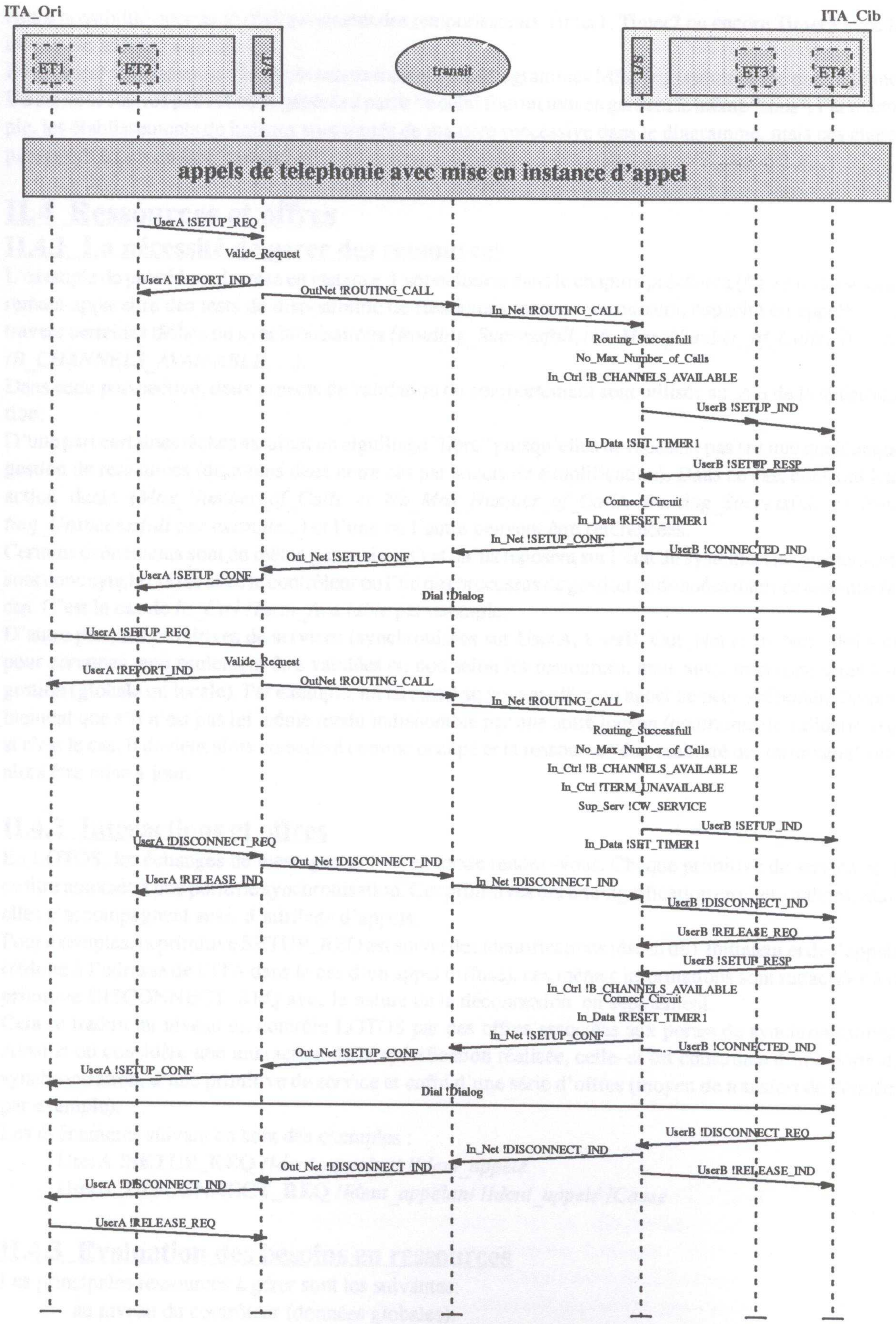


fig.13: exemple de procédure de mise en instance d'appel avec portes de synchronisations et tâches internes

Autre possibilité encore, le déclenchement des temporisateurs Timer1, Timer2 ou encore Timer3 avec le traitement adéquat en aval.
Il faut aussi remarquer que la représentation employée (diagrammes MSC) ne traduit pas le parallélisme. Un tas de scénarios peuvent être générés à partir de celui fournit tout en gardant le même "sens". Par exemple, les établissements de liaisons sont traités de manière successive dans le diagramme, mais ces étapes pourraient très bien être "parallèles".

II.4 Ressources et offres

II.4.1 La nécessité de gérer des ressources

L'exemple de procédure de mise en instance d'appel fourni dans le chapitre précédent (*fig.13*), laisse clairement apparaître des tests de disponibilité de ressources (canaux, terminaux, capacité en appels, ...) à travers certaines tâches ou synchronisations *(Routing_Successfull, No_Max_Number_of_Calls, In_Ctrl !B_CHANNELS_AVAILABLE, ...)*.
Dans cette perspective, deux aspects de validation de comportement sont utilisés au sein de la spécification:
D'une part certaines tâches assurent un aiguillage "libre" puisqu'elles ne reposent pas sur une quelconque gestion de ressources (du moins dans notre cas par soucis de simplification). Dans ce cas, elles ont leur action duale *(Max_Number_of_Calls et No_Max_Number_of_Calls, Routing_Successfull et Routing_Unsuccessfull par exemple...)* et l'une ou l'autre peuvent être référencées.
Certains événements sont du même genre si ce n'est qu'ils reposent sur l'état du système. Ces événements sont donc synchronisés avec le contrôleur ou l'un des processus de gestion de données internes aux interfaces. C'est le cas de *In_Ctrl !Term_Available* par exemple.
D'autre part, les primitives de services (synchronisées sur UserA, UserB, Out_Net et In_Net) doivent, pour certaines, non seulement être validées ou non selon les ressources, mais aussi intervenir dans leur gestion (globale ou locale). Par exemple, un terminal se voyant offrir un appel ne peut y répondre favorablement que s'il n'est pas lui même rendu indisponible par une autre liaison (contrainte de validation) et si c'est le cas, il devient alors considéré comme occupé et la ressource "disponibilité des terminaux" doit alors être mise à jour.

II.4.2 Interactions et offres

En LOTOS, les échanges de messages se font en mode rendez-vous. Chaque primitive de service est à ce titre associée à une porte de synchronisation. Ces primitives ont une signification en elles-mêmes, mais elles s'accompagnent aussi d'attributs d'appels.
Pour exemples, la primitive SETUP_REQ est suivie des identifications réseau de l'initiateur et de l'appelé (réduite à l'adresse de l'ITA dans le cas d'un appel diffusé); ces mêmes informations sont rattachées à la primitive DISCONNECT_REQ avec la nature de la déconnexion en supplément.
Cela se traduit au niveau du contrôle LOTOS par des offres associées aux portes de synchronisations. Ainsi si on considère une interaction de la spécification réalisée, celle-ci est constituée d'une porte de synchronisation, d'une primitive de service et enfin d'une série d'offres (moyen de transfert de données par exemple).
Les événements suivant en sont des exemples :

UserA **!SETUP_REQ** *!Ident_appelant !Ident_appelé*
UserB **!DISCONNECT_REQ** *!Ident_appelant !Ident_appelé !Cause*

II.4.3 Evaluation des besoins en ressources

Les principales ressources à gérer sont les suivantes:
- au niveau du contrôleur (données globales):
  - *La disponibilité des terminaux,*
  - *La disponibilité en canal d'information des interfaces,*

– *Les connexions établies au niveau de chacun des terminaux.*

– au niveau du gestionnaire de données de l'interface S/T côté appelé (données locales):

– *La liste des terminaux ayant opéré une alerte,*

– *La liste des terminaux ayant opéré un rejet,*

– *Les états des temporisateurs,*

– *Les caractéristiques de la liaison* (identification de l'appelant et de l'appelé).

– au niveau du gestionnaire de données de l'interface S/T côté appelant:

– *les caractéristiques de la liaison* (identification de l'appelant et de l'appelé).

II.4.4 Description des types LOTOS nécessaires

Une fois établies les nécessités en gestion de données, les types LOTOS résultants ont été conçus.
Les entités appelantes sont caractérisées par deux composantes: leur *adresse réseau* et leur *sous-adresse*:

– La première est représentée par les occurrences d'un type "Terminal":

Diffusion, ET_Ori, ET_Cib, ET_1, ET_2, ET_3 ,....

– La seconde par celles issues d'un type "ITA" :

ITA_Ori, ITA_Cib, ITA_1, ITA_2,...

Ainsi une requête d'établissement d'appel devient :

– pour un appel ciblé: UserA !SETUP_REQ *!ITA_Ori !ET_Ori !ITA_Cib !Term_Cib*

– pour un appel diffusé: UserA !SETUP_REQ *!ITA_Ori !ET_Ori !ITA_Cib !Diffusion*

Certaines primitives se voient associer, outre ces caractéristiques, la cause d'une déconnexion ou encore la nature d'une alerte. Des types ont donc été crées en fonction des besoins avec des énumérations d'attributs. Par ce biais, une requête de déconnexion peut être UserB !DISCONNECT_REQ...*!User_Determines_User_Busy* et une requête d'alerte Out_Net *!REPORT_IND.. !Progress*

D'autre part, de nombreuses ressources nécessitent l'utilisation de listes. En ce qui concerne la disponibilité des terminaux, celle-ci est effectuée au moyen d'une liste qui intègre l'ensemble des identifications des terminaux en communication. Par soucis d'efficacité, un terminal est alors caractérisé par un couple: *Id_Term(ITA_Ori,ET_Ori)*, par exemple, ce qui facilite les opérations.
Pour la gestion des capacités d'accès des interfaces, le même principe a été retenu avec une liste de couples; chacun de ces couples caractérisant l'identification de l'interface et sa disponibilité en canaux d'information. L'hypothèse que chaque ITA ne dispose que d'une interface d'accès ayant été émise, une interface est donc référencée par cette une installation (ITA_Ori,...).
Les tests sur la capacité des interfaces se font ainsi par rapport au contenu de la liste et des événements In_Ctrl !B_CHANNELS_AVAILABLE *!ITA_Cib* et In_Ctrl !B_CHANNELS_UNAVAILABLE *!ITA_Cib*. Si l'ITA précisée n'apparait pas dans la liste ou si sa capacité est supérieure à 0, alors le premier cas est validé, au détriment du second. Et vice-versa si la capacité est nulle.

La gestion des terminaux ayant opérés une alerte, un rejet, ou connectés se fait de la même manière en traitant une liste d'identification de terminaux. Enfin ,outre ces types spécifiques, des types de bases (Booléen, Entier Naturel,...) ont également été utilisés

II.5 Description et vérification graduelle de la spécification

L'architecture de processus a fait l'objet d'une description dans les paragraphes précédents. La seconde phase de l'élaboration de la spécification, après les types abstraits, concerne la partie contrôle LOTOS. La description de l'architecture en fait partie mais nous nous intéressons ici au comportement de chacune des "boîtes" les plus internes: contrôleur global, réseau de transit, et pour chacune des interfaces, gestionnaires de données, d'appel de base et de service supplémentaire.

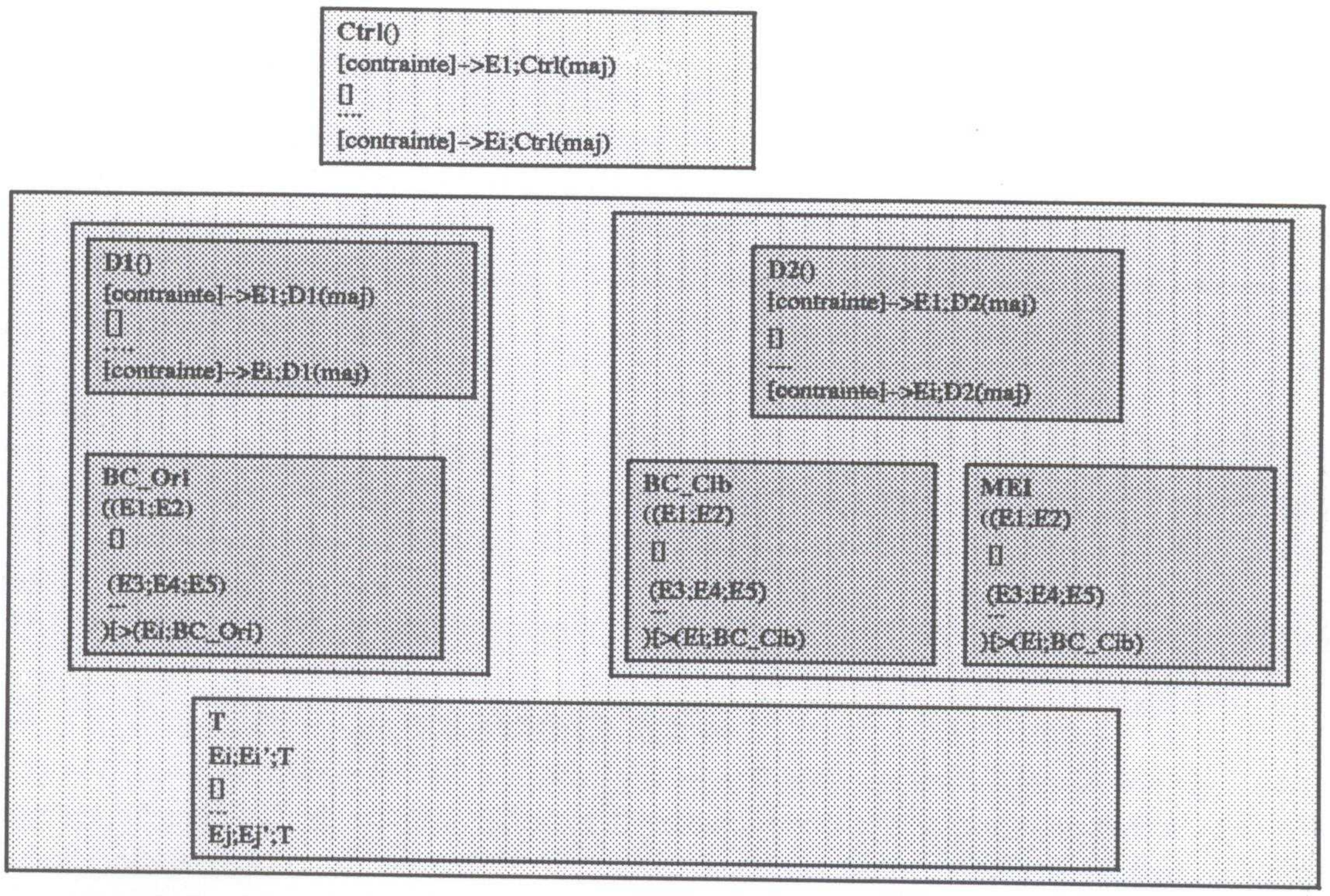
II.5.1 La description des comportements élémentaires

Trois styles de spécification sont répertoriés selon les fonctions attribuées:

- *un disjonction d' événements* avec contraintes et mise à jour de données pour les gestionnaires de données et le contrôleur global.
- *un contrôle de flux* pour le réseau de transit (dans ce cas pour gérer l'asynchronisme)
- *un séquencement temporel d' événements* (primitives de services, synchronisation secondaires et tâches pour les gestionnaires des services d'appel de base et d'appel en instance).

Le schéma (*fig.14*) représente ces trois aspects différents. Les comportements décrits ici ont aucun autre intérêt (et nulle signification). Les processus *Ctrl(), D1(), et D2()* correspondent au *contrôleur*, et *aux gestionnaires de données*. Les processus *BC_Ori* et *BC_Cib* décrivent le comportement inhérent au *téléservice de la téléphonie RNIS* alors que *MEI* est rattaché au *service supplémentaire intégré*. Enfin, le processus *T* gère l'*asynchronisme*.
C'est dans la perspective d'associer un sens à chacun de ces processus, que les diagrammes SDL issus des recommandations de l'ITU sont utilisés. Ils traduisent l'aspect sémantique de chacun des services. Les combinaisons d'actions potentielles à chaque entité sont tirées de l'analyse de ces documents et chacune des actions se voit associer, en fonction de sa nature et de ses contraintes, une synchronisation.

.



Ctrl(), D1(), BC_Ori, D2(), BC_Cib, MEI et T : processus récurssifs
E1,E2,D3,E4,E5,Ei,Ei',Ej': évènements élémentaires

maj:mise à jour des ressources
[], ; , [> opérateurs LOTOS

fig.14: schématisation des styles de contrôle abordés

II.5.2 Approche graduelle

La première partie de ce rapport met en évidence l'intérêt (et même la nécessité) d'une vérification de protocole dés les premières phases du processus de développement.

Ainsi les étapes dans l'intégration de fonctionnalités au système furent les suivantes:

- *appel de base*

- *ciblé puis diffusé*
- *sans puis avec contrôleur global*

– *appels simultanés non concurrents*
– *appels simultanés et concurrents avec mise en instance*

La complexité de la spécification a ainsi évolué au rythme des services. Cette approche a permis de spécifier le système par étapes et de le simuler avant chaque évolution (la même démarche a été entreprise pour les types abstraits)
Chacune des entités terminales : *appel sortant* et *appel entrant*, avec processus gestionnaires de données et de services, fût abordée dans un premier temps, dans le cadre de l'appel de base. Puis vînt le réseau de transit et enfin le contrôleur global. Le même schéma fut ensuite repris en intégrant les modifications inhérentes au service d'appel en instance.
Ainsi, par rapport à la figure précédente, les processus D1() et BC_Ori puis D2(), BC_Cib et MEI furent élaborés puis simulés. La description fut ensuite enrichie avec T et Ctrl().

La spécification ainsi réalisée est fournie en annexe B.

III . Analyse de la spécification

III.1 Définition

Une spécification peut être réalisée dans deux perspectives: *support formel à l'étude du système décrit*, ou encore *support d'implémentation*. Dans les deux cas, le terme "support" implique que la description doit fournir le plus de garanties possibles quant à sa conformité aux fonctionnalités réelles du système. Elle fait donc l'objet d'une analyse à l'aide des outils supports à la technique formelle utilisée.
Bien qu'abordée dans ce rapport à la suite de la partie sur la conception de la spécification LOTOS, il ne faut pas y voir une découpe en deux phases successives: description *complète* puis *analyse*. Cette dernière fait au contraire partie intégrante de l'élaboration et elle est réalisée en parallèle à la description. Une spécification ne peut être considérée terminée que lorsqu'elle est validée.

Deux étapes sont à distinguer:

- *la validation de la spécification en regard aux recommandations de références,*
- *la validation du système décrit par rapport aux fonctionnalités définies.*

Validation de la spécification
Cette première étape est importante (impérative en théorie). La réalisation de la spécification LOTOS sur les services de téléphonie RNIS et d'appel en instance repose sur les recommandations de l'ITU. Cette description est sensée apporter les vertus du formalisme par rapport à ces supports de références. Néanmoins, cela ne saurait être significatif si le comportement du système qu'elle décrit en est différent. La spécification est donc simulée et comparée aux diagrammes SDL issus des *séries I.220* et I.253.1, qui caractérisent l'aspect dynamique.

Validation du système
Cette seconde perspective est réalisée suite à la précédente, et correspond à une analyse plus fine. La spécification est alors supposée conforme aux recommandations, et elle peut dès lors servir elle même de support, formel cette fois, pour étudier en détail les services. Il s'agit donc ici de valider non plus la spécification mais le système qu'elle décrit.
Les outils supports à LOTOS sont utilisés dans ces objectifs, en fonction des caractéristiques recherchées.

III.2 Protocoles et automates à états

Le concept d'automate à états finis est très présent dans le monde des techniques de description formelle [*Aut89*]. En effet les automates, ou graphes, permettent de modéliser le comportement d'un système de processus asynchrones, et de développer des outils de vérification. D'ailleurs de part sa définition, LOTOS, langage de spécification d'ordonnancements temporels, exprime sa sémantique dynamique en fonction d'un graphe (ou automate).
L'exemple suivant (*fig.15*) représente sommairement le service de téléphonie au niveau de l'interface usager–réseau côté (côté appelant: appel sortant, avec restrictions sur déconnexions). Les transitions correspondent à des primitives de services synchronisées. Le graphe résultant montre les interactions de l'interface vers l'usager (*synchronisation UserA*), le réseau (*synchronisation Out_Net*), directes avec l'autre interface (synchronisation Dial), ainsi que certaines tâches internes (*Valide_Request* et *No_Valide_Request*).

III.2.1 Vers le modèle graphe: approche statique

Dans le cadre des "vérificateurs", pour lesquelles l'exploitation de la spécification est basée sur l'utilisa-

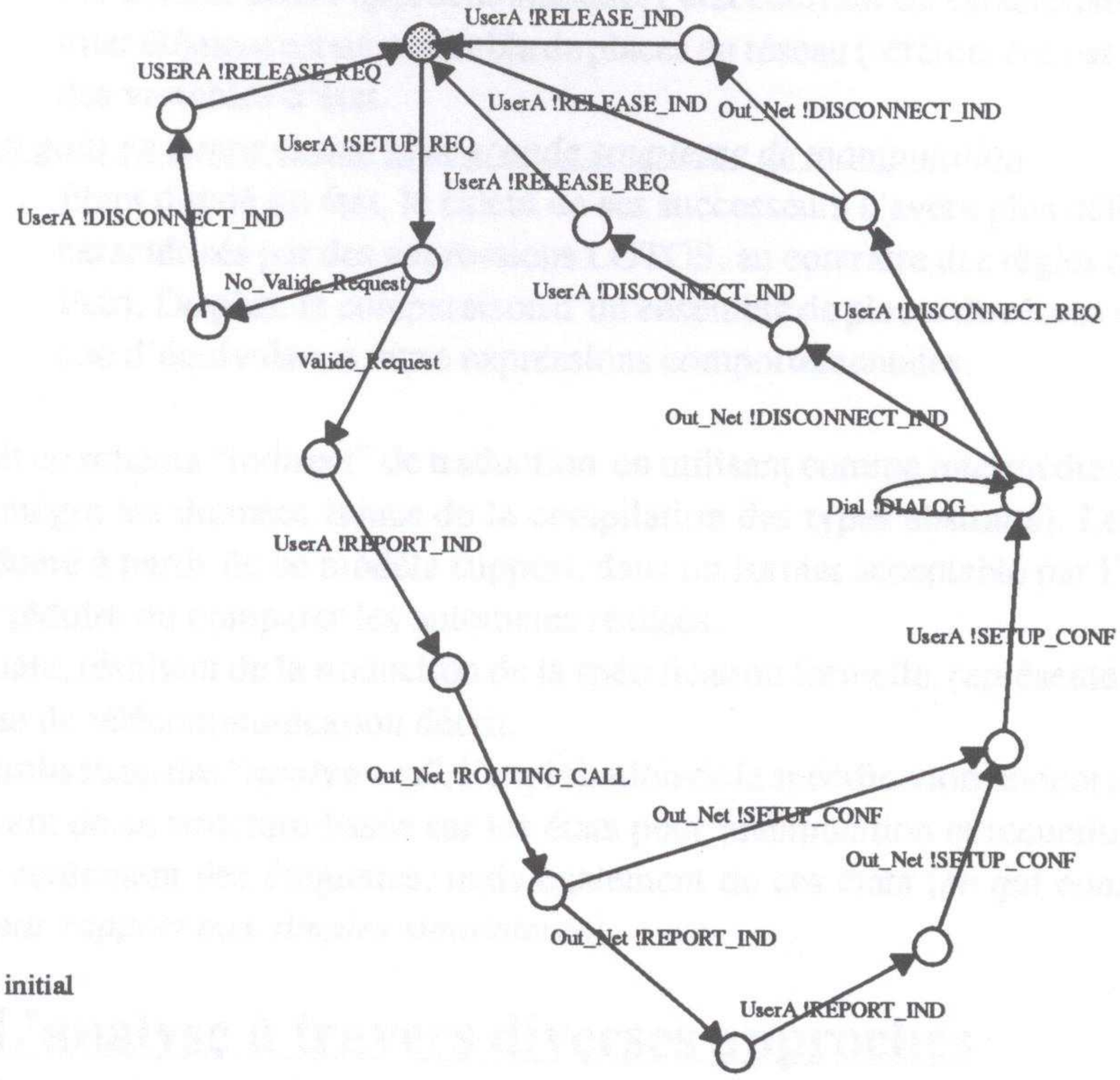


fig.15: modélisation du comportement simplifié d'une interface: appel sortant

tion d'un modèle mathématique, deux schémas de traduction vers le modèle sont répertoriés: l'approche *dynamique* et l'*approche statique* [*Gar89*].
La première citée consiste à traduire directement la spécification LOTOS en un graphe. La seconde décompose cette traduction en deux étapes: *la génération* et *la simulation* où une forme intermédiaire, le réseau, joue le rôle d'interface (*fig.16*).

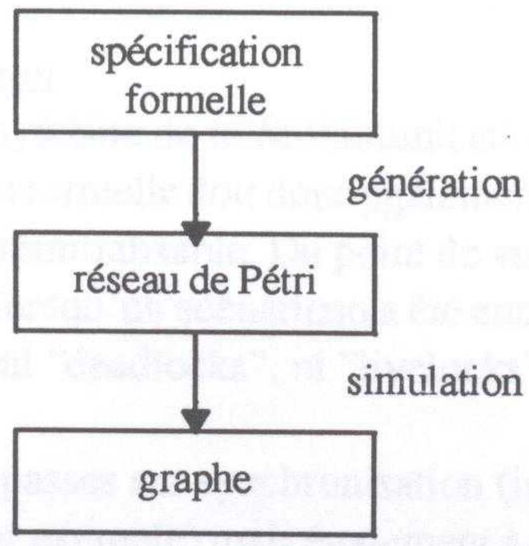


fig.16: approche statique de la modélisation

Ce modèle intermédiaire offre une description complètement statique de la structure de contrôle, sans création dynamique de processus. De plus ils sont bien adaptés aux algèbres de processus, et sont suffisamment expressif pour décrire un système asynchrone dont le contrôle est statique.

La référence à ce modèle intermédiaire offre trois avantages principaux:

- *un gain en mémoire*

  La taille des états d'un réseau est beaucoup plus compacte.

Dans l'approche dynamique, chaque état est une expression de comportementale LOTOS. Par contre, dans l'approche statique, l'état courrant est caractérisée par un couple dont le premier élément est un ensemble de places du réseau (référencées par des nombres), et le second des variables d'état.

- *un gain en temps et une plus grande souplesse de manipulation*

  Etant donné un état, le calcul de ses successeurs s'avère plus délicat lorsque ceux-ci sont caractérisés par des expressions LOTOS, au contraire des règles d'évolution des réseaux de Pétri. De plus, la comparaison d'un ensemble de places de réseau est plus aisée que la recherche d'équivalence entre expressions comportementales.

Caesar suit ce schéma "indirect" de traduction en utilisant comme intermédiaire un réseau de Pétri interprété (il intégre les données issues de la compilation des types abstraits). Le graphe de simulation est ensuite généré à partir de ce modèle support, dans un format acceptable par l'outil aldebaran, utilisé en aval pour réduire ou comparer les automates réalisés.
Cet automate, résultant de la traduction de la spécification formelle, représente la sémantique dynamique du système de télécommunication décrit.
De part l'utilisation des "*analyseurs*", l'exploitation de la spécification revient alors à parcourir le graphe, en se servant de sa structure basée sur les états pour manipulation et recueillir l'information pertinente issue non seulement des étiquettes, mais également de ces états (*ce qui constitue la richesse de cette méthode par rapport aux simples simulateurs*).

III.3 L'analyse à travers diverses approches

La spécification doit faire l'objet d'une validation la plus complète possible. Cela consiste à rechercher des propriétés par le biais des scénarios issus de la simulation.

III.3.1 Le concept de propriétés

L'ensemble des propriétés peuvent être répertoriées en deux groupes, selon leur portabilité: *les propriétés génériques*, et *les propriétés spécifiques* à une description [*Groz89*].

Les propriétés génériques

Elles sont généralisables à l'ensemble des descriptions formelles.
Les principales sont :

- *la détection de la réinitialisation*

  Le fonctionnement d'un système de télécommunication est en théorie infini dans un cadre "normal". Sa spécification formelle doit donc également offrir un comportement dynamique continu et par ce fait être réinitialisable. Du point de vue utilisateur, cela implique un retour aux événements initiaux lorsqu'un scénarioso a été entièrement abordé, soit encore une spécification ne comportant ni "deadlocks", ni "livelocks".

- *la détection de "deadlocks"*

  Cela correspond à des impasses sur synchronisation (inhérentes à une non spécification de réception d'événement par exemple) mais également à des terminaisons "normales" (action stop). La première citée doit être considérée comme une erreur de conception alors que la seconde, même si elle va à l'encontre de la propriété de réinitialisation, peut être vue comme une simplification vis à vis d'une approche exhaustive des scénarios potentiels.

- *la détection de "livelocks"*

  Il s'agit de cycles absorbants. Leur présence au sein du système implique que celui-ci peut boucler sur une séquence réduite, ce qui ne saurait être concrètement le cas. Dans l'exemple de la spécification réalisée sur la téléphonie, la phase active d'un appel est assimilable au séquencement d'une même action basée sur la primitive DIALOG , ce qui se traduit au niveau du modèle graphe correspondant par un cycle élémentaire avec même état de départ

et d'arrivée. Ce cycle ne saurait être absorbant mais interruptible par une requête de déconnexion émanant indifféremment des entités appelante ou appelée.
Bien entendu, la réinitialisation fait de la spécification un système qui ne satisfait pas, à priori, ce critère; mais concepts de réinitialisation et de cycles absorbants sont distingués.

Les propriétés spécifiques

Elles sont propres à chacune des spécifications et ne s'intéressent plus seulement à la forme des ordonnancements mais au fond, c'est à dire les événements proprement dits. L'objectif est de mettre en évidence l'existence de séquences bien précises d'actions (interactions et tâches) dans la description.

L'une des approches peut consister à établir si un certain événement peut se produire une fois un autre réalisé (forme de logique temporelle). Dans le cadre des services spécifiés par exemple, si une indication de la présence d'un appel en instance (événement *UserB !SETUP_IND !Ident_Ori !Ident_Cib !No_Available_Info_Channels*) peut être suivie, via d'autres actions, d'une requête de prise en charge d'appel (*UserB !SETUP_RESP !Ident_Ori !Ident_Cib*).

Une extension peut consister à rechercher un ensemble de scénarios aux extrémités bien définies, et par là répondre à une question qui pourrait être par exemple:

quelles sont les procédures potentielles de déconnexion un fois un appel pris en ligne ?

Cela revient alors à présenter, sous certaines contraintes, les scénarios menant d'un événement caractérisé par la primitive de service SETUP_RESP à une autre basée sur RELEASE_IND ou RELEASE_REQ (libération de ressources).

III.3.2 Intérêts de la modélisation graphique

De part la vision complète qu'il offre du comportement dynamique, et sa structure aisément manipulable, le modèle graphe s'avère être approprié à la recherche des propriétés précédemment énoncées.
En effet, il peut être assimilé à un squelette, support de déroulement de la spécification, fondé sur des transitions. Ainsi, ceci rapporté aux propriétés génériques, une impasse sur synchronisation ou une terminaison "normale" correspondent à un état qui ne possède pas de successeurs. De même il apparait que la caractérisation d'un cycle absorbant est simplifiée par un traitement sur les états que sur les événements directement (*fig.17*).

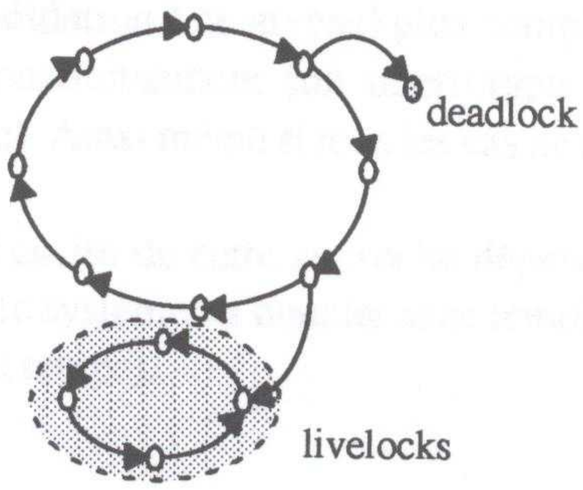


fig.17: caractérisation de "deadlocks" et de "livelocks" dans un graphe de simulation

III.3.3 Les méthodes d'analyse

L'exploration de la spécification peut être réalisée à travers plusieurs méthodes:

- *la simulation interactive*
- *la simulation aléatoire*
- *l'approche exhaustive*

III.3.3.1 La simulation interactive

Dans ce cas, l'orientation de la simulation est laissée à l'initiative de l'utilisateur. L'arbre de simulation (et non le graphe) se construit ainsi petit à petit en fonction des événements sélectionnés. C'est le plus

basique des trois aspects énoncés.
Cette approche a surtout été utilisée dans les premières phases de développement pour une *"vérification de premier niveau"* (analyses syntaxique, sémantique et de scénarios élémentaires) où les contraintes posées par l'explosion combinatoire des scénarios ne se posent pas. Dans le cas d'un appel de base par exemple, cette forme de simulation s'avère être tout à fait appropriée à la vérification de la conformité du comportement indépendant de chacune des entités terminales "plans" (appel sortant, appel entrant). Puis les tests intègrent la zone de transit et le contrôleur global .

Par contre dès que les configurations sont plus conséquentes, les contraintes de temps et de quantité font que cette simulation n'est plus exploitable. En effet, les scénarios se multiplient. L'explosion combinatoire, induite par l'asynchronisme, fait que de quelques scénarios à envisager, la quantité s'est très fortement accrue.
Pour citer quelques chiffres, si un simple appel de base se limite à quelques dizaines de scénarios possibles (une cinquantaine dans la version réellement exploitée mais variable selon la quantité d'attributs de primtives définis), la mise en concurrence de deux appels simultanés aboutit à plusieurs milliers de combinaison. Inutile de préciser que l'approche interactive perd de sa valeur dans ce contexte. Celle–ci se limite ainsi à une certaine forme de "débogage" de la spécification.

III..3.3.2 La simulation aléatoire

Cette approche consiste à sélectionner automatiquement et de manière aléatoire, au cours du temps, un événement parmi ceux réalisables. l'utilisateur n'a alors plus d'emprise sur la simulation (si ce n'est de préciser la racine).
L'inconvénient majeure repose ici sur la mauvaise répartition probabiliste des branches de développement. En effet, un événement a d'autant plus de chance d'être réalisé qu'il se situe prés de la racine, ce qui correspond dans le cas de la spécification réalisée aux tests erronés de validation de requête ou de routage et qui aboutissent à une déconnexion système. Par contre, les scénarios plus intéressants qui aboutissent à une phase active de dialogue ont une probabilité plus faible de "visite".

Néanmoins, dans le cas d'une spécification réinitialisable, cet handicap est comblé et la quantité de branches explorées est plus importante. Le caractère automatique trouve alors tout son intérêt et le choix de cette méthode peut alors se justifier en aval à la simulation interactive, afin de parvenir à une vérification (on ne peut tout de même pas parler de validation à ce niveau) plus complète. En effet, une fois lancé, le système "tourne" jusqu'à ce que son fonctionnement soit interrompu par une erreur de description (impasse sur synchronisation par exemple). Ainsi même si tous les cas ne sauraient être étudiés, un grand nombre le sera.
Une contrainte s'impose toutefois. L'efficacité de cette approche dépend de sa capacité à détecter les cycles absorbants. Dans le cas contraire, le système va boucler sans témoignage apparent d'erreurs pour l'utilisateur (du moins pendant un certain temps).

III.3.3.3 L'approche exhaustive

Celle–ci offre un caractère plus complet que les précédentes, ce qui en fait pratiquement un passage obligatoire. En effet, les simulations interactives et aléatoires ne sauraient permettre d'aborder un ensemble aussi important de scénarios que dans ce cas, or de cette complétude dépend la qualité de la vérification de la description formelle. On ne parle plus alors de vérification mais de validation. D'autre part, celle–ci fait référence au concept de modèle graphe, qui représente la meilleure, voire la seule, voie d'exploration complète de la sémantique dynamique du système.
En théorie, l'analyse consiste en l'exploration de la spécification jusqu'à ce que tous les scénarios aient été envisagés. En pratique, cela ne peut être le cas, du fait d'un contrôle dynamique potentiellement infini, et cela ne se fait que sous certaines contraintes, inhérentes à l'existence de scénarios cycliques (induit par la réinitialisation par exemple).

III.3.4 l'exploitation de la spécification à l'aide de caesar/aldebaran

Ces méthodes d'exploitation sont accessibles par le biais des outils supports au langage de description employé: LOTOS. *Hippo*[*hip87*] et *Smile*[*Smi93*] sont confinés dans la simulation interactive, ce qui ne suscite pas de commentaires particuliers à ce niveau. Nous nous intéressons ici aux analyseurs à travers *la boite à outils caesar/aldebaran*[*Boi94*].

Cette boite à outil;permet d'aborder l'ensemble des méthodes citées, de part la richesse des traitements supportés par le modèle graphe généré. Ses trois composantes sont utilisés dans cette perspective:

- *caesar*
- *aldebaran*
- *l'environnement open/caesar*

III.3.4.1 Caesar

le "noyau" de cette boite a pour fonction la génération du réseau de Pétri interprété puis la dérivation du graphe de simulation [*Cae94*, *Gar89*]. L'utilisateur a dès lors en sa possession un support d'étude, on ne peut plus complet, des services de téléphonie et de mise en instance d'appel. Toutes les séquences d'actions envisageables sont représentées indirectement par cette structure.

Ce squelette peut ensuite être examiné soit par rapport aux états, pour mettre en évidence les propriétés génériques (deadlocks, livelocks,..) , mais également par rapport aux événements pour les propriétés plus spécifiques portant sur des procédures de communication (établissement d'appel, requêtes de déconnexion, prise en charge d'un appel en instance,...).

Le réseau de Pétri généré est décomposé en unités à l'intérieur desquelles le contrôle est purement séquentiel. Ses caractéristiques sont ainsi le nombre d'unités, de transitions, de places, de marques (jetons initiaux) et de variables.

Le format de graphe obtenu en sorti est très lisible, la structure des transitions correspondant à celle acceptée par aldebaran:

(*état de départ*, *label*, *état d'arrivée*)

ce qui donne par exemple

(0, UserA !SETUP_REQ !ITA_Pri !ET_Ori !ITA_Cib !ET_Cib, 1)
(1, Valide_Request, 2)
(1, No_Valide_Request, 3)

Un tel exemple de spécification de graphe est fourni en annexe E.1.

III.3.4.2 Aldebaran

La traduction de la description formelle LOTOS en un graphe à l'aide de caesar n'est pas optimisée pour des raisons inhérentes aux algorithmes de traduction employés par cet outil (gestion des variables d'état notamment). Ainsi, le graphe résultant présente plus d'états et de transitions qu'il ne devrait, par rapport à la vue principale que peut avoir l'utilisateur du comportement de son système (autrement dit les labels des transitions et non les états).

Il est courrant de voir se multiplier les ordonnancements de transitions identiques au point de vue des labels, mais pas des états (hormis le dernier).

Traduction et optimisation

Rappelons qu'un état du graphe est caractérisé par un couple: une ensemble de places du réseau de Pétri et un ensemble de variable d'état. Celles–ci correspondent aux informations nécessaires quant à la prise de décision sur des comportements potentiels. C'est à ce niveau que la différenciation des états trouve son origine. Les états sont identiques du point de vue des places, les transitions le sont également, mais ces variables "internes" ont été modifiées de manière différente de part les événements réalisés en amont. Ainsi, au lieu de faire converger les comportements, en les rattachant à un même état, le graphe opère une

distinction des scénarios.

Réduction de graphe

Aldebaran a dès lors pour première fonction de réduire le graphe généré en ne s'attachant plus aux états mais uniquement aux labels. Pour réaliser cette opération, cet outil s'appuie sur la comparaison de graphe via des relations d'équivalence.

Quatre d'entre elles sont implémentées par la version utilisée (aldebaran 5.6):

- *La bisimulation forte*

  il s'agit d'un produit de synchronisation sur des arbres de simulation. Autrement dit, deux arbres sont bisimulables lorsque chacun des événements réalisables à un instant au niveau d'un arbre, l'est également au niveau de l'autre. *Un exemple de réduction de graphe via cette relation est fourni en annexe E.2.*

- *la bisimulation faible*

  même principe mais restreint aux événements observables (et non les tâches et les actions internes remplacées par l'action i).

- *la tau*.a bisimulation*

  idem que la bisimulation faible avec une implémentation différente de l'algorithme.

- *la relation de sécurité*

  les contraintes reposent sur les propriétés génériques (réinitialisation,terminaison,...)

Chacune de ses relations offre des niveaux différents de contraintes et leur présentation figure d'un ordonnancement de la plus contraignante (la bisimulation forte) vers celle qui l'est le moins (la relation de sécurité).

Ainsi, de l'utilisation de la première résultera un graphe ayant toujours au moins plus d'états que celui issu de la bisimulation faible; et ainsi de suite avec l'accroissement des critères d'abstraction. Cependant, il ne faut pas en conclure que la relation de sécurité sera toujours utilisée pour la simple raison qu'elle engendre le graphe le plus réduit. En effet, tout dépend de l'analyse souhaitée.

Ainsi, si cette dernière méthode parait la plus efficace au niveau de la réduction, cela peut cependant se faire au détriment de la qualité. La mise en évidence des propriétés spécifiques y est plus aléatoire, puisque le comportement décrit peut très bien avoir subi une amputation. Certains scénarios disparaissent ainsi. Au contraire, la bisimulation forte apparait moins puissante et moins attrayante de ce fait, mais elle engendre un graphe qui est complètement conforme à l'initial du point de vue dynamique. (c'est elle qui traduit le mieux la spécification). Elle joue donc un rôle central.

Résultats

Les résultats suivants sont tirés de l'utilisation de ces outils sur la spécification réalisée (orientation interface usager–réseau), mais limitée à la description du seul service de la téléphonie et avec une seule liaison traitée. ce graphe est ensuite réduit à l'aide des relations d'équivalences présentées.

réseau de Pétri	*caesar*	*bisimulation forte*	*bisimulation faible*	*réduction tau*.a*	*relation de sécurité*
variables:58 *unités:7* *places:121* *transitions:229* *marques:6*	*états: 10362* *transitions: 20421*	*états: 118* *transitions: 282*	*états:77* *transitions: 715*	*états: 77* *transitions: 175*	*états: 77* *transitions: 175*

Les caractéristiques du réseau sont juste fournies à titre d'information, car sans autres références, point de comparaison et les chiffres ne représentent pas grand chose alors.

En ce qui concerne le graphe, les résultats de la réduction sont assez surprenant: pour la bisimulation forte, qui garanti une adéquation totale à la spécification du service, graphe est réduit d'environ 88 fois pour

ce qui est des états et 72 fois pour les transitions! Ces chiffres sont encore plus importants pour les autres relations.

la comparaison de graphes

La seconde fonction de cet outil concerne la comparaison de graphes. Les applications sont nombreuses. Par exemple, dans le cas général, cela permet de tester l'implémentation d'un service par un protocole, en spécifiant formellement le protocole et en décrivant le service directement sous forme de graphe s'il s'agit d'une simple séquence d'événements, ou bien en le spécifiant si la complexité le nécessite.
Dans le cas de la spécification réalisée, cette approche a surtout permis de vérifier l'existence de certains scénarios à l'intérieur du "graphe de services" réduit.

III.3.4.3 L'environnement open/caesar

III.3.4.3.1 Modules fonctionnels

Cet environnement peut être considéré comme un véritable outil d'exploitation. Il ne s'agit plus ici d'un outil aval à caesar mais d'un substitut, dont l'objectif est de permettre à l'utilisateur d'influer sur la traduction de la description formelle en un graphe. L'environnement caesar a ainsi été "ouvert" ("*open/caesar*").
Le principe repose sur une séparation des fonctions de caesar, au niveau de la phase de simulation (après la création du réseau), en un ensemble de trois modules : *graphe, stockage et exploration* [*ope93*].

Le module graphe

Il procure une représentation en langage C des états et des labels du système de transition correspondant au source LOTOS et un ensemble de primitives de manipulation de ces éléments. Il permet ainsi de calculer l'état initial puis l'ensemble des états successeurs à un état donné et de travailler sur ces états (comparaison,...).

Le module stockage

Il procure les structures de données nécessaires à la mémorisation partielle ou entière du graphe, et les moyens de les exploiter.
Ainsi, parmi les structures sont répertoriées:

- *une table d'état avec accès hashcode.*
- *une file d'état pour le parcours en largeur d'abord.*
- *une pile d'état pour un parcours en profondeur d'abord.*
- *une table bitmap*

Le module d'exploration

Il s'agit de la partie principale et celle qui intéresse directement l'utilisateur. Elle utilise les deux modules précédents (graphe et stockage) et a pour objectif d'orienter la construction du graphe : choix du parcours en largeur ou en profondeur d'abord, sélection des successeurs, politique de mémorisation, ...

Dans cette perspective, l'utilisateur dispose d'un ensemble de librairies de fonctions C (seules les entêtes et les caractéristiques principales y sont présentées) :

- librairie "*standard*" : types basiques partagées par les modules,
- librairie "*graphe*" : représentation et primitives de manipulation d'états et de labels,
- librairie "*edge*" : déclaration et manipulation d'une liste de transitions,
- librairie "*stack_1*" : déclaration et manipulation d'une pile d'état,
- librairie "*hash*" : fonctions de hashcodage sur les labels ou les états,
- librarie "*bitmap*" : déclaration et utilisation d'une table bitmap,
- librairie "*table_1*" : déclaration et manipulation d'une table d'état,

(La librairie "stack_1" est fournie en annexe C à titre d'exemple.)

Celui-ci peut à l'aide de cette multitude de fonctions (environ 160!) développer ses propres outils d'analyse, lui permettant de générer un graphe dans le format qu'il souhaite (acceptable par un outil de gestion de graphe qu'il a développé auparavant, ou un générateur de MSC par exemple), de façon interactive, aléatoire ou encore exhaustive, avec détection de propriétés spécifiques dans le même temps.

III.3.4.3.2 Création de modules d'exploration

Un certain nombre de modules sont fournis avec la boite à outils. Il s'agit de:

"terminator" : détection de deadlocks à l'intérieur d'une spécification,
"exécutor" : exploration aléatoire d'une séquence d'événements dans le graphe,
"générator" : génération du graphe exhaustif au format aldebaran,
"prédictor" : estimations en taille et nombre d'états générables en vue d'une analyse,
"simulator" et *"xsimulator"* : simulation interactive (environnement Xwindow),
"réductor" : génération exhaustive (accompagnée d'une réduction "à la volée" en respect à la relation d'équivalence tau*.a.

Ces modules offrent un large panel d'exploitation, ainsi qu'un support de développement d'autres outils autour des fonctions C proposées.
Des modules d'exploration ont ainsi progressivement été réalisés, avant tout dans un souci d'estimation de faisabilité de cet outil, et non pas avec l'objectif d'une utilisation future (mais cela peut néanmoins être le cas) en tant que support de développement avec les contraintes de rigueur que cela suppose. Il serait, à priori, certainement plus efficace de développer des produits, rendants ces mêmes services, en aval de aldebaran pour bénéficier des effets de la réduction, et des algorithmes basés sur le traitement matriciel (matrice de routage,...: inopérant ici de part la structure et la quantité des états)[*Bos94*].
D'autre part, ce rapport ne prétend pas en exposer les détails de conception mais uniquement les concepts importants.
Les choix de réalisation de ces modules proviennent de constats sur l'utilisation des autres outils, et ne reposent pas sur des exigences formulées explicitement. Ainsi, furent d'abord développés les modules *"cycle"*, *"exhaustive"* et *"livelock"*.

Le module *"cycle"*

Ce module s'appuie sur la génération du graphe pour en présenter tout les cycles sous la forme de séquences d'événements. Cette tâche est nécessaire pour éviter de boucler lors du parcours.

Le module *"exhaustive"*

Celui-ci naît du constat que l'idéal serait d'obtenir de manière très claire une formulation de l'ensemble des scénarios possibles pour une spécification (non plus par une interprétation posteriori postériori du graphe généré mais directement sous forme de séquences).La validation s'en verrait d'autant plus facilitée et complète.
Néanmoins cette complétude est en pratique impossible à obtenir du fait de l'éventuelle présence de scénarios cycliques.
Plusieurs versions ont donc été réalisées avec gestion des cycles, de la redondance des scénarios et réduction de l'explosion combinatoire sur les événements internes. Néanmoins, ces "options" nécessitent des disponibilités mémoires et plus la spécification est complexe, plus la quantité d'état et leur taille est importante, et moins ces options sont exploitables. Dans les cas où les cycles ne sont pas gérés, la spécification ne doit pas être réinitialisable sous risque de bouclage infini (intégration d'actions "stop"). Dans le cas contraire, les contraintes consistent à ne parcourir au maximum qu'une seule fois un même cycle.

Le module *"livelock"*

Le module "terminator" permet de détecter les impasses sur synchronisations et les terminaisons "normales". "Livelock" permet de complèter les possibilités dans la mise en évidence de propriétés génériques,

en détectant et présentant les cycles absorbants. *Le programme correspondant est fourni en guise d' exemple en annexe D.2.*

Caractéristiques générales des modules

La structure de parcours

L'ensemble des modules développés au cours du tutorat font appel à la même structure, support à la progression dans le graphe de simulation.
Il s'agit d'une pile, définie et manipulée à l'aide des librairies de fonctions, et comportant trois composantes (*fig.18*):

- *un champ "label"*
- *un champ "state" (état)*
- *un champ "edge" (liste de transitions)*

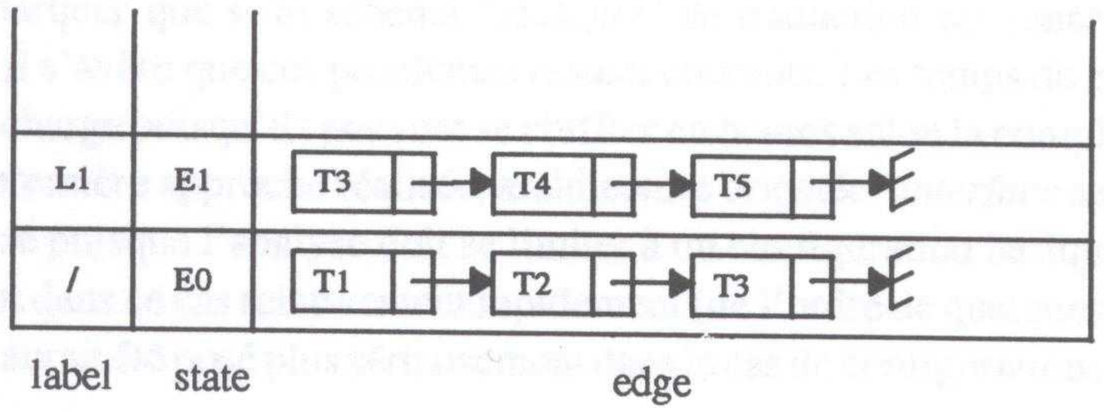


fig.18: structure de base à la génération de graphe sous open-caesar

La première occurrence concerne l'état initial E0. Le champ "edge" correspondant caractérise l'ensemble de ses états successeurs et des labels des arcs pour y parvenir, sous forme d'une liste de transitions.
Les deux premiers champs du second niveau, label L1 et état E1, caractérisent une transition qui faisait partie de la liste du niveau précédent, et qui a été développée. Ainsi E1 est un état successeur à E0 et l'arc *(E0, L1, E1)* existe dans le graphe.
Cette structure est la pierre angulaire des outils réalisés, car elle permet d'orienter directement la construction du modèle: sélection des successeur, tri des listes de transitions,examen de l'arbre de simulation,...

Table bitmap et fonction de hashcode

L'élaboration du graphe nécessite la mémorisation des états déjà traités. Cette mémorisation se fait par l'intermédiaire d'une table bitmap (booléenne) ou d' une table de liste de traversé d'état (critère booléen remplacé par nombre de passage), avec accès par hashcodage. Utilisée lors du parcours du modèle, cette dernière permet d'éliminer le risque de collision des hashcodes. Ces tables sont utilisées pour la détection de cycles.

III.4 Modélisation et contraintes

Si l'exploitation de la spécification à travers un graphe apparait attrayante, cela ne va pas sans poser un certain nombre de contraintes qui ternissent cette vision à priori.
Deux types de contraintes ont été répertoriées:

- *les contraintes spatiales et temporelles*
- *les contraintes combinatoires*

Les contraintes spatiales et temporelles

La contrainte la plus importante, puisque impérative, est celle qui assure que les modèles issus de la spécification seront finis. Ces restrictions qui aboutissent à un contrôle statique LOTOS inhibent toute instanciation récurssive potentiellement infini de processus. Autrement dit, cela se traduit au niveau du système décrit, par le gel des plans "liaison". Ainsi, la quantité d'appels doit être fixée à priori, alors que dans le

cas des interpréteurs hippo et smile, un nouvel appel peut intervenir au cours de la simulation. Il n'y a pas de contraintes statiques dans ce cas.
D'autre part la complexité des systèmes décrits et les algorithmes de génération utilisés par l'outil caesar, font que le graphe peut atteindre un nombre d'états très (trop) importants. Le problème d'"explosion des états" se pose alors.
Ce problème se pose aussi bien au niveau du réseau que du graphe dérivé. Il se peut ainsi que cette structure intermédiaire ne puisse être générée car le nombre de transitions ou de places est trop important. Or si point de réseau alors point de graphe.
Par exemple, cette génération a échouée dans le cadre de la spécification réalisée, dans la configuration de deux appels simultanés concurrents avec mise en instance. Le réseau comprenait lors de l' interruption plus de 110000 transitions. Pour ce qui est de l' analyse à travers les modèles, l' exploitation de la spécification s' est donc vue limitée aux configurations ne mettant en jeu qu' un appel de base (réduit à sa forme élémentaire).
Il faut également remarquer que si le schéma "*statique*" de traduction est sensé apporter un gain en mémoire et en temps, il s'avère que ces problèmes restent courants. Les temps de génération constituent également une lourde charge puisqu'ils peuvent se chiffrer en heures selon la complexité de la configuration retenue. Dans la première approche réalisée, architecture orientée "*interface usager-réseau*", le problème est peu rencontré puisque l'analyse doit se limiter à un configuration basique. Les générations de réseau et graphe se font dans ce cas relativement rapidement (de l'ordre de quelques minutes). Par contre, ce problème temporel aurait été posé plus sérieusement dans le cas de configurations plus "fournies", avec plusieurs appels.

Les contraintes combinatoires
Ces problèmes ont surtout pour origine l'explosion combinatoire lié au comportement asynchrone des entités de télécommunications, à tout les niveaux de la spécification (chacun des appels l'un par rapport à l'autre, l'appel sortant par rapport à l'appel entrant au niveau de chaque "plan" de liaison...). Cela a pour effet d'accroître de manière considérable la quantité de chemins et d'états dans le modèle.
Si cet aspect traduit les possibilités réelles du système décrit, il s'avère qu'il s'agit d'une véritable "plaie" au niveau de la simulation. En effet, les scénarios résultant ne diffèrent que sur l'ordonnancement des événements qui les composent, et ils ont tous la même valeur au sens de la vérification. Or en fonction des besoins de validation, il se peut que cette multilpication n'apporte rien si l'analyse de l'une des séquences suffit à se forger un avis.
Cela a pour conséquence l'impossibilité de génération du graphe mais d'autre part, si celui–ci l'est, la quantité de transitions fait qu'il est souvent inexploitable directement par l'utilisateur. Si cela est réellement intéressant pour des spécification réduites où le graphe ne comporte que quelques dizaines d'états, (et peut être tracé), il s'avère que c'est rarement le cas, même après réduction du graphe à l'aide d'aldebaran.

III.5 Les solutions envisagées
L'ensemble de ces contraintes a des répercutions importantes sur la qualité de l'analyse de la spécification réalisée. Dans un cas, aucune analyse (basée sur un graphe) ne peut être réalisée, et dans l'autre la richesse d'information devient un handicap!
Malgré ces inconvénients, cette méthode de validation reste incontournable puisqu'elle offre une plus grande complétude par rapport aux autres outils abordès. Des adaptations sont donc nécessaires soit au niveau matériel (puissance et capacité mémoire), soit au niveau des outils de génération de modèle (caesar: optimisation de la conception), soit au niveau de la spécification. C'est cette dernière forme qui a été envisagée.
Trois types de modifications ont été abordées:
- *des restrictions sur l'analyse,*
- *une adaptation de la spécification,*

- *le choix d'une nouvelle orientation de spécification.*

III.5.1 Restrictions sur l'analyse

Deux cas sont distingués ici: les problèmes liés à l'exploitation du graphe issu de caesar, et ceux inhérents à la non génération de ce "squelette" support d'analyse.
Dans le premier cas, la quantité de transitions fait que cette nouvelle forme de représentation de la spécification ne peut être utilisée directement. Comment aborder des milliers de transitions? Au niveau de la seule analyse, les solutions consistent à utiliser (éventuellement concevoir) des outils qui réduisent cette richesse de données, à savoir *aldebaran* et *open/caesar*.
Dans le second cas (impossibilité de génération du modèle complet), open/caesar peut s'avérer encore très utile dans l'objectif d'examen d'un sous-ensemble des chemins si le calcul du réseau n'a pas posé de difficulté. Dans le cas contraire, l'analyse ne peut reposer sur ce support, et un retour aux "interpréteurs" s'avère nécessaire.

III.5.1.1 Aldebaran

Outre la réduction et la comparaison de graphes, qui entrent pleinement dans ce cadre, cet outil offre des possibilités de calcul de sous-graphe (*option -hide*). L'utilisateur peut préciser un ensemble d'événements, dont il souhaite obtenir les relations via le modèle, dans un fichier avec l'option "*all but*". Le résultat correspond à un graphe ne comportant que les événements précisés pour transitions (labels) visibles; celles_ci étant liées par des transitions invisibles (événement "i" au sens LOTOS). Cela permet de réduire nettement la quantité d'information, avec la contrainte de devoir aborder la spécification par étapes.
Par exemple, il est possible par ce biais d'obtenir un sous-graphe ne comportant que les primitives de services (dérivées de SETUP, REPORT, RELEASE, et DISCONNECT) et ainsi d'étudier l'ensemble des combinaisons possibles de façon plus aisée. L'approche inverse, qui consiste à préciser les événements qui ne doivent pas apparaître dans le graphe, est également possible.

III.5.1.2 L'environnement open/caesar

Certains des modules présentés dans le paragraphe III.3.4.3.2 entrent dans le cadre décrit ici (simulation aléatoire notamment). Outre les trois modules "*cycle*", "*exhaustive*" et "*livelock*", d'autres modules ont été créés afin de palier les problèmes de richesse d'information et de génération de graphe complet.

Le module "recherche"

Ce module a pour objectif de fournir l'ensemble des scénarios (sous certaines contraintes liées aux cycles) intégrants un événement précisé par l'utilisateur. Ces scénarios ne sont pas présentés sous forme de spécification de graphe mais directement d'un ensemble de séquences (plus convivial). Ainsi, il est possible d'obtenir, par exemple, les différentes combinaisons d'actions incluant une mise en instance d'appel (événement UserB !SETUP_IND ... !No_Available_Info_Channels).
En fonction des caractéristiques de la spécification, il est parfois nécessaire de restreindre cette approche à la présentation des scénarios aboutissant (et non plus intégrant) à un événement donné. Ce cas intervient lorsque la spécification est réinitialisable et que des contraintes de terminaison n'ont pas été insérées.
Plusieurs versions ont été réalisées avec gestion des cycles, de la redondance et de l'explosion combinatoire.

Le module "fromto"

Celui-ci peut être vu comme une extension de "recherche". Il présente en résultat l'ensemble (avec la contrainte des cycles) des scénarios ayant pour extrémités des actions bien précises, toujours directement sous forme de séquences.
L'utilisateur a ainsi la possibilité de tester l'existence et d'analyser les différentes procédures menant d'une action liée à un appel, à une autre. Par exemple, ces actions peuvent concerner la présentation d'un appel, en instance ou non, d'une part (UserB !SETUP_IND ...), et sa prise en considération d'autre part

(UserB !SETUP_RESP ...). Cela peut également concerner les phases de déconnexion une fois un appel pris en ligne.
Afin de pouvoir être exploité avec un large spectre de configuration, même complexes, des versions plus ou moins "évoluées" ont été réalisées. Certaines intègrent la gestion des cycles, de la redondance ou de l'explosion combinatoire ou encore se limitent à la présentation d'un seul scénarioso voire même à sa seule détection (résultat sous forme de booléen). *Ce programme est fourni à titre d'exemple en annexe D.1 et une utilisation en annexe E.3.*

III.5.1.3 Retour aux "interpréteurs"

Dans le cas où le modèle réseau ne peut être conçu, l'analyse ne peut reposer (dans le cadre de notre étude bien sûr) que sur l'utilisation d'hippo ou smile, et donc une simulation interactive. La qualité d'analyse s'en ressent nettement et il est alors sans doute préférable de passer au niveau "supérieur" de restrictions, à savoir opérer des modification sur la spécification, afin de pouvoir baser l'analyse exhaustive sur ces graphes.

III.5.2 Adaptation de la spécification

Les contraintes d'analyse sont suffisamment importantes ici pour que la spécification ne soit plus exploitée comme telle, c'est à dire dans sa version "idéale" la plus conforme au comportement du système décrit, mais réduite aux actions pertinentes vis vis à de la vérification. Cette adaptation peut être effectuée à deux niveaux: Les données et/ou le contrôle.

III.5.2.1 Au niveau des données

Les restrictions consistent à limiter les occurrences d'installations terminales ou de terminaux, susceptibles d'intervenir au niveau d'un appel diffusé par exemple (puisque tous les cas sont abordès dans le graphe). Dans le même registre, la quantité d'attributs pour ce qui est des primitives REPORT, DISCONNECT ou RELEASE peut être réduite et ceux-ci envisagés pour certains de manière abstraite sous la forme d'une seule offre "Cause" ou "Alerting". Les conséquences sur l'analyse ne sont pas dommageables puisque les scénarios issus de ces événements ont la même valeur et il ne diffèrent que par ces attributs.

IV.5.2.2 Au niveau du contrôle

Les restrictions peuvent intervenir à divers niveaux: comportement de chacune des "entités" les plus internes (appel sortant, appel entrant, zone de transit ou encore contrôleur), mais également au niveau de l'architecture.

Dans le premier cas, certaines branches de développement sont "élaguées" (par rapport au diagramme SDL). Ainsi les tests initiaux de validité de requête, ou de succès de routage, peuvent être laissés de côté (validès dans un autre temps). Le but est ici de valider tour à tour les scénarios envisageables, en ignorant les autres cas. Ainsi seront analysées séparément les procédures d'activation de temporisateurs (*timer1, timer2 et timer3*), les déconnexions et les prises en charge d'appel avec ou sans alerte(s) préalable(s).
De plus chacune de ces branches peut être conclue par une terminaison "normale" (action "*stop*"), ce qui réduit fortement la taille des graphes générés. Cela prive la description de sa propriété de réinitialisation mais celle-ci peut faire l'objet d'une validation ultérieure, avec des restrictions sur les comportements internes (analyse plus axée sur les extrémités).

Par rapport à ces comportements, l'une des principales justifications, quant à la quantité de scénarios, provient des procédures de déconnexion, invocables dès la requête d'établissement d'appel pour l'appelant (primitive SETUP_REQ), dès la prise de ligne pour l'interlocuteur (primitive SETUP_RESP). Cela a pour conséquence d'accroître de manière considérable le comportement dynamique du système. Afin de palier ce problème, ces phases de déconnexion sont retardées et autorisées uniquement une fois le dialogue établi (primitive DIALOGUE).

Une autre forme de restriction consiste à intervenir au niveau de l'architecture de processus. Une application concerne le retrait de la zone de transit, dont la seule tâche est d'introduire l'asynchronisme entre interfaces au niveau d'une liaison. Ainsi les processus traitant des fonctionnalités "appel sortant" et "appel entrant" sont directement synchronisés. L'explosion combinatoire liée à l'asynchronisme de ces deux entités est de ce fait en partie réduite (*fig.19*).

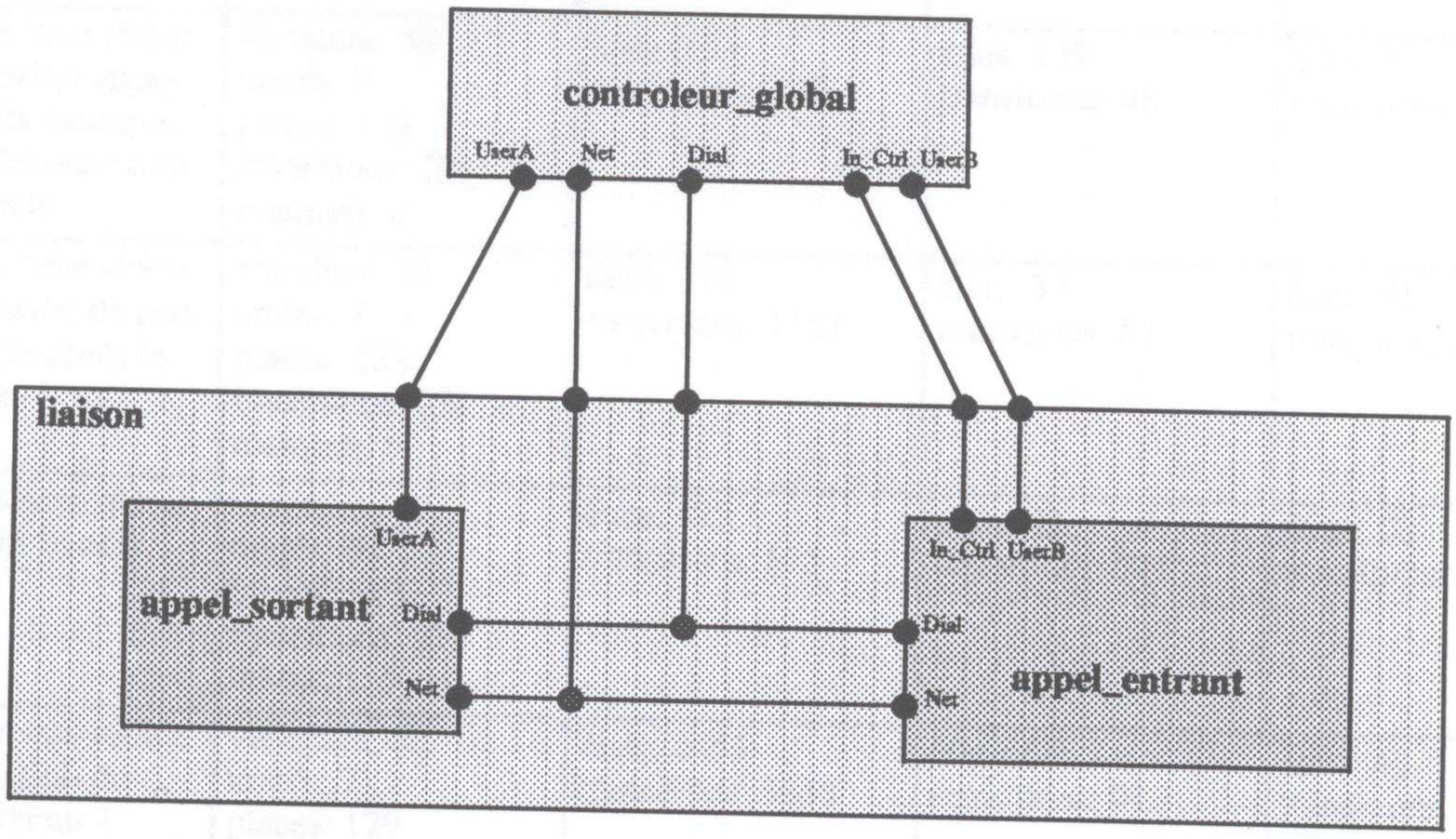


fig.19: structue de contrôle et synchronisation directe entre interfaces

Enfin, une autre possibilité consiste à restreindre un comportement aux seules actions qui interviennent dans la gestion des ressources (disponibilité en canaux ou en terminaux) et qui sont de ce fait incontournables. Dans ce cas, le plan "liaison"est réduit à un simple processus, qui décrit une vision "globale" de ce comportement par une simple combinaison d'événements. Cela permet d'une part, de cibler l'analyse sur les ressources mais également sur un appel bien précis en réduisant les autres à leur "dynamique basique" (par exemple, un appel de base pourra être se résumer à quelques primitives de services nécessaires alors qu'un appel en instance, objet de l'étude, pourra être décrit complètement. Ce cas sera détaillé à travers la seconde architecture envisagée).

III.5.2.3 Résultats

Une large gamme de ces mesures restrictives ont été appliquées à la spécification réalisée. Les valeurs présentées ci-dessous concernent, pour certains de ces cas, les caractéristiques du réseau, du graphe issu de caesar puis réduit avec aldebaran. Ces chiffres, pour chacune des configurations envisagées, sont importants et ils sont vus comme le baromètre de la spécification réalisée. En effet, la qualité de cette description dépend du degré de confiance qui lui est attribuée, et donc de la complétude de l'analyse réalisée. Or celle-ci repose surtout sur l'étude de l'automate à états associé.

L'appel considéré est dit (très) "réduit" car il ne comporte ni tests de validité, ni alertes, ni activation de temporisateurs mais se limite à une requête d'établissement d'appel, un dialogue et une déconnexion. Les configurations évoquées diffèrent de part les possibilités de déconnexion offertes (réduites à l'appelant, dès que possible ou pas avant le dialogue), l'intégration de l'activation du temporisateur timer1 et la synchronisation directe entre interfaces. Chaque cas envisagé est une évolution du précédent.

configuration	réseau (caesar)	graphe (caesar)	bisimulation forte (aldebaran)	bisimulation faible (aldebaran)
appel de base réduit déconnexion dès SETUP_REQ pour l'appelant, dès SETUP_RESP pour l'appelé	variables: 58 unités: 7 places: 121 transitions: 229 marques: 6	états: 10362 transistions: 20421	états: 118 transitions: 282	états: 77 transitions: 175
appel de base réduit déconnexion appelant aprés dialogue. pas de deconnexion de l'appelé.	variables: 58 unités: 7 places: 122 transitions: 233 marques: 6	états: 103 transitions: 133	états: 139 transitions: 48	états: 25 transitions: 31
appel de base réduit déconnexion de part ou d'autre aprés le dialogue	variables: 58 unités: 7 places: 122 transitions: 233 marques: 6	états: 726 transitions: 1153	états: 55 transitions: 83	états: 41 transitions: 66
appel de base réduit idem sans transit	variables: 47 unités: 6 places: 119 transitions: 233 marques: 5	états: 425 transitions: 615	états: 42 transitions: 57	états: 28 transitions: 40
appel de base réduit réhabilitation du temporisateur 1	variables: 62 unités: 7 places: 129 transitions: 251 marques: 6	états: 2455 transitions: 3848	états: 61 transitions: 91	états: 46 transitions: 73
appel de base réduit idem sans transit	variables: 51 unités: 6 places: 119 transitions: 233 marques: 5	états: 1520 transitions: 2210	états: 47 transitions: 64	états: 32 transitions: 46

Les résultats obtenus montrent l'importance de l'influence de la phase de déconnexion sur la taille du réseau et du graphe générés. Ces différences s'amoindrissent avec les réductions, mais l'analyse du graphe est en partie réalisée en amont de ces réductions et là les écarts sont très importants. Ainsi, le graphe passe de plus de 20000 transitions, dans le cas où les déconnexions sont autorisées dès la requête pour l'initiateur et dès la prise en ligne pour l'interlocuteur, à un peu plus de 1100 lorsque ces déconnexions sont autorisées uniquement une fois le dialogue établi!

Bien évidemment, ces différences seraient encore plus importantes si toutes les branches de développement avaient été intégrées. Les chiffres résultants de l'incorporation du temporisateur timer1 mettent ce problème en évidence. D'ailleurs, pour rappel, le réseau correspondant à la spécification complète avec déconnexion dès que possible, ne peut être calculé (stoppée alors que la taille du réseau en transitions est supérieure à 110000!).

En bilan, ces mesures sont nécessaires de part le caractère impératif posé par une validation basée sur un modèle graphe. Mais si les répercutions apparaissent positives sur la quantité d' états et de transitions, ils ont pour autre conséquence de dénaturer la description du système, du fait des restrictions sur les données ou sur l' asynchronisme. La spécification est alors dédiée à l' analyse et elle ne se base plus seulement sur l' orientation des recommandations de références. Elle fait l' objet d' une adaptation plus ou moins importante en fonction des contraintes de traduction, et ne saurait dès lors être utilisée comme telle dans un objectif d' implémentation.

D'autre part, ces restrictions ne sont pas toujours facilement réalisables de part les synchronisations, bien

que le caractère très modulaire de la description, avec séparation des données et des différents services, est sensé rendre cette opération plus aisée. Il est ainsi courant d'aboutir à des impasses sur synchronisation dans la mesure où un comportement a été éliminé au sein d'une entité, mais pas au niveau de l'entité duale. L'exploitation de la spécification orientée "*interface usager-réseau*" trouvant ses limites dans les outils disponibles (notamment pour ce qui est de la mise en instance d'appel), une nouvelle orientation à été choisie suite aux constats tirés des analyses précédentes.

III.5.3 Choix d'une nouvelle architecture

III.5.3.1 Caractéristiques

Le principal constat concerne la nécessité de réduire les unités, et de ce fait l'asynchronisme, source de l'explosion combinatoire. L'approche par "plan" (basée sur le contrôleur global adapté) est conservée mais la spécification des liaisons au niveau de ces plans est modifiée.
Ainsi la nouvelle approche est caractérisée par :

- *une gestion des ressources au niveau du réseau*, et non plus au niveau des entités terminales "plans" (appel sortant, appel entrant).
- *une non-séparation des services* de téléphonie et de mise en instance.

Ce réseau ne sert plus seulement à gérer l'asynchronisme entre les entités terminales (séquencement de mêmes actions, sur des portes de synchronisations différentes) mais il opère un véritable contrôle de flux et une gestion des données locales à un appel. Il est dèsormais dénommé "contrôleur point à point". Une requête est ainsi transformée en indication, et une réponse en confirmation (SETUP_REQ en provenance de l'initiateur reproduit en SETUP_IND vers l'installation cible, avec mise en instance ou non selon le contexte, par exemple)
D'autre part, les entités terminales décrites ne sont plus seulement les interfaces, mais sont étendues aux installations "globales" d'abonnés: interfaces usager-réseau, et terminaux ne sont plus distingués. Le comportement spécifié est associé à l'ITA de manière abstraite. L'approche est ainsi dite orientée "Installation Terminale d'Abonné"(*fig.20*).

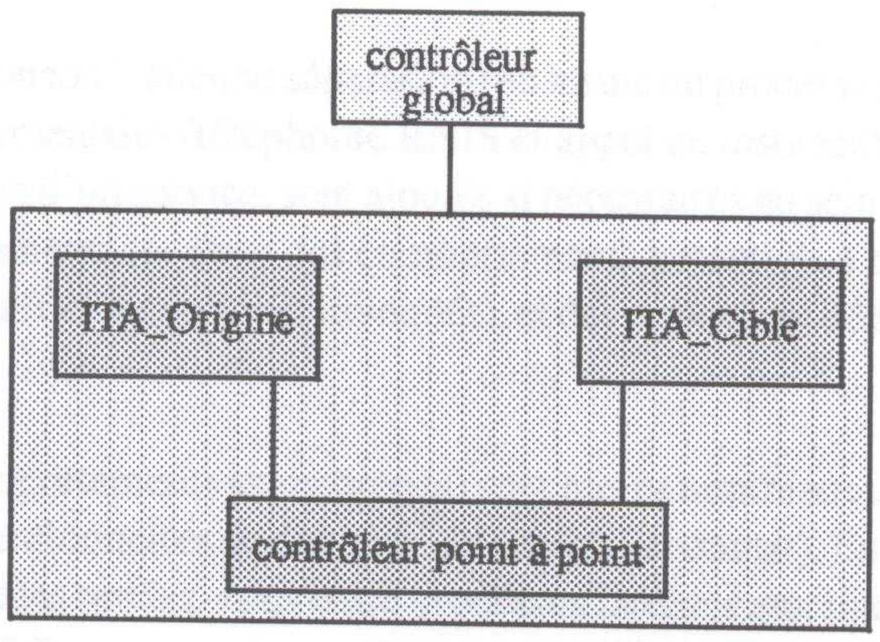


fig.20: structure de base d'un appel dans la spécification orientée "ITA"

Le flux d'information est toujours présenté au niveau de l'interface usager réseau, mais par rapport à l'orientation précédente (*fig.21*), les fonctionnalités des interfaces ont été intégrées au réseau (*fig.22*).

Ainsi, au niveau du contrôle LOTOS, le processus "contrôleur point à point" regroupe sous une entité monobloc (pas de distinction) les fonctions des interfaces ainsi que le transit. Les ressources locales sont gérées, de manière centralisée, alors qu'auparavant elles étaient disséminées entre les interfaces. Ce processus possède de ce fait une tâche pratiquement identique au contrôleur global (processus récurssif carac-

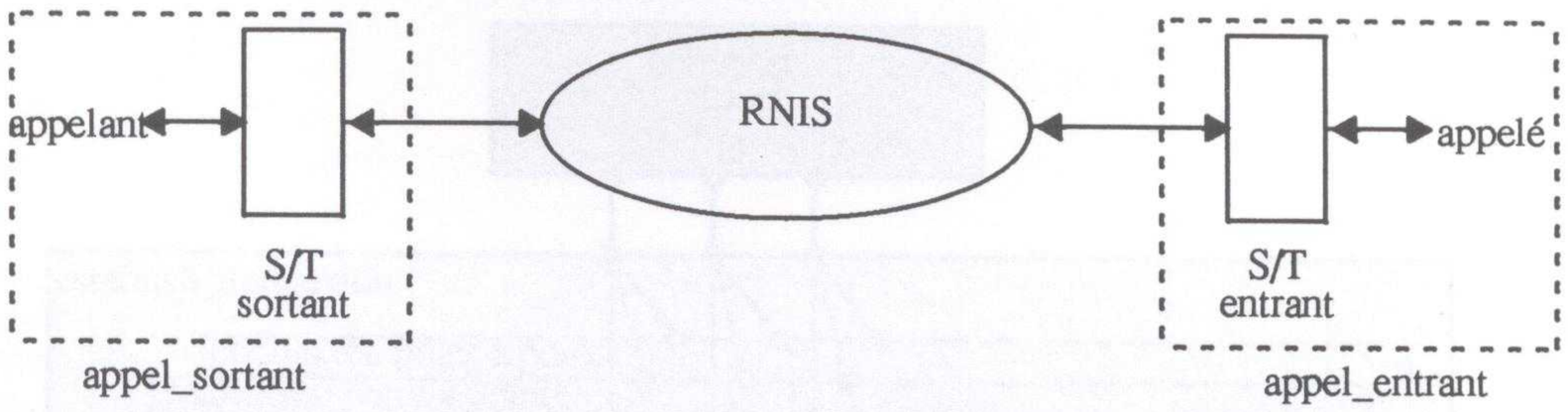


fig.21: entités de communication dans la spécification orientée "interface"

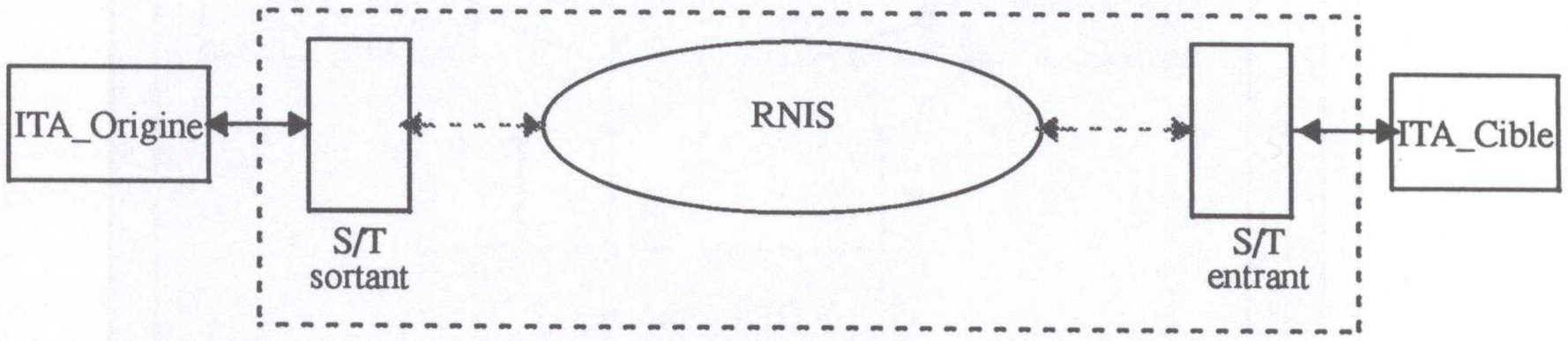


fig.22: intégration des fonctionnalités des interfaces dans le réseau

térisé par des disjonctions de séquencements avec contraintes de validation et mise à jour de données), si ce n'est que les ressources à gérer ne le sont pas.

En ce qui concerne les installations terminales d'abonnés, le comportement est décrit sur la base de séquences possibles de primitives de services (SETUP, REPORT, DISCONNECT, RELEASE, DIALOGUE...). Tout comme pour les interfaces (appel sortant, appel entrant), leur comportement est divisé selon deux aspects ITA_Ori et ITA_Cib, non symétriques, qui correspondent à la gestion de l'appel vis à vis de l'initiateur, pour le premier, vis à vis de l'ITA ciblée pour le second. Le comportement potentiel d'une ITA à un instant donné est donc partagé dans les différents plans où ses terminaux sont concernés.

Enfin, au niveau d'un plan "liaison", aucune séparation (en terme de processus) n'est effectuée entre service de base et service supplémentaire (téléphonie RNIS et appel en instance). Les nouveaux comportements, inhérents à l'intégration d'un service, sont ajoutés si nécessaires au sein des installations, alors que la validation des flux d'événements (et donc des comportements potentiels à ces installations) se fait par le biais de booléens (pour identifier les services invoqués), et des ressources déjà référencées dans l'approche précédente.

Les unités au sens réseau (comportements asynchrones) sont moins nombreuses, trois (ITA_Ori, ITA_Cib et contrôleur) au lieu de six (gestionnaires de données, de services et transit). En conséquence, les synchronisations le sont aussi puisque les rendez-vous entre processus internes aux interfaces ont disparus, et que les autres ont été réduits (*fig.23*).

Bien que ces "boites fonctionnelles" différent des précédentes, une certaine similitude apparait au niveau des rendez-vous. ainsi :

- *L est une synchronisation locale qui lie les deux types de contrôleurs*. Elle est équivalente à In_Ctrl dans l'autre perspective : test de disponibilité de canaux, de terminaux, ...
- *G, synchronisation globale, est à rattacher aux synchronisations Out_Net et In_Net*. Elle assure le transfert d'information entre les blocs et la gestion de ces actions vis à vis des ressources (validation et mise à jour).
- *D, synchronisation de dialogue, est équivalente à Dial et complète au niveau d'un plan.*

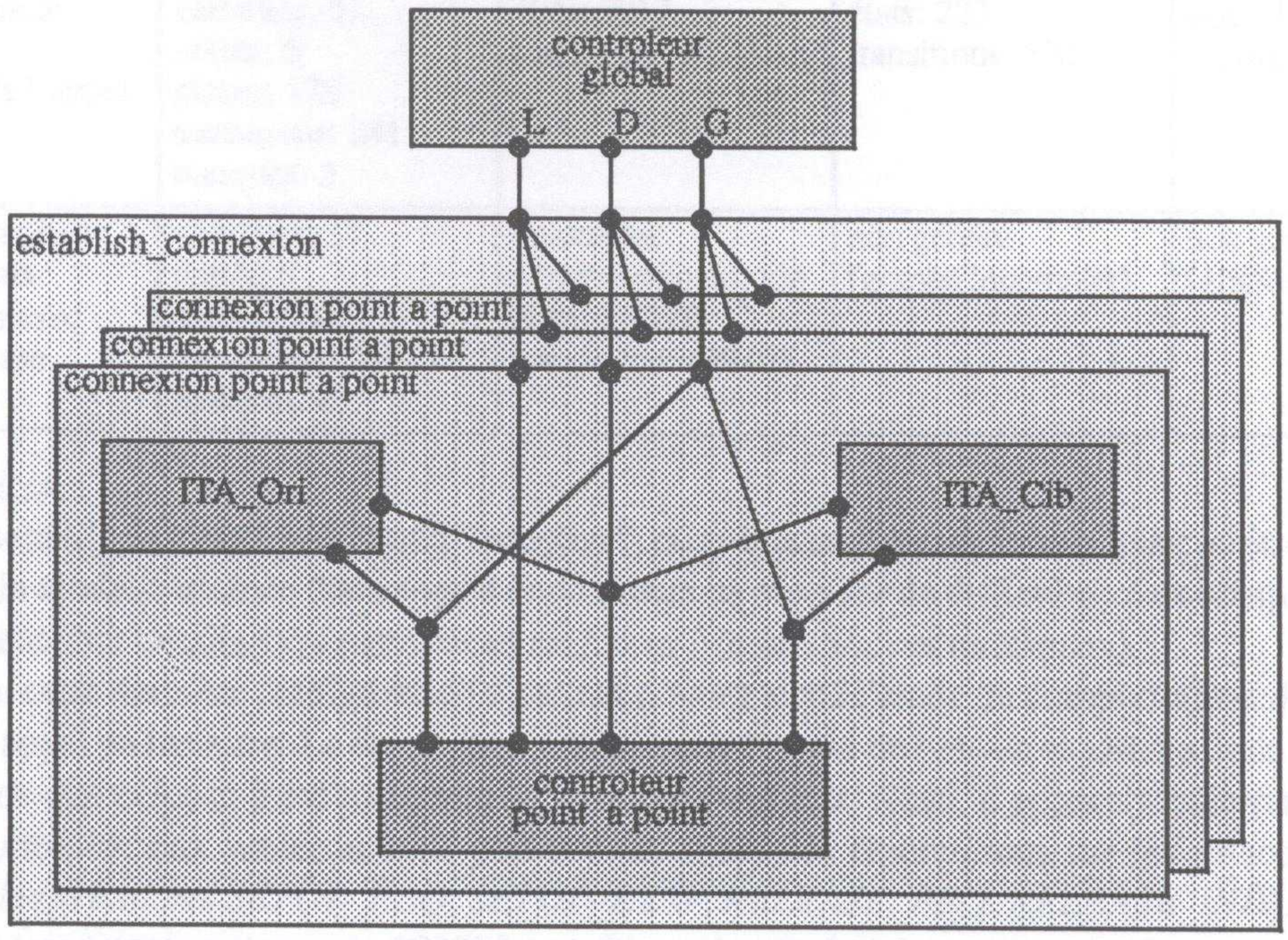


fig.23: synchronisations et entités de communication dans la spécification orientée "ITA"

La partie contrôle de la spécification obtenue est fournie en annexe F.

III.5.3.2 Résultats

Par le biais de cette spécification, des configurations d'appels plus ou moins complexes, avec mise en instance ont pu être analysées. La spécification obtenue est plus exploitable que sa devancière, et son analyse exhaustive basée sur un graphe pose moins de difficultés.

"Centralisation" des contrôles de flux et de ressources, diminution des processus asynchrones ont pour conséquences la diminution des événements subalternes, des combinaisons d'ordonnancements, et de la taille des états (moins d'unités et de variables).La génération du graphe par caesar aboutit ainsi plus souvent.

Le tableau suivant présente les chiffres caractéristiques des graphes issus de quelques configurations du système:

configuration	réseau (caesar)	graphe (caesar)	bisimulation forte (aldebaran)	bisimulation faible (aldebaran)
1 seul appel	variables: 55 unités: 5 places: 124 transitions: 234 marques: 4	états: 110 transitions: 132	états: 51 transitions: 74	états: 36 transitions: 55
2 appels concurrents	variables: 91 unités: 8 places: 239 transitions: 460 marques: 7	états: 49887 transitions: 108288	états: 3645 transitions: 14558	états: 1152 transitions: 3480

2 appels concurrents réduction de l'appel de base	variables: 57 unités: 6 places: 128 transitions: 241 marques: 5	états: 3017 transitions: 5270	états: 223 transitions: 520	états: 168 transitions: 408
3 appels simultannés dont deux concurrents avec appels de base réduits	variables: 57 unités: 7 places: 133 transitions: 250 marques: 6	incomplet pénurie de mémoire états: 169604 transitions: 439107	non généré (graphe incomplet)	non généré (graphe incomplet)

Commentaires

Contrairement à la description orientée "*interface*", où même un appel de base n'avait pu être traduit complètement, la modélisation de deux appels concurrents ne pose pas de difficultés. Les comportements qui s'y rapportent sont décrits sans restrictions majeures (installations et terminaux).
Néanmoins, cette dernière configuration a surtout pour objectif l'étude du comportement du système vis à vis d'une appel en instance. La description complète de l'appel de base ne se justifie donc pas vraiment, d'autant plus que celui-ci à été logiquement validé auparavant. Ainsi, il apparait intéressant de réduire leur comportement aux seules actions pertinentes, ayant des répercutions sur les ressources *(actions basées sur les primitives SETUP_REQ, SETUP_RESP, RELEASE_REQ, RELEASE_IND*). La réduction obtenue est importante puisque de 108288 transitions, le graphe n'en comporte plus que 5270!.

Par contre, il apparait que cette modélisation n'aboutit pas dans le cas de trois appels simultanés dont deux concurrents. La génération du graphe est stoppée au 146300ème état alors que celui-ci doit en comporter 169604. Cette interruption est inhérente à la capacité mémoire de la station de travail (la taille du graphe à cet instant est de 36 Mo). De toute manière, la quantité de transitions (439107) aurait rendu ce graphe inexploitable (vis à vis des des outils d'analyse envisagés dans le cadre de ce travail).
Ce problème est intéressant puisque si l'on s'intéresse aux caractéristiques des réseaux, on remarque que celui associé à cette configuration de trois appels (4ème rangée du tableau) possède moins de variables, d'unités, de places, de transitions et de marques que celui associé à la configuration de deux appels simultanés non réduits (2ème rangée)! A priori, le graphe généré aurait donc du comporter moins d'états et de transitions mais cela n'est pas le cas. Cela montre que la réduction de l'asynchronisme (et donc des unités réseau) ne suffit pas et que ces caractéristiques ne prédisent en rien celles du graphe de simulation!

Bilan

Cette spécification a donc rendu possible l'analyse, supportée par les graphes, de configurations plus complexes que sa devancière. Ses atouts sont sa *compacité* et ses *possibilités d' exploitation*. Par contre celle-ci est *moins lisible*, et *moins évolutive* (isolation de scénarios et intégration de nouveaux services).
Par rapport au service d'appel en instance, le seul point obscur concerne la vérification de la mise en instance pour cause d'indisponibilité de canaux d'information au niveau de l'installation appelée. Cet aspect nécessite en effet trois appels simultanés dont deux concurrents, or la traduction aboutit à un échec dans ce cas.
La solution consiste ici a se retourner vers les différentes restrictions envisagées dans le paragraphe III.5 (restrictions sur analyse ou spécification). Cette forme de mise en instance a ainsi pu être simulée à travers certains modules d'exploration (*recherche*, *fromto*, *exécutor* (aléatoire),...) et les interpréteurs *hippo* et *smile*.

IV . CONCLUSION

Il convient de conclure ce rapport en dressant un bilan de l'étude réalisée, à travers la spécification LOTOS des services de téléphonie RNIS et d'appel en instance. Bien évidemment, ma faible expérience ne saurait me permettre d'élargir ce point de vue au domaine de "la spécification formelle". Les lignes suivantes ne prétendent pas non plus contenir que de la vérité, mais présentent de simples constats personnels.

Les recommandations

Ce rapport met en évidence la nécessité de la vérification de protocoles et services à travers leur description formelle. L'utilisation des recommandations de l'ITU en est assurément une preuve. En effet, celles-ci ne peuvent être considérées que comme des supports d'analyse partielle. De part leur aspect ambigu (dû à la prose) et leur description trop statique des services, elles conduisent à une étude trop sujette aux interprétations de chacun, et limitée en terme de complétude.
Néanmoins ces documents peuvent être utilisés comme supports de référence à une description plus rigoureuse, voie empruntée ici, bien que cela ne soit pas toujours aisé. C'est ainsi le cas aux niveaux des attributs de primitives et de la distribution des comportements (interactions et tâches). Les recours à l'intuition y sont parfois nécessaires.
Ce manque d'information peut sans doute être rapporté au fait que les diagrammes fournis ne constituent qu'une partie réduite de la spécification SDL. Certaines ambiguïtés seraient certainement levées si référence était faite à cette description complète.
Malgré ces inconvénients, ces informations sont incontournables; cela tient au fait qu'il n'existe apparemment pas de meilleures références.

Les spécifications

Si les descriptions LOTOS réalisées (orientées interface ou ITA) offrent des perspectives d'analyse plus complète, elles mettent aussi en évidence un paradoxe. Les outils supports au langage LOTOS représentent un plaidoyer pour le recours à cette technique, mais ils témoignent également de sa limite actuelle. L'expérimentation de la boite à outils caesar-aldebaran justifie pleinement ce point de vue. La qualité de la spécification dépend surtout des possibilités d'examen offertes. Son étude exhaustive passe ainsi par la traduction en automate à états, voie empruntée par caesar. Or, à travers les problèmes d'explosion d'états et de richesse d'information, les expérimentations menées mettent en évidence les contraintes posées par la génération de ce modèle.
Le caractère impératif (en théorie) de la validation implique alors une recours à une simulation interactive, donc incomplète, ou à des restrictions. Cela aboutit à une description parfois dénaturée, et dédiée à l'étude graduelle du système par le biais de critères d'abstraction.
D'autre part celle-ci ne peut plus dans ce cas, faire figure de support à l'implémentation, ce qui réduit son spectre d'utilisation.
Il apparaît ainsi que si LOTOS tient sa richesse de sa sémantique formelle et de la puissance de ses opérateurs, son avenir est surtout lié au développement d'outils supports performants en terme de complétude d'analyse et d'ergonomie.

D'un point de vue général, ce tutorat m'a apporté plusieurs satisfactions. Il m'a permis d'aborder un nouveau langage, LOTOS, ce qui constitue un bon complément à l'étude du langage ESTELLE dans le cadre du cursus à l'ENSSAT. Deux des trois standards des techniques de description formelle ont ainsi été exploités. D'autre part, la nature du sujet a nécessité une approche de certains aspects du RNIS, au niveau raccordement ou services, et ne s'est pas limité à la simple utilisation d'un langage.
Celui-ci m'a fait prendre conscience des besoins en matière de méthodes et supports de spécification, mais aussi de la difficulté d'implantation parmi les groupes de développement. Dans ce domaine les exigences en matière de formalisme sont importantes mais la prise de contact avec les techniques inhérentes est souvent ardue.
Enfin, je finirais par souligner la place prépondérante faite à la réflexion et à l'initiative personnelle, dans ce stage, notamment en matière d'orientation de spécification et d'expérimentation des outils.

Bibliographie

[*Aut89*] "Systèmes de transitions finis et sémantique des processus communicants", A. Arnold, 1989.

[*Boi93*] Une boite à outils pour la vérification de programmes LOTOS", J.C. Fernandez, H.Garavel, L. Mounier, A. Rasse, C. Rodriguez, J. Sirfakis, Institut IMAG., 1993.

[*Bos94*] Cours de graphe, G. Bosc, ENSSAT, 1993.

[*Bou93*] Les couches du modèle OSI, M. Bouri, ENSSAT, Cours de réseau, 1993.

[*Cae94*] "Caesar reference manual", H. Garavel, Institut IMAG;

[*Com87*] "Les caractéristiques du RNIS", M. Clost, A. Vomscheid, revue Commutation et transmission3: le RNIS, 1987.

[*For90*] "Formal design of communication protocols", H. Hansson, B. Jonsson, F. Orava, B. Pehrson, XIII international switching symposium, 1990.

[*Gar89*] "compilation et vérification de programmes LOTOS", H. Garavel, institut IMAG, sujet de thèse, 1989.

[*Gro89*] "Vérification de propriétés logiques des protocoles et systèmes répartis par observation de simulation", R. Groz,CNET Lannion, sujet de thèse, 1989.

[*Hip87*] "Hippo manual page V2.1", J. Tretmans, University of Twente, 1987.

[*Ing93*] "L'ingénierie des protocoles", P. Angosto, B. Caillaud, interéditions 1993

[*Lot87*] "Introduction to the ISO specification language LOTOS", T. Bolognesi, E. Brinksma, Computer networks and ISDN systems, 1987, pp. 25-59.

[*Loy92*] "An industrial experimentation on LOTOS technique for better software development", B. Loyer, H. Nirschl, P. San Martin, XIV international switching symposium, 1992.

[*Ope93*] "The open/caesar reference manual", H. Garavel, Institut IMAG, 1993

[*Rec88*] Recommandations de l'ITU sur les services RNIS, série I.200, IXème assemblée plénière, Melbourne 1988.

[*Rni88*] "Le RNIS: techniques et atouts", G. Dicenet, Collection technique et scientifique des télécommunications, CNET-ENST, 1988.

[*Smi93*] "Smile User manual - release 3.1", H. Eertink, University of Twente, 1993.

[*Top91*] "Topo: Quick reference", J. A Manas, Université de Madrid, 1991.

[*Zim85*] "Réflexions sur l'ingénierie des protocoles", H. Zimmerman, "protocole specification, testing and vérification", IFIP 1985, pp. 123-133.

Liste des figures