Intégration de systèmes dans un réseau local d'ordinateurs : application au centre de calcul réparti

Jean Angelides

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT.ETIENNE

INSTITUT NATIONAL POLYTECHNIQUE
DE GRENOBLE

N° d'ordre : 17 IS

THÈSE

présentée par

Jean ANGELIDES

pour obtenir

le grade de Docteur de 3° cycle

"SYSTEMES ET RESEAUX INFORMATIQUES"

INTEGRATION DE SYSTEMES DANS UN RESEAU LOCAL D'ORDINATEURS. APPLICATION AU CENTRE DE CALCUL REPARTI

Soutenue à Saint-Etienne le 28 Janvier 1980 devant la commission d'examen :

MM. L. BOLLIET	Président
J.F. CHAMBON S. GUIBOUD-RIBAUD B. LE BIHAN	Examinateurs

ECOLE NATIONALE SUPERIEURE
DES MINES DE SAINT.ETIENNE

INSTITUT NATIONAL POLYTECHNIQUE
DE GRENOBLE

N° d'ordre : 17 IS

THÈSE

présentée par

Jean ANGELIDES

pour obtenir

le grade de Docteur de 3° cycle

"SYSTEMES ET RESEAUX INFORMATIQUES"

INTEGRATION DE SYSTEMES DANS UN RESEAU LOCAL D'ORDINATEURS. APPLICATION AU CENTRE DE CALCUL REPARTI

Soutenue à Saint-Etienne le 28 Janvier 1980 devant la commission d'examen :

MM. L. BOLLIET — Président

J.F. CHAMBON
S. GUIBOUD-RIBAUD
B. LE BIHAN
— Examinateurs

ECOLE NATIONALE SUPERIEURE DES MINES DE SAINT-ETIENNE

Directeur	: M. G.	ARNOUIL
Directeur des Etudes et de la Formation	: M. R.	SOULAT
Directeur des Recherches	: M. Ph.	COUEIGNOUX
Directeur Administratif et Financier	: M. A.	COINDE

PROFESSEURS DE 1ère CATEGORIE

MM.	BOOS	Jean-Yves	Métallurgie
	COINDE	Alexandre	Gestion
	GOUX	Claude	Métallurgie
	LEVY	Jacques	Métallurgie
	RIEU	Jean	Mécanique - Résistance des Matériaux
	SOUSTELLE	Michel	Chimie
	FORMERY	Philippe	Mathématiques Appliquées

PROFESSEURS DE 2ème CATEGORIE

MM.	GUIBOUD-RIBAUD	Serge	Informatique
	LOWYS	Jean-Pierre	Physique
	TOUCHARD	Bernard	Physique Industrielle

DIRECTEUR DE RECHERCHE

M.	LESBATS	Pierre	Métallurgie

MAITRES DE RECHERCHE

MM.	BISCONDI	Michel	Métallurgie
	COUEIGNOUX	Philippe	Informatique
	DAVOINE	Philippe	Géologie
Mle	FOURDEUX	Angeline	Métallurgie
MM.	KOBYLANSKI	André	Métallurgie
	LALAUZE	René	Chimie
	LANCELOT	Francis	Chimie
	LE COZE	Jean	Métallurgie
	MATHON	Albert	Gestion
	PERRIN	Michel	Géologie
	THEVENOT	François	Chimie
	TRAN MINH	Canh	Chimie

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. Philippe TRAYNARD

Année Universitaire 1979-1980

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS TITULAIRES

MM.	BENOIT	Jean	Electronique - Automatique
	BESSON	Jean	Chimie Minérale
	BLOCH	Daniel	Physique du Solide - Cristallographie
	BONNETAIN	Lucien	Génie Chimique
	BONNIER	Etienne	Métallurgie
	*BOUDOURIS	Georges	Electronique - Automatique
	BRISSONNEAU	Pierre	Physique du Solide - Cristallographie
	BUYLE-BODIN	Maurice	Electronique - Automatique
	COUMES	André	Electronique - Automatique
	DURAND	Francis	Métallurgie
	FELICI	Noel	Electronique - Automatique
	FOULARD	Claude	Electronique - Automatique
	LANCIA	Roland	Electronique - Automatique
	LONGEQUEUE	Jean-Pierre	Physique Nucléaire Corpusculaire
	LESPINARD	Georges	Mécanique
	MOREAU	René	Mécanique
	PARIAUD	Jean-Charles	Chimie-Physique
	PAUTHENET	René	Electronique - Automatique
	PERRET	René	Electronique - Automatique
	POLOUJADOFF	Michel	Electronique - Automatique
	ROBERT	André	Chimie Appliquée et des Matériaux
	TRAYNARD	Philippe	Chimie - Physique
	VEILLON	Gérard	Informatique Fondamentale et Appliquée

* en congé pour études.

PROFESSEURS SANS CHAIRE

MM.	BLIMAN	Samuel	Electronique - Automatique
	BOUVARD	Maurice	Génie Mécanique
	COHEN	Joseph	Electronique - Automatique
	GUYOT	Pierre	Métallurgie Physique
	JOUBERT	Jean-Claude	Physique du Solide - Cristallographie
	LACOUME	Jean-Louis	Electronique - Automatique
	ROBERT	François	Analyse Numérique
	SABONNADIERE	Jean-Claude	Electronique - Automatique
	ZADWORNY	François	Electronique - Automatique

MAITRES DE CONFERENCES

MM.	ANCEAU	François	Informatique Fondamentale et Appliquée
	CHARTIER	Germain	Electronique - Automatique
Mme	CHERUY	Arlette	Automatique
MM.	CHIAVERINA	Jean	Biologie, Biochimie, Agronomie
	IVANES	Marcel	Electronique - Automatique
	LESIEUR	Marcel	Mécanique
	MORET	Roger	Physique Nucléaire - corpusculaire
	PIAU	Jean-Michel	Mécanique
	PIERRARD	Jean-Marie	Mécanique
Mme	SAUCIER	Gabrielle	Informatique Fondamentale et Appliquée
M.	SOHM	Jean-Claude	Chimie Physique

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

MM.	FRUCHAT	Robert	Directeur de Recherche
	ANSARA	Ibrahim	Maître de Recherche
	BRONOEL	Guy	Maître de Recherche
	CARRE	René	Maître de Recherche
	DAVID	René	Maître de Recherche
	DRIOLE	Jean	Maître de Recherche
	KLEITZ	Michel	Maître de Recherche
	LANDAU	Ioan-Doré	Maître de Recherche
	MERMET	Jean	Maître de Recherche
	MUNIER	Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (Décision du Conseil Scientifique)

E.N.S.E.E.G.

MM.	BISCONDI	Michel	Ecole des Mines ST ETIENNE (Dépt Métallurgie)
	BOOS	Jean-Yves	Ecole des Mines ST ETIENNE (Métallurgie)
	DRIVER	Julian	Ecole des Mines ST ETIENNE (Métallurgie)
	KOBYLANSKI	André	Ecole des Mines ST ETIENNE (Métallurgie)
	LE COZE	Jean	Ecole des Mines ST ETIENNE (Métallurgie)
	LESBATS	Pierre	Ecole des Mines ST ETIENNE (Métallurgie)
	RIEU	Jean	Ecole des Mines ST ETIENNE (Métallurgie)
	SAINFORT		C.E.N Grenoble (Métallurgie)
	SOUQUET	Jean-Louis	U.S.M.G.
	CAILLET	Marcel	E.N.S.E.E.G. (Chimie Minérale Physique)
	COULON	Michel	E.N.S.E.E.G. (Chimie Minérale Physique)
	GUILHOT	Bernard	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	LALAUZE	René	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	LANCELOT	Francis	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	SARRAZIN	Pierre	E.N.S.E.E.G. (Chimie Minérale Physique)
	SOUSTELLE	Michel	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	THEVENOT	François	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	THOMAS	Gérard	Ecole des Mines ST ETIENNE (Chim.Min.Ph)
	TOUZAIN	Philippe	E.N.S.E.E.G. (Chimie Minérale Physique)
	TRAN MINH	Canh	Ecole des Mines ST ETIENNE (Chim.Min.Ph)

E.N.S.E.E.G.

MM.	BOREL	Joseph	Centre d'Etudes Nucléaires de GRENOBLE
	KAMARINOS	Georges	Centre National Recherche Scientifique

E.N.S.E.G.P.

MM.	BORNARD	Guy	Centre National Recherche Scientifique
	DAVID	René	Centre National Recherche Scientifique
	DESCHIZEAUX	Pierre	Centre National Recherche Scientifique

E.N.S.I.M.A.G.

MM.	COURTIN	Jacques	Université des Sciences Sociales
	LATOMBE	Jean-Claude	Institut National Polytechnique GRENOBLE
	LUCAS	Michel	Université Scientifique et Médicale GRENOBLE.

Je tiens à remercier :

Monsieur Louis BOLLIET, Professeur à l'Université Scientifique et Médicale de Grenoble qui m'a fait l'honneur de présider le jury de cette thèse.

Monsieur Bernard LE BIHAN, Ingénieur à la Régie Renault, qui a accepté de juger ce travail.

Monsieur Serge GUIBOUD-RIBAUD, Directeur du Département Informatique de l'Ecole des Mines de Saint-Etienne, grâce auquel ce projet a pu aboutir.

Monsieur Jean François CHAMBON, Responsable Scientifique à l'Ecole des Mines de Saint-Etienne, qui est l'initiateur de cette étude et qui ne m'a jamais ménagé son aide ni ses conseils.

Les membres de l'Equipe du Centre de Calcul Réparti qui ont contribué directement au projet : Mademoiselle Michèle CART, Monsieur Yves TOSAN, Monsieur Merval JUREMA et Amadou NIANG, ainsi que Paul PAYS, Ingénieur au Centre de Calcul de l'Ecole, pour sa collaboration.

Madame LALLICH qui m'a aidé à établir la bibliographie de cette étude et Madame RETRUS qui a assuré, avec beaucoup de soin et de gentillesse la frappe de cet ouvrage.

SOMMAIRE

CHAPITRE I

ENVIRONNEMENT CENTRE DE CALCUL REPARTI

Après une présentation de l'ensemble du Centre de Calcul Réparti, nous définirons les différentes entités en présence, leurs fonctions principales et la façon dont elles dialoguent. Nous verrons donc comment à partir d'un service local on définit un service réseau et les accès dont un utilisateur dispose pour accéder à des services réseaux. Ensuite, nous examinerons l'ensemble des objectifs définis pour la réalisation d'un Centre de Calcul Réparti (CCR), les contraintes qui nous ont été imposées à cause justement de l'environnement réseau, et les problèmes d'intégration d'un nouvel ordinateur au sein du Centre de Calcul Réparti.

1.1. INTRODUCTION

Le développement très rapide de la technologie électronique en général et des machines de traitement de l'information en particulier ont comme conséquence l'apparition sur le marché de matériels informatiques de plus en plus performants à des prix de moins en moins élevés.

Les considérations économiques à prendre en compte, lors de la production d'un système informatique, étant ainsi bouleversées à leur base, les informaticiens et les utilisateurs sont amenés à changer leur vision du problème.

En effet, la baisse du coût du matériel a comme conséquence l'introduction de l'informatique dans des domaines qui n'avaient pas la possibilité de l'utiliser à cause, justement, du prix assez élevé.

On voit par exemple que certains organismes (entreprises, centres de recherche très spécialisés, etc...) qui, jusqu'alors ne faisaient pas du traitement informatique, du moins pas sur place, sont amenés à créer des moyens informatiques locaux (Centre de Calcul) et souvent très particuliers, adaptés à leurs besoins spécifiques.

L'évolution donc du matériel ainsi observé conduira inévitablement à une évolution du logiciel adequate. Car, il ne faut pas perdre de vue que si le prix du matériel ne fait que baisser, le prix du logiciel lui augmente. Donc, on est amené, de plus en plus, vers le développement des logiciels particuliers et spécifiques (CCR-8) à des applications données qui sont plus simples à concevoir, à mettre en oeuvre et à maintenir et par conséquent, moins coûteux que les logiciels dits "universels" capables de "tout" traiter (souvent d'ailleurs mal).

On arrive donc à penser qu'on devrait affecter à un ordinateur une fonction (voir n si n reste petit) simple, facile à réaliser. Cette idée, bien qu'elle ne soit pas nouvelle, trouve maintenant sa justification économique.

Cependant, il ne faut pas perdre de vue qu'un centre de calcul n'a pas toujours un nombre restreint de fonctions à réaliser. Pour ces centres, deux solutions se présentent (CCR-6) :

(1) La première (elle l'est historiquement) consiste à baser tout autour d'un gros ordinateur qui permet de traiter tous les problèmes, avec tous les inconvénients que cela comporte :

- coût très élevé.
- une panne entraine en général la supression de toutes les fonctions réalisées.
- leurs systèmes (ensemble des outils logiciels mis en oeuvre pour réaliser des fonctions au service des utilisateurs) sont vastes et compliqués.

(2) La seconde consiste à concevoir un centre de calcul décentralisé basé sur un ensemble de mini-ordinateurs spécialisés (RES-1, RES-5). Avec les avantages suivants :

- coût moins élevé (ensemble des mini inférieur au prix d'un gros).
- possibilité de fonctionner en dégradé
- dimension de chaque système réalisé réduite et plus compréhensible, donc plus "humaine".

Ainsi, on voit apparaître la notion d'un centre de calcul basé sur un ensemble de minis spécialisés.

L'idée maintenant serait d'essayer de tirer meilleur profit à l'ensemble en faisant coopérer les différents ordinateurs. On définit ainsi un réseau d'ordinateurs (GEN-10) locaux.

La réalisation de cette coopération nécessite des dialogues entre les différents éléments du réseau et plus précisément le dialogue entre les différents systèmes. Mais ces systèmes, à l'heure actuelle, ne sont pas conçus dans un environnement télé-informatique, ce qui fait que leur ouverture vers le monde extérieur est réduite ou inexistante. Ceci est dû, essentiellement, à des raisons politiques de différents constructeurs, qui, jusqu'à présent, proposaient des solutions de type "monolytiques", c'est-à-dire tout baser autour d'un seul ordinateur dont la configuration dépendait des besoins de leurs clients. Si le problème des coopérations avec un autre ordinateur était parfois envisagé, il s'agissait, presque toujours, d'un ordinateur du même constructeur. Ainsi, les constructeurs concevaient des logiciels de connexion spécifiques et adaptés à leur matériel. Cependant, en ce qui concerne le matériel, il y a certaines normes internationales qui sont, heureusement, très souvent respectées.

On voit donc que dans la plupart des cas le problème d'interconnexion est un problème qui se situe au niveau du logiciel. Pour le résoudre, on est obligé de rajouter au système des éléments nouveaux pour leur apporter une ouverture. Ce travail n'est pas évident à cause de la complexité des systèmes actuels. Cependant, on peut espérer qu'avec le développement actuel de la télé-informatique (GEN-9) et télématique, les constructeurs seront amenés à modifier leur politique : rendre leur système plus ouvert et que surtout des normes internationales portant sur le logiciel de connexion seront définies (RES-16). Le jour d'ailleurs où de telles normes seront définies et appliquées par tout le monde, le problème de communication des systèmes disparaîtra et, en particulier, des études, comme la présente, ne verront plus le jour.

On voit donc une première solution qui consisterait à faire ce que les constructeurs ne veulent pas faire pour le moment : abandonner les différents systèmes de machines que l'on veut connecter et développer un nouveau système qui répondrait aux normes que l'on se serait fixées et qui seraient réalisées pour chaque machine du réseau.

L'inconvénient majeur de cette solution est que cela nécessite un très gros investissement de programmation. Et ce travail recommence chaque fois qu'un nouvel ordinateur est à connecter au réseau (il faut créer son système suivant les normes citées plus haut).

Il paraît donc plus raisonnable d'essayer de définir une architecture fonctionnelle permettant d'intégrer les systèmes existants en les modifiant éventuellement.

L'architecture définie repose sur la notion du FRONTAL: parmi les ordinateurs du réseau, on en distingue un qui a une fonction particulière : il joue le rôle de concentrateur diffuseur des terminaux (local ou distant) et celui de "plaque tournante" c'est-à-dire de relais dans les dialogues entre les différents éléments du réseau, en s'interposant d'une part entre chaque ordinateur local et le réseau, d'autre part, entre chaque couple d'ordinateurs locaux. A cet ordinateur de rôle un peu particulier, sur lequel d'ailleurs la plupart des terminaux du centre son connectés, on a donné le nom de FRONTAL (CCR-3, CCR-4,GEN-4).

Le réseau local a alors une configuration étoilée dont le centre est constitué par le FRONTAL. L'inconvénient majeur de cette solution est un certain manque de fiabilité car une panne du FRONTAL entraine l'arrêt du réseau en tant que tel. Par contre, l'extensibilité du réseau est plus facile (modification localisée au FRONTAL et au nouvel ordinateur).

Quant à l'architecture du FRONTAL, elle est modulaire comprenant plusieurs niveaux hiérarchiques : entrées-sorties, gestion ligne, gestion procédure, gestion protocole, niveau multiplexage/démultiplexage. Ceci afin de réaliser un "calculateur étendu" orienté entrées-sorties qui fournit un surensemble de la machine de base.

Voyons maintenant les différents objectifs et contraintes que l'on s'est fixé pour ne pas bouleverser ceux qui existent (les différents systèmes) et apporter, en plus, l'ouverture.

- Unicité du langage de commande : la définition d'un langage unique paraît nécessaire car sinon la connaissance de tous les langages des différents ordinateurs est indispensable et le réseau équivaut à un ensemble des mini-ordinateurs isolés. La présence d'un tel langage et la conservation de chaque système d'exploitation sur chacun des ordinateurs implique une traduction. Ce langage unique s'appelle LCE (Langage de Commande Externe).

- Flexibilité : introduire de nouvelles machines facilement.

- Fiabilité : limiter les conséquences des pannes et en aucun cas la panne d'un composant ne doit impliquer l'arrêt de l'ensemble.

- Transparence d'utilisation : l'utilisateur doit ignorer la localisation des services.

- Protection : du système vis-à-vis des utilisateurs et des utilisateurs entre eux.

L'objet de cette thèse est de relier logiquement un ordinateur (son système d'exploitation) à un réseau local d'ordinateurs pour permettre :

- l'accès d'une manière banale, par les différents points d'accès du réseau, de toutes les applications que le nouveau système offre.

- l'accès de toutes les applications, par les terminaux locaux du nouvel ordinateur connecté, offertes sur le réseau.

1.2. ENTITES EN PRESENCE

On se contentera d'une description très sommaire de ces entités étant donné qu'elles font l'objet, par ailleurs, d'une étude détaillée (CCR-5).

- L'utilisateur : c'est un être humain qui utilise les différents services du CCR. Il est caractérisé par :

son nom (identification)
son accès : le terminal à partir duquel il travaille.

Quand il veut travailler, il se fait connaître du système (LOGIN) ce qui a comme effet l'association de son nom à son accès. Cette association prend fin à l'émission de la commande LOGOUT. On appelle session toute séquence de l'utilisateur commençant par LOGIN et finissant par LOGOUT.

- L'interpréteur : il interprète le langage de commande (qui est unique) à la disposition des utilisateurs et réalise les actions adéquates.

- Service réseau : voir I.1.5.

- Terminal virtuel (RES-9, RES-14, RES-15) : il permet à tout système du réseau de voir les terminaux d'une manière standard. Chaque système, vis-à-vis du réseau offre le même type d'appareil. La correspondance entre un appareil virtuel et un appareil réel est effectuée par le système lui-même à l'aide de sa gestion d'appareil réel.

- Correspondant : il est divisé en deux. Le correspondant utilisateur (CU) qui se trouve en relation avec les utilisateurs et le correspondant service (CS) qui se trouve en relation avec les services. Cette correspondance, à la fois, avec les utilisateurs et les services lui a valu le nom de correspondant.

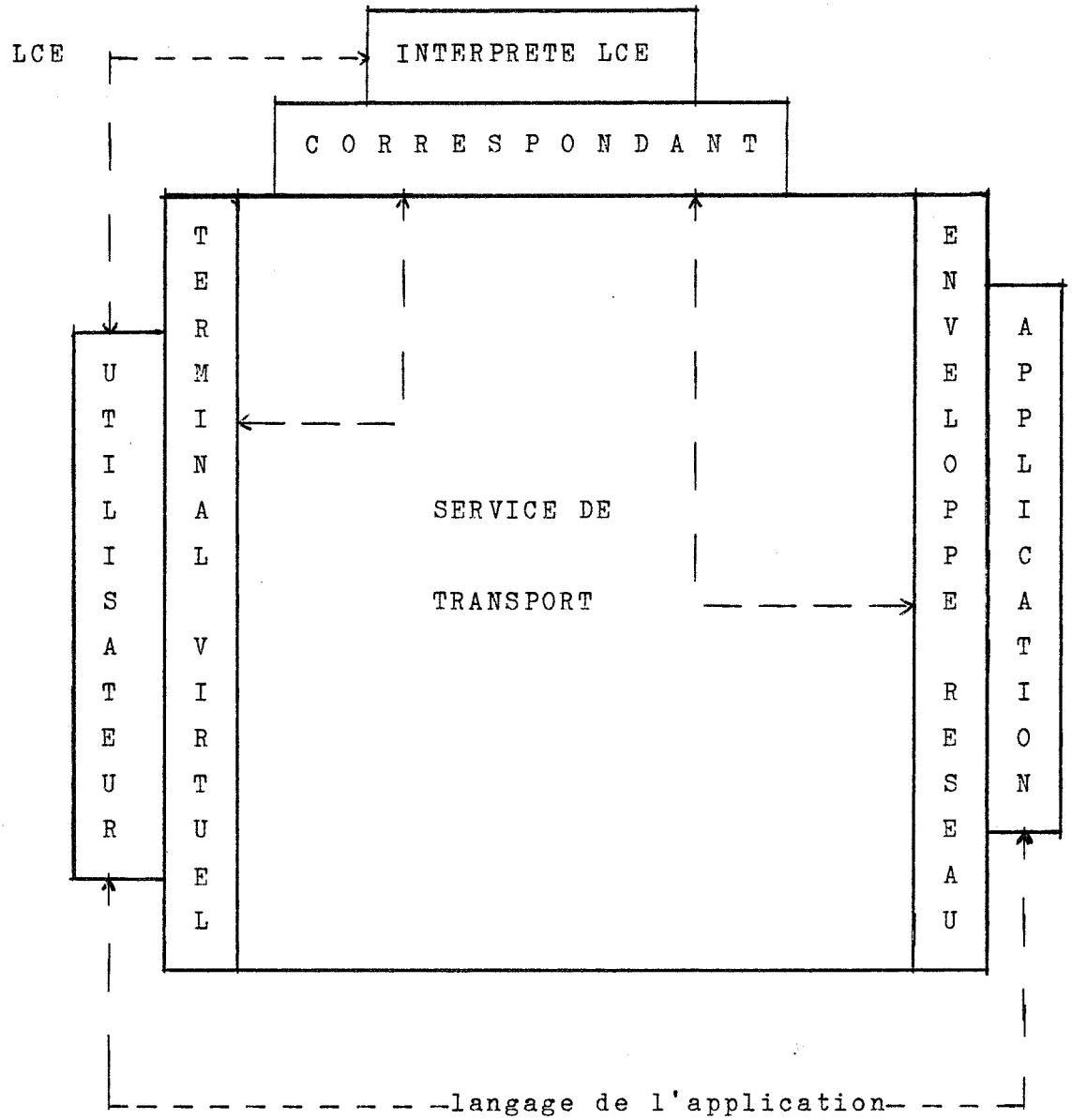

LCE
INTERPRETE LCE
C O R R E S P O N D A N T
UTILISATEUR
TERMINAL VIRTUEL
SERVICE DE
TRANSPORT
ENVELOPPE RESEAU
APPLICATION
langage de l'application

Figure 1.1.

1.3. INTERPRETE-CORRESPONDANT

Interprète : il s'agit de l'interlocuteur des utilisateurs, dans le système CCR. En effet, c'est à lui que les utilisateurs s'adressent par des commandes exprimées en LCE. Ces commandes sont interprétées par lui et les actions qui en découlent sont réalisées par lui également. Il traduit eventuellement le LCE en langage interne (LCI), ce dernier étant le langage vehiculé entre les differents éléments du réseau (voir 3.2.2.). Il connaît tous les utilisateurs potentiels du Centre de Calcul Réparti. A chaque utilisateur il associe un environnement. Toute commande de l'utilisateur est interprétée en fonction de son environnement, ce qui permet la définition d'un certain "pouvoir" pour chaque utilisateur.

En plus de cet environnement relatif aux utilisateurs, l'interprète a besoin des informations relatives au service. En effet, c'est à travers lui qu'un utilisateur demande l'accès à un service ou la destruction de cet accès.

Correspondant : il gère les liaisons entre les utilisateurs et les applications : il les crée ou il les modifie ou encore il les détruit. Il prend en charge des messages en provenance des utilisateurs et à destination de l'interpréteur ou des applications. Inversement, les messages en provenance d'une application sont dirigés par lui, soit vers les utilisateurs soit vers d'autres applications.

1.4. ACCES UTILISATEUR

Un utilisateur voulant travailler sous le CCR se présente devant un terminal. C'est lui qui lui permet d'accéder au système pour envoyer des commandes et recevoir des résultats. On appelle aussi son terminal "accès utilisateur". Il existe une très grande variété de terminaux et tous n'ont pas les mêmes fonctions physiques.

Une homogénisation de ces terminaux paraît nécessaire pour permettre leur vision standardisée aux utilisateurs et aux applications. Cette standardisation a une double signification :

- la banalisation de tous les terminaux (locaux et distants par rapport au CCR)

- la définition d'un terminal virtuel, et faire en sorte que tous les terminaux se présentent en tant que tels.

1.5. SERVICE RESEAU

Un service réseau, en général situé sur une machine hôte, est constitué par trois niveaux (RES-4).

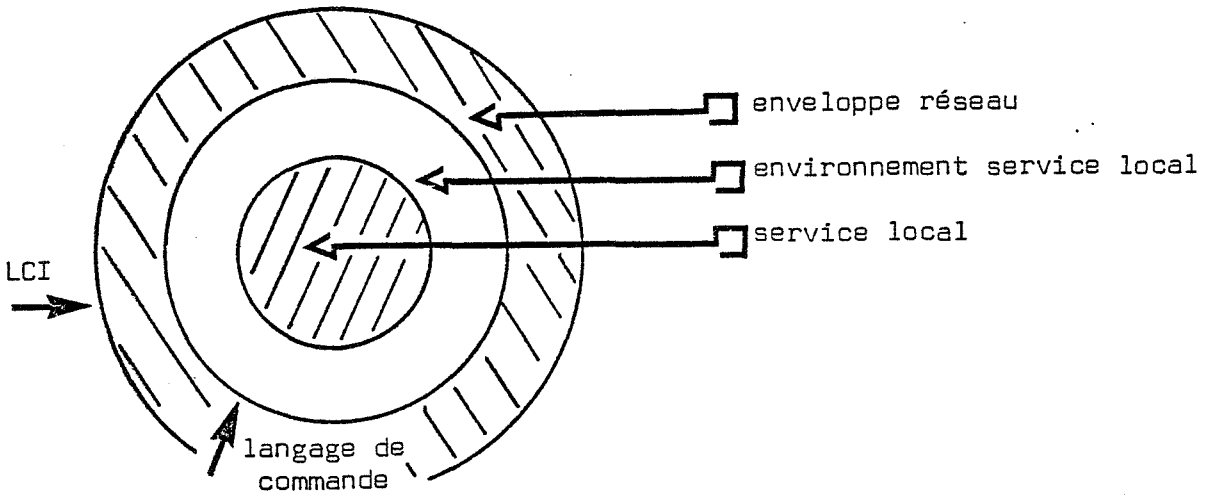


Figure 1.2.

- service local : service offert par la machine hôte (compilation, gestion de fichiers, éditeur de texte...)

- Environnement service local : parties du système d'exploitation de la machine hôte nécessaires au service local (OS). Ce niveau reçoit des commandes dans le lange de l'OS. Dans certains cas, il peut ne pas exister.

- Environnement réseau : Il s'agit du niveau qui reçoit des commandes en LCI, les interprète et effectue la correspondance avec le langage de commande (JCL) propre au système d'exploitation.

Un service réseau peut être réparti sur plusieurs sites (on entend par site tout ordinateur connecté ou réseau externe accessible) mais il n'est connu du CCR que sur un site donné. C'est le composant du service sur ce site qui est appelé service réseau et est maître des actions des différentes parties du service. Les éléments répartis du service sont reliés entre eux soit par des lignes spécialisées, soit par l'intermédiaire de points d'entrée du correspondant service. Le maître présente la même structure que celle décrite plus haut. Ce n'est pas le cas des autres parties s'il y a des lignes spécialisées, mais si on passe par des points d'entrée du correspondant il faut qu'elles présentent le même aspect qu'un service réseau même si en fait elles ne sont pas accessibles en tant que tel.

Un exemple d'un tel service réparti est le service gestion de fichiers (CCR-9). Chaque partie du service contient la gestion de fichiers fournie par le système d'exploitation du site. Le maître tient à jour des tables de localisation des fichiers. Dans le cas d'un transfert de fichier d'un site à un autre, le maître prévient la (ou les) partie(s) concernée(s) et donne au site émetteur le nom réseau du site récepteur. S'il n'existe pas de ligne spécialisée entre les deux sites en jeu, le maître fournit en plus au site émetteur le point d'entrée du correspondant identifiant le destinataire.

Dans la suite, nous ne considérons plus que les services réseaux que nous appelerons simplement services.

1.6. ARCHITECTURE FONCTIONNELLE ET RELATIONS ENTRE ENTITES

Pour réaliser le FRONTAL un système spécialisé a été conçu et réalisé sur le T1600. Ce système "orienté entrées-sorties" réalise à partir du calculateur de base un "calculateur étendu" qui fournit un sur-ensemble des possibilités du T1600. Ce "calculateur étendu" offre les fonctions du calculateur de base et, par logiciel, des fonctions plus évoluées :

- gestion des tâches et des interruptions
- gestion dynamique de la mémoire principale
- gestion d'une mémoire auxiliaire sur disque
- gestion des files d'attente
- gestion de réveils
- fonctions élémentaires diverses (décalage, positionnement de bits dans une file, etc...)

Ces fonctions constituent des outils pour la programmation du FRONTAL.

L'architecture logique du FRONTAL repose sur le concept de machine.

Une machine est un ensemble fonctionnel spécialisé dans l'exécution d'une fonction (plus ou moins complexe). Toute machine se situe à un niveau hiérarchique donné. Elle reçoit des commandes des machines de niveau supérieur et leur signale des évènements. Pour exécuter ces commandes, elle peut elle-même adresser des commandes aux machines de niveau inférieur dont elle est également susceptible de recevoir des évènements. Enfin, elle est de nature séquentielle et possède un état courant qui évolue depuis un état initial.

L'élément essentiel du FRONTAL est la machine d'entrées-sorties, elle-même décomposée en machines plus élémentaires.

Les différents niveaux reconnus sont :

- niveau gestion du bus d'entrée et des interruptions (M : IT).

- niveau ligne : ensemble de machines (M : TR) qui gèrent chacune un type de coupleur d'entrée-sortie.

- niveau procédure : ensemble de machines (M : GT) qui gèrent les procédures de transmission.

- niveau protocole : ensemble de machines (M : GP) gérant chacune un protocole de communication

- niveau boîte aux lettres (M : DM) qui assure un multiplexage / démultiplexage (CCR-1, CCR-2).

A l'intérieur du CCR, on distingue trois types de dialogues :

- dialogue utilisateur interprète : il s'agit de requêtes en LCE, moyennant lesquelles un utilisateur fait connaître ses désirs ou impose ses volontés. On les appelle commandes.

- dialogue utilisateur application : il s'agit des interactions en langage de l'application (par exemple APL, langage PL, etc...). On les appelle échanges.

- dialogue application application : se fait en langage interne, prédéfini et connu des deux applications, et sert à la coopération des applications.

1.7. OBJECTIFS ET CONTRAINTES GENERAUX

La réalisation d'un centre de calcul réparti fondé sur un réseau de mini-ordinateurs spécialisés dans une fonction, c'est un projet qui a débuté il y a trois ans.

L'objectif est de fournir un ensemble de services (APL, FORTRAN, différents compilateurs, éditeurs, etc...) et non pas seulement un ensemble d'ordinateurs comme c'est souvent le cas avec des minis, accessibles par un langage de commande unique depuis une console, un ordinateur ou le monde extérieur.

A son démarrage, le projet reposait déjà sur des bases solides. En effet, un système d'exploitation NOS (Network Operating System) a été conçu et réalisé sur un T1600 à l'Ecole des Mines pour faciliter les échanges dans un environnement téléinformatique et permettre de résoudre facilement les différents problèmes de connexion de terminaux d'ordinateurs et de raccordement à un réseau.

Voyons maintenant un peu plus en détail les objectifs (et les contraintes qui en découlent) que l'on s'est fixés au départ et qui ont motivé nos choix faits pour résoudre le problème posé.

Flexibilité : on a voulu un réseau (par réseau dans la suite, on entendra réseau local sauf précision contraire) extensible (GEN-7), autrement dit l'introduction d'une ou plusieurs nouvelles machines doit être possible et ceci facilement.

Il y a plusieurs raisons à cela : d'abord des raisons économiques. L'adjonction facile de nouvelles machines permet un investissement progressif. On achète des ordinateurs un par un et on les ajoute au réseau. On n'est pas donc obligé de tout remettre en cause chaque fois que de nouvelles charges importantes apparaissent.

Si on ne peut pas les traiter avec ce qui existe sur place, ou éventuellement le sous-traiter à l'extérieur, à l'aide d'un réseau externe (si ceci n'est pas économiquement viable), alors, on envisage l'extension du centre par l'adjonction d'une nouvelle machine.

Ensuite, des raisons politiques : le matériel peut être hétérogène et provenir d'origines diverses, on n'est donc plus sous la dépendance d'un seul constructeur pour son équipement. Cependant, un inconvénient à ce propos c'est que les différents constructeurs , à l'heure actuelle, n'ont pas fait d'efforts majeurs pour rendre leurs machines "ouvertes" vers le monde extérieur si ce monde extérieur n'est pas constitué de matériel de leur provenance.

L'intégration donc d'un nouveau système doit se faire facilement sans bouleverser la logique du C.C.R. et en particulier sans arrêter l'exploitation déjà existante. Les nouveaux services donc peuvent être mis en exploitation progressivement au fur et à mesure qu'ils deviennent opérationnels sans supprimer temporairement un des services déjà existants.

Du côté du nouveau venu, on cherchera à intégrer le logiciel de connexion le plus indépendant possible de son système de façon à ce que sa logique ne soit pas affectée. On cherchera donc à l'intégrer le plus souvent sous forme d'une tâche d'application.

Fiabilité : il ne faut pas qu'une panne éventuelle d'un des ordinateurs affecte le réseau entier. Certains services si une telle panne subsiste, peuvent être temporairement supprimés, ou traités sur une autre machine que la machine où on les traite habituellement, au détriment peut être de leur performance, mais le reste doit fonctionner sans problème.

En particulier, il est indispensable qu'en cas de panne d'un ordinateur participant, chacun des autres puisse travailler au moins en local avec ses terminaux locaux et avec le même système d'exploitation qu'auparavant en rendant ineffectif le logiciel de connexion (c'est le cas si le FRONTAL tombe en panne). Pour cela il faudrait, qu'en cas de panne, ou incident du CCR, en cours de session, le système d'un ordinateur participant puisse se récupérer c'est-à-dire continue à fonctionner en local sans être réinitialisé. Evidemment, certains de ces terminaux distants peuvent devenir ineffectifs.

Il ne faudrait pas aussi négliger le problème des reprises une fois l'incident réparé. Il faut avoir la possibilité de réintégrer un système initialement connecté et déconnecté ensuite à cause de l'incident. La transportabilité du FRONTAL est également envisagée.

Rentabilité : On a vu que la fondation d'un centre autour des minis est économiquement viable. Cependant, on ne se limitera pas obligatoirement sur des minis, l'intégration d'un gros ordinateur reste toujours possible. D'autant plus, qu'il ne faut pas perdre de vue qu'un réseau de minis n'a pas toutes les performances d'un gros ordinateur pour certaines applications.

Donc, d'une part, les gros ordinateurs ne sont pas condamnés et d'autre part, il faut toujours envisager la possibilité d'intégrer une application qui nécessite, par ses performances, la présence d'un gros, si la présence de cette application, à notre centre, se justifiait économiquement.

Transparence d'utilisation : Un utilisateur doit ignorer la façon dont est réalisé un service c'est-à-dire sa localisation qui peut varier dans le temps. En effet, on a vu que des transferts de charges doivent être possibles en cas de panne par exemple.

Le même mécanisme peut donc être utilisé si la charge globale d'une machine devient trop importante. On décide de transférer un ou plusieurs services dans la mesure du possible sur une autre machine temporairement.

L'utilisateur ne doit pas faire la différence sauf peut être la constatation d'une légère baisse de performance due au fonctionnement temporaire du service en dégradé.

Protection : (GEN-6) il faut définir un système de désignation des services et ressources permettant la protection des utilisateurs entre eux et vis-à-vis des services indépendamment des machines du réseau et de leurs systèmes d'exploitation respectifs.

Ce système de protection doit donc fournir une sûreté de fonctionnement à tous les utilisateurs.

Unicité du langage de commande : un utilisateur veut en général utiliser un service donné. Pour accéder à ce service, qui se trouve sur une des machines du réseau à un instant donné, doit disposer d'un langage qui est unique pour l'ensemble des services. Ceci afin de ne pas connaître le JCL de chaque ordinateur du réseau. Par ailleurs, on souhaite conserver le système d'exploitation pour chaque ordinateur, ce qui implique une traduction à un instant donné (LAN-1).

En effet, ce n'est pas un surensemble des langages des différents systèmes d'exploitation mais un langage indépendant ayant sa propre syntaxe et sa propre sémantique.

En fait, c'est à travers lui, par sa connaissance, que l'utilisateur "voit" le système, ainsi, par une bonne définition du langage, on ne fournit à l'utilisateur que les services et les fonctions qui lui sont indispensables, l'ensemble des systèmes sous-jacents nécessaires à la bonne gestion des services lui est alors transparent.

Pour faciliter la lecture et l'écriture des commandes au système, il semble préférable de donner à ce langage une structure plus évoluée que celle des JCL existant actuellement; par exemple, en écrivant les demandes de service sous forme d'appels procéduraux avec comme paramètres les fichiers à traiter ou le résultat du travail d'un autre service, ce qui évite d'avoir à écrire un trop grand nombre de commandes et fait apparaître plus clairement la suite des "steps" demandés ainsi que ce à quoi ils se rapportent.

1.8 INTEGRATION D'UNE NOUVELLE MACHINE

Intégrer un nouvel ordinateur veut dire le faire coopérer avec les autres. Mais pour que des entités coopèrent, il faut qu'ils aient les moyens de communiquer, c'est-à-dire : qu'ils se connaissent (chacun connaît l'existence de l'autre) et qu'ils ont en plus un support de communication, autrement dit, il existe un "lien" entre eux et des moyens de se désigner les uns les autres s'ils sont plus nombreux que deux. Attention communiquer ne veut pas dire ici obligatoirement se comprendre !

Les principes nécessaires pour la communication évoquée plus haut qui sont évidemment valables pour les humains, (prise de conscience de l'existence de l'autre par la vue en général, support de communication : la voix, désignation : en général par un nom) sans qu'ils ne soient obligatoirement conscients, sont valables pour les ordinateurs.

Examinons maintenant quelques exemples de façon à mettre en communication plusieurs personnes :

(1) Chacun connaît tous les autres interlocuteurs et peut leur adresser la parole. Mis à part le problème de synchronisation (une personne ne comprend pas en général si deux ou plusieurs personnes lui parlent en même temps) c'est la façon la plus simple et la plus fiable de communiquer.

(2) Une autre façon c'est de s'adresser toujours à quelqu'un (nommé transmetteur) qui a été désigné d'office, et qui est connu de tout le monde, en lui demandant de transmettre vers l'un des interlocuteurs.

(3) une variante de la façon précédente : le transmetteur existe toujours mais l'existence des quelques liens particuliers, c'est-à-dire la possibilité de communiquer avec quelqu'un sans faire appel au transmetteur existe.

Si maintenant à la place des hommes on met des machines, en considérant évidemment comme "lien" une liaison physique entre deux d'entre elles, on constate que la première solution, bien que paraissant meilleure, du point de vue fiabilité, est rarement utilisée si le nombre d'ordinateurs est grand (supérieur à 3). On préfère donc une solution du type (2)ou (3).

Dans ce cas, l'introduction d'une nouvelle machine dans un réseau d'ordinateurs locaux existants, se résume, en général, à la connexion de cette machine au "transmetteur".

Ceci concerne bien évidemment les connexions physiques Rien ne nous empêche de définir ensuite des connexions logiques (différentes des connexions physiques) entre tous ou une partie des ordinateurs. Le problème est d'appliquer ensuite le graphe de connexion logique au graphe de connexion physique existant. Enfin, une connexion logique entre deux ordinateurs établie, on essaie de faire dialoguer les deux systèmes. (A ce niveau, il s'agit non seulement de communiquer mais de se comprendre en plus).

Le rôle du "transmetteur", en ce qui concerne le CCR, est évidemment joué par le FRONTAL.

Intégrer donc un nouvel ordinateur au sein du C.C.R. nécessite le développement sur le FRONTAL d'un logiciel de connexion qui gère la ligne réalisant la connexion physique. Ce logiciel est un ensemble en couches qui assure : la gestion du coupleur au bout de la ligne, la gestion d'une procédure de transmission et la gestion d'un protocole.

D'une manière générale, la gestion du coupleur existe, dans la mesure où la gestion d'un coupleur semblable existe déjà sur le FRONTAL. Il en est de même en ce qui concerne la gestion d'une procédure sauf dans le cas où le nouveau système ne saurait gérer qu'une procèdure particulière de type non standard.

Enfin, quant à la gestion du protocole, soit le nouveau possède la gestion d'un tel protocole auquel cas on a intérêt de développer la gestion du même protocole sur le FRONTAL, soit il n'en possède pas une, auquel cas on intègre dans son système la gestion d'un protocole dont les spécifications ont été réalisées par nous-mêmes et dont on possède une gestion convenable sur le FRONTAL.

De cette manière, pour réaliser la connexion en ce qui concerne la gestion d'un protocole, on la réalisera soit sur le FRONTAL soit sur le nouvel ordinateur.

CHAPITRE II

METHODOLOGIE DE CONNEXION D'UN SYSTEME

Dans le chapitre précédent on a vu l'intérêt d'un Centre de Calcul Réparti décentralisé et la nécessité de pouvoir connecter facilement un nouvel ordinateur. Ici, on verra comment une telle connexion pourra se faire en suivant un certain nombre des règles. Ces règles nous permettent de définir une certaine méthodologie qui, contrairement aux apparences, existe.

Dans un premier temps, on passera en revue les différentes possibilités qui nous sont offertes habituellement par les systèmes d'exploitation fournis par les constructeurs. Ensuite, on verra ce dont on a besoin en plus et des façons simples de le réaliser. On insistera sur le fait que la réalisation d'une connexion se fait dans un contexte d'exploitation et qu'il n'est pas question de stopper l'exploitation de l'ensemble du centre de calcul pour de longues périodes. La mise au point donc doit se faire en grande partie pendant cette exploitation. Pour ce faire, on verra la réalisation des outils nécessaires. D'autre part, on cherchera à minimiser l'investissement en logiciel global en portant nos efforts soit du côté C.C.R. soit du côté du nouvel ordinateur. Le système d'exploitation de l'ordinateur connecté étant plus ou moins ouvert sur le monde extérieur, on explicitera les fonctions fondamentales et les niveaux qu'on sera amené à rajouter afin qu'il s'intègre à l'ensemble.

Un autre point important dans ce chapitre est de voir les avantages et inconvénients de la connexion d'un système par rapport à la connexion d'une application.

2.1. NATURE DU PROBLEME

On a vu au premier chapitre que pour avoir une évolution aisée de notre Centre de Calcul Réparti, la connexion facile de nouveaux ordinateurs s'impose. Pour qu'une telle connexion soit en effet facile, il faudrait qu'elle puisse se faire d'une façon presque automatique afin que l'on n'ait pas à reconsidérer le problème à chaque fois qu'un nouvel ordinateur se présente et que l'expérience acquise lors de la première connexion soit bénéfique en grande partie pour les suivantes. D'où la nécessité de pouvoir définir une certaine méthodologie pour procéder à une connexion c'est-à-dire une liste des actions à faire lorsqu'une telle connexion est à réaliser. Bien entendu, suivant le système à connecter, elle sera plus ou moins aisée mais en grande partie la méthode sera la même pour la plupart des systèmes.

Pour rendre systématique la démarche à entreprendre pour effectuer une connexion, on a besoin d'un certain nombre de facilités, c'est-à-dire des outils logiciels. Un certain nombre de ces outils sont fournis en général par le système du constructeur. Pour les autres, on sera amenés à les réaliser. Pour cela, la première chose à faire sera d'examiner de près le système de l'ordinateur à connecter, de bien déterminer quelles fonctions nous intéresse et ensuite, voir de quelle manière on peut réaliser ce qui nous manque. Notre souci principal restera toujours le fait de modifier le moins possible le système, pour rester ouvert à d'éventuelles versions ultérieures, et de bien localiser ces modifications le cas échéant. On pourra, dans certains cas, par exemple, inclure le logiciel de connexion sous forme de programme utilisateur (meilleure solution car on ne touche pas du tout au système) ou des tâches d'application si le système nous permet la définition de telles tâches. Ceci nous garantit une indépendance complète entre le système et le logiciel de connexion, en particulier, en ce qui concerne la gestion du protocole (RES-16).

D'autre part, presque tous les systèmes modernes sont réalisés par un logiciel en "couche". La couche du niveau le plus haut c'est celle qu'un utilisateur voit en général et celle le plus bas est la plus "proche" au matériel et traite les entrées-sorties.

Entre chaque niveau il y a les interfaces bien définies et souvent standardisées s'il s'agit des systèmes du même constructeur. En effet, le constructeur se laisse toujours la possibilité de rajouter un logiciel de gestion d'un périphérique particulier et maintient stables les interfaces à ce niveau.

Notre effort doit se porter à intégrer le logiciel de connexion dans un niveau donné (voire n si n reste petit) et de façon à ce que les interfaces de ce niveau n'en soit pas affectées. Parfois, nous pouvons être amenés à rajouter un ou plusieurs niveaux entiers entre deux niveaux du système en respectant si possible les interfaces.

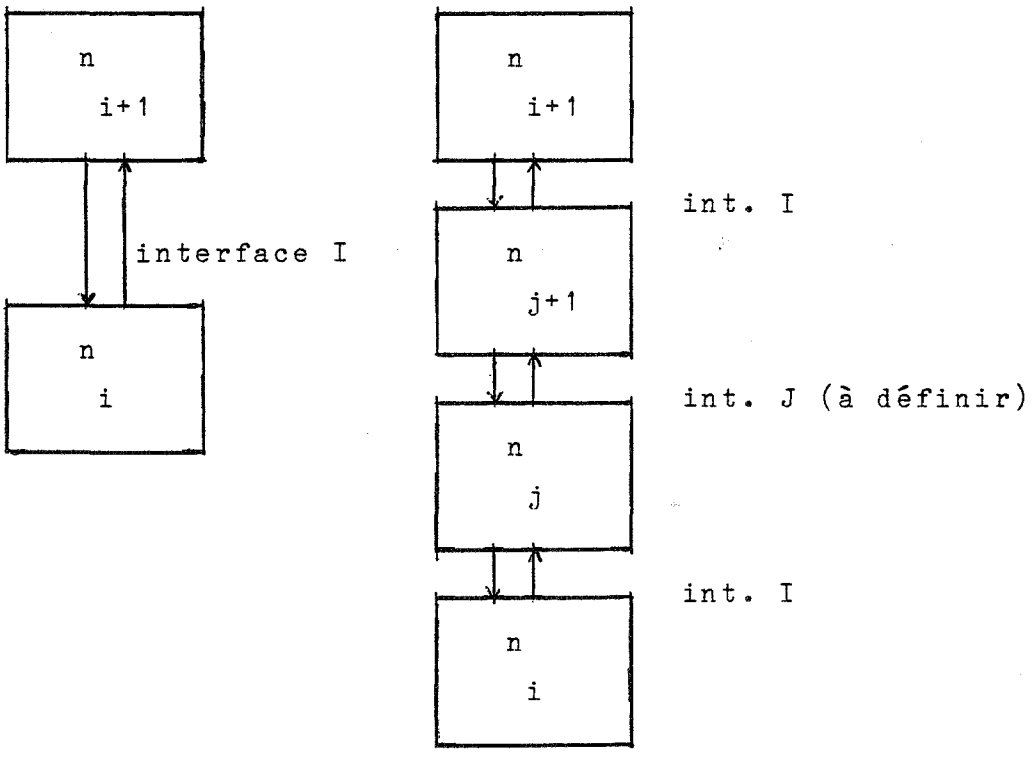


Figure 2.1.

La modification, dans un cas comme celui-là, d'une interface est une opération très délicate et il faut opérer avec beaucoup de prudence et tout faire en connaissance de cause car la complexité des systèmes actuels fait que des "effets de bord" sont souvent imprévisibles.

Tout ce que l'on vient de voir est valable dans le cas où le système à connecter ne possède pas une gestion convenable d'un protocole donné. Dans le cas contraire, on a tout intérêt à rajouter la gestion de ce protocole côté C.C.R. Plusieurs raisons à cela :

- D'abord on n'a pas à "toucher" le nouveau système.

- Ensuite, l'architecture du logiciel du CCR permet une réalisation plus facile et une mise à jour plus aisée grâce à certains outils de mise au point sophistiqués dont on dispose (trace par exemple).

- Enfin, l'investissement peut être très rentable dans la mesure où un autre ordinateur à connecter se présente, qui possède une gestion du même protocole que le précédent. Dans ce cas, la connexion peut s'effectuer sans frais.

Notons quand même que dans la présente étude, on est intéressés surtout par le premier aspect du problème, c'est-à-dire que l'ordinateur à connecter ne possède pas une gestion de protocole convenable car l'intégration de la gestion d'un nouveau protocole côté C.C.R. peut être faite suivant les mêmes principes que l'intégration du protocole ordinateur connecté qui a été réalisée et dont on parle au paragraphe 3.1.2.

Mais revenons maintenant sur les outils logiciels dont on a besoin pour réaliser une connexion et que l'on peut résumer ainsi :

(1) - Gestion des entrées-sorties : la liaison étant assurée matériellement par une ligne, sa gestion doit être assurée.

(2) - Gestion des processus : la présence de certains évènements asynchrones et la nécessité de leur traitement en pseudo-parallélisme, nous amènent à penser qu'une machine à processus est mieux adaptée à ce genre de connexion. Dans le cas contraire, on peut toujours par logiciel faire en sorte que la machine apparaisse comme si elle était à processus mais cela est compliqué et introduit souvent un "overhead" important. Un exemple cependant de la possibilité de cette solution dans un environnement télé-informatique réside dans la réalisation d'un superviseur de synchronisation, SUSY, qui permet le partage d'un niveau d'interruption du MITRA-15 entre plusieurs processus. Le MITRA-15 est utilisé comme FRONTAL pour la réalisation de la connexion d'un ordinateur XDS 94 de TELESYSTEMES au réseau R C P (RES-6).

(3) - Gestion mémoire : nécessité d'un certain nombre de tampons d'entrées-sorties.

(4) - Gestion du temps : possibilité d'enclencher des évènements asynchrones à volonté sur une base temporelle.

(5) - Utilitaires : il s'agit des outils rendant la programmation modulaire et la mise au point plus agréable.

Ces outils peuvent être offerts en partie ou en totalité par le système. Au paragraphe suivant, le lecteur trouvera une description, et une justification plus complète de la nécessité, de chacun de ces cinq outils, tandis qu'au paragraphe 2.2.2. il trouvera quelques idées sur leur réalisation, le cas échéant.

2.2. POSSIBILITES OFFERTES PAR UN SYSTEME

2.2.1. CE QU'ON TROUVE EN GENERAL

C'est en grande partie fonction du système en question. Cependant, il existe un minimum d'outils qu'on trouve dans presque tous les systèmes, on verra en quoi cela consiste et comment on peut s'en servir.

Gestion des entrées sorties : tous les systèmes possèdant un moniteur d'entrées-sorties qui, à l'aide des programmes spécifiques de gestion des entrées-sorties appelés DRIVERS, sache gérer les différentes périphériques et lignes. Notons qu'en général les DRIVERS travaillent au niveau de l'octet (voire du bloc) c'est-à-dire qu'ils savent comment faire pour envoyer ou recevoir des octets (ou des blocs). Par ailleurs, les erreurs dues aux lignes de transmission ne sont pas obligatoirement résolues à leur niveau. Dans ce cas, il faut résoudre ces problèmes à un niveau supérieur par la gestion d'une procédure (GEN-1) de transmission et d'un protocole.

La connexion de deux ordinateurs se faisant à l'aide d'une ligne de transmission, il faut d'abord un logiciel qui sache gérer cette ligne, c'est-à-dire qui sache gérer le coupleur au bout de cette ligne. En principe ce logiciel est fourni par le constructeur. Notons au passage qu'il ne s'agit pas obligatoirement de logiciel mais une telle gestion peut être assurée matériellement par un coupleur intelligent. Cependant, la chose essentielle, en ce qui nous concerne est que cette gestion (logicielle ou matérielle) est fournie par le constructeur et qu'elle constitue un outil indispensable pour notre réalisation.

Gestion des processus : il s'agit d'une gestion dynamique de processus disponibles mais comme dans la plupart des systèmes cette gestion n'est pas offerte, on sera amené à la réaliser. On verra comment au paragraphe suivant.

Gestion mémoire : Les liaisons étant effectuées d'une façon dynamique (ouverture, fermeture) les contextes de liaison (mémoire nécessaire pour stocker des informations relatives à une liaison : file de lettres en émission, file de lettres en réception, crédits etc...) peuvent être acquis dynamiquement. En plus, les tampons utilisateurs se voient agrandis d'un entête servant à la gestion du protocole. Ces tampons sont des tampons d'entrées-sorties donc il faut qu'ils se trouvent en mémoire centrale, et leurs existences se limitent en général le temps de l'entrée-sortie. Donc on voit que leur réservation d'une façon statique, bien que possible, n'est pas souhaitable si on veut une utilisation de la mémoire centrale optimale.

Beaucoup de systèmes possèdent une gestion dynamique de la mémoire centrale. En général, l'allocation se fait par bloc dont la taille dépend du système en question et on pourra s'en servir sans la modifier si la longueur d'un bloc élémentaire (de taille minimale) allouable est de l'ordre de la longueur des tampons cités ci-dessus.

Gestion du temps : tous les systèmes en possèdent une, ne serait-ce que pour avoir la date et l'heure. Ceci est réalisé grâce à une comptabilité du temps. Cependant, tous les systèmes n'éprouvent pas le besoin de fournir à un processeur utilisateur la possibilité d'enclencher des évènements asynchrones et de ce fait, bien qu'une comptabilité du temps existe, ils ne fournissent pas des outils pour réaliser des REVEILS.

Qu'est-ce qu'un REVEIL ? La possibilité d'avertir le système que l'on désire être prévenu quand un délai d'une durée spécifiée s'est écoulée.

Pour la réalisation du logiciel de connexion, l'existence de tels REVEILS est un outil indispensable car c'est la seule manière d'enclencher des évènements asynchrones du style : renvoi d'une lettre (car elle n'a pas été correctement transmise et par conséquent elle n'a pas été acquittée). D'un point de vue plus général pour ne pas que l'automate reste indéfiniment dans un état intermédiaire.

Utilitaires : enfin, comme facilités offertes par le système, on peut citer l'existence de certaines procédures de traitement de files de bits, de conversion de caractères, de mise en file d'un élément ou son retrait, de réalisation de piles, etc...

Leur utilisation, par le programme écrit, pour la réalisation d'une connexion évite la duplication du code, mais on peut le faire à condition que :

le logiciel de connexion soit réalisé en tant que partie intégrante du système, c'est-à-dire que le programme qui réalise la connexion, a subi la même édition de liens avec le reste des modules du système donc l'intégration du logiciel de connexion en tant qu'application est à exclure dans ce cas là (sauf si les procédures en question sont offertes en tant que primitives par le système).

Un autre inconvénient d'utiliser des procédures internes au système c'est que leur mise à jour, pour sa version ultérieure, risque de modifier leurs spécifications.

Il faut aussi utiliser, le plus possible, si elles existent, toutes les facilités de mise au point de programmes, offerts par le système, telles que DUMP, PATCH, TRACE, etc...

2.2.2. CE QU'ON EST AMENE A REALISER

Une partie ou même la totalité des fonctions que l'on vient de voir peut ne pas exister et que dans ce cas on est obligé de la réaliser.

Gestion de processus : Certains systèmes sont constitués de tâches ou de processus se déroulant en pseudo parallélisme sans pour autant qu'il existe une gestion dynamique de ces processus. Car, dans ces systèmes le nombre ainsi que la place des processus sont définis statiquement à la génération et on se contente de les initialiser au fur et à mesure des besoins.

On peut, bien entendu, introduire les processus nécessaires à la réalisation de la connexion de la même manière. Mais on perdrait en efficacité dans le sens où la réservation des contextes de ces processus deviendrait statique (donc perte de place mémoire si la liaison n'est pas en fonctionnement) et perte d'un certain aspect "dynamique" lors des initialisations et de l'arrêt du logiciel de la connexion.

Pour conserver cet aspect, il faut donc réaliser une gestion dynamique de processus. Pour ce faire, on utilisera les processus non utilisés par le système, donc disponibles, qui constituent un ensemble que l'on peut appeler un "pool de processus". Pour créer un processus, on choisit une priorité parmi celles des processus du pool, on initialise son contexte et on le lance. La mort d'un processus est effectuée par suppression de ses activités et libération de son contexte.

Pour ce faire, soit on peut admettre qu'un processus peut en tuer un autre avec toutes les difficultés que cela comporte notamment la libération de toutes les ressources qu'il utilise, soit qu'un processus peut signaler à un autre de se tuer.

Gestion mémoire : en ce qui concerne la gestion dynamique de la mémoire centrale, si elle n'est pas réalisée par le système ou si elle est mal adaptée, il y a dans la littérature beaucoup d'algorithmes où le lecteur peut se référer et dont la présentation ici n'est pas d'un grand intérêt (GEN-5). Disons tout simplement à titre d'exemple que pour notre application (connexion du SOLAR au CCR voir CH. IV) à l'absence d'une gestion mémoire adaptée, on a pris un algorithme qui gère des blocs de taille fixe (16 mots) à l'aide d'une MAP (image d'état d'occupation pour chaque bloc mémoire).

Gestion d'entrées-sorties : de la même manière, la gestion d'un coupleur de ligne, si le logiciel ou l'équivalent en matériel n'est pas fourni par le constructeur, est un problème spécifique qui ne sera pas traité ici: ces deux points (entrées-sorties, gestion mémoire) peuvent se résoudre assez simplement sans rentrer dans les logiciels internes du système.

Gestion du temps : elle est réalisée par un dispositif matériel qui provoque des interruptions, (c'est l'horloge temps réel), avec une certaine fréquence. Etant donné que le temps écoulé entre deux interruptions est constant, on peut ensuite comptabiliser le temps en comptabilisant le nombre des interruptions. Ceci est fait le plus souvent à l'aide d'un processus qui est activé chaque fois qu'une interruption d'horloge survient. Une première idée serait de se servir de ce processus pour réaliser des REVEILS. Cependant, ceci est à éviter dans la mesure du possible car l'introduction de traitements supplémentaires à ce niveau risque de modifier le temps élémentaire de traitement de ce processus. Le système peut se trouver perturbé si, lors de sa conception, on a tenu compte de ce temps élémentaire de traitement.

Utilitaires : voyons maintenant certaines facilités de la mise au point qu'on a intérêt à réaliser. Pour cela, après avoir examiné un utilitaire indispensable, on prendra quelques exemples car suivant le système et leur mode de fonctionnement, certains de ces outils peuvent exister totalement ou partiellement et les outils à réaliser sont donc très dépendants du système.

Cas de trace : il est indispensable, si l'on veut déverminer rapidement une connexion, de pouvoir tracer tout ce qu'on envoie et qu'on reçoit sur la ligne aussi bien au niveau de l'octet qu'au niveau du message (au sens d'une procédure et d'un protocole).

En fait, pour un niveau hiérarchique donné, on a intérêt à tracer tout ce qui lui est passé par le niveau supérieur et tout ce qui lui est passé par le niveau inférieur. Evidemment, on n'a pas intérêt à faire marcher toutes les traces en même temps. Surtout à cause du fonctionnement qui n'est plus du temps réel et en plus parce que cela fait souvent trop d'informations à exploiter.

On peut, par exemple, au début, pendant la mise au point de la liaison "physique" (voir paragraphe suivant) utiliser la trace au niveau de l'octet, et une fois que l'on est sûr du fonctionnement de la liaison, utiliser la trace au niveau procédure et protocole.

Utilitaires disponibles par le système RTX 25 (RES-17) qui permet l'accès à partir de terminaux reliés à TRANSPAC d'utiliser les services d'un IBM 370 via un FRONTAL:

- Visualisation mémoire
- Visualisation des paquets erronés
- Visualisation de l'état de la ligne de communication
- Visualisation de la configuration réseau
- Modification des zones mémoire.

Toutes ces informations sont exploitées à l'aide d'une console opérateur dite "réseau" qui est reliée au FRONTAL.

Utilitaire réalisé pour la connexion d'un ordinateur XDS 94 à RCP (RES-6). Il s'agit d'un ensemble d'utilitaires (TALYSMAN) de gestion des files d'attente tampons circulaires et listes de tampons. En plus, quelques utilitaires de tests :

- Simulateur d'évènements
- Générateur de données
- etc...

En conclusion, on constate que la frontière entre les possibilités offertes par un système et celle à réaliser par le concepteur d'une liaison logique entre deux machines n'est pas du tout nette et qu'elle est en grande partie dépendante du système. Par conséquent, suivant le système, il y a plus ou moins d'outils à réaliser. Ici on a essayé d'énumérer un ensemble d'outils et des façons de les réaliser qui nous sont indispensables sinon souhaitables pour la réalisation d'une connexion entre deux systèmes.

2.3. CHOIX ET MOTIVATIONS

En ce qui concerne la réalisation de la connexion d'un système à un réseau, notre désir est de pouvoir le faire d'une façon systématique suivant des étapes bien définies. Des choix relatifs à chaque étape sont faits.

Ces choix sont en général soit d'ordre politique, soit relatifs à la stratégie de réalisation, c'est-à-dire d'ordre logiciel ou encore d'ordre matériel. Cependant le raisonnement inverse peut être envisagé à savoir que certains choix essentiels faits au départs peuvent influencer la méthodologie. La deuxième manière de procéder nous parait d'ailleurs meilleure car si on peut revenir assez facilement sur la modification d'une étape, on le peut moins en ce qui concerne un choix d'ordre politique par exemple.

En ce qui concerne la connexion de toute nouvelle machine au sein de notre CCR, on a fait un choix qui reste à notre avis rationnel pour toute connexion d'un système à un réseau : effectuer la mise au point du logiciel pendant l'exploitation aussi bien sur le nouvel ordinateur que sur le réseau.

Les autres choix, d'ordre général, sont :

- choix du niveau d'intégration du logiciel de connexion.

- choix du système comme entité terminale.

- choix d'ordre matériel.

On a consacré un paragraphe à chacun des points précédents, en essayant de le justifier au mieux. En même temps, les différentes étapes proposées, pour la réalisation de la connexion, sont exposées.

2.3.1. MISE AU POINT PENDANT L'EXPLOITATION

La contrainte principale qu'on se fixe lors de la réalisation d'une connexion, qui influe évidemment sur le choix des démarches à suivre est que le tout doit se faire dans un contexte d'exploitation. En effet, le nouvel ordinateur arrive avec un système qui peut être mis en exploitation à l'aide notamment des terminaux qui lui sont propres, et cette exploitation commence tout de suite.

De la même façon côté C.C.R. où on fournit déjà un certain nombre de services, il est hors de question de stopper l'exploitation pour de longues périodes, solution jugée inacceptable par les utilisateurs. Cependant on est quand même obligés d'arrêter l'exploitation sur les deux pendant de petites périodes, ne serait-ce que pour faire des générations (en incluant le logiciel de connexion, on verra sous quelle forme) mais le maximum de tests doit s'effectuer en "ligne", pendant les périodes d'exploitation et sans "trop" gêner les utilisateurs présents.

Pour ce faire, on doit procéder par étapes.

La première consiste à bien étudier le système à connecter d'abord d'un point de vue spécification externe, de façon à savoir au mieux les possibilités offertes aussi bien pour la réalisation d'une application utilisateur que d'une application système. Ensuite, en particularisant le problème de la connexion, on peut concevoir une application utilisateur ou système qui n'effectue qu'une partie du travail souhaité, qu'on peut facilement mettre au point du fait qu'on n'a pas modifié le système et par conséquent on est à peu près sûrs de sa fiabilité.

On peut, par exemple, dans un premier temps, utiliser des gestionnaires de la ligne un de chaque côté de celle-ci, en essayant d'échanger des messages entre deux périphériques qui se trouvent sur chaque ordinateur ou encore échanger des données des deux programmes exécutés se trouvant un de chaque côté. De cette manière, on teste la liaison d'un point de vue "physique". Cette étape, entre autres, peut mettre en évidence certains problèmes purement matériels (comme interface matérielle non respectée par l'un des ordinateurs, etc...)

Ensuite, on peut chercher à avoir une utilisation plus rationnelle de la liaison physique ainsi obtenue en se définissant deux applications qui s'échangeront des informations en utilisant cette liaison physique. Cela peut, par exemple, être un utilitaire de transfert de fichier.

De cette manière, on arrive à avoir une première validation du logiciel de connexion sans avoir à toucher au système. Notons quand même un autre avantage de cette méthode c'est que les tests peuvent être effectués pendant l'exploitation car au pire le processeur utilisateur qui supporte l'utilitaire "avorte" tandis que le système continue à travailler pour le reste des utilisateurs.

La prochaine étape consiste à étudier les spécifications internes du système pour voir à quel niveau hiérarchique on doit intégrer le logiciel de connexion en étudiant plus particulièrement, une fois le niveau choisi, les interfaces de ce niveau avec le reste (ch.2.1.). D'ailleurs, les différents interfaces entre les différents niveaux du système risquent d'influer sur le choix du niveau d'intégration. Car encore une fois, notre souci principal est de modifier le moins possible les interfaces entres les différents niveaux.

2.3.2. NIVEAU D'INTEGRATION

Une première idée serait la possibilité d'intégrer, d'un point de vue fonctionnel, le logiciel de connection comme un DRIVER plus complexe qui sache faire la gestion d'un protocole et d'une procédure et qui est obtenue soit en modifiant le DRIVER initial qui gérait la ligne soit en utilisant celui-ci.

L'avantage d'une solution de ce style est que l'on ne rajoute pas des niveaux supplémentaires entre les niveaux du système d'exploitation et que les interfaces entre les différentes niveaux ne sont absolument pas touchés. Les modifications sont donc très localisées.

Une autre solution serait d'introduire le logiciel de connexion dans le niveau hiérarchiquement supérieur au moniteur d'entrées-sorties, ou encore d'en créer un supplémentaire entre les deux (figure 2.2.).

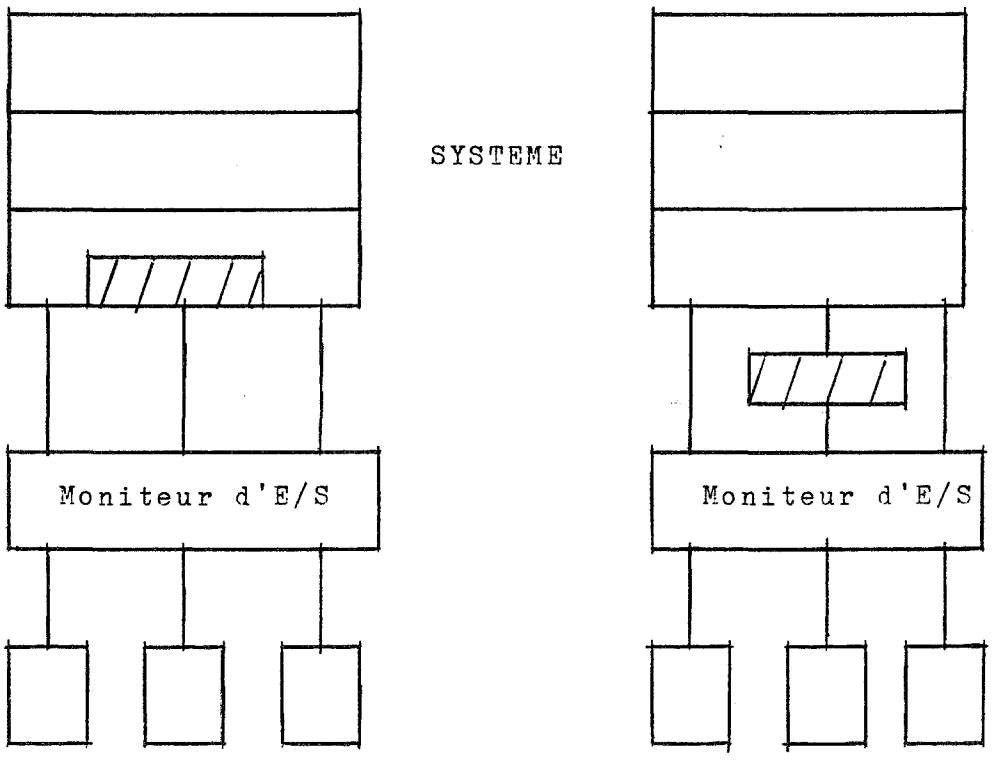


Figure 2.2

L'intégration du logiciel de connexion, à l'intérieur même du moniteur d'entrées-sorties, peut aussi être envisagé, ce qui veut dire en d'autres termes sa réécriture.

Par exemple, pour l'architecture DSA de CII-HB réalisée pour gérer l'interconnexion des ordinateurs HB-64 et MINI-6, le moniteur d'entrées-sorties a été complètement réécrit afin que les terminaux locaux et distants aient la même interface vis-à-vis de lui.

Un autre exemple de réalisation du logiciel de connexion en rajoutant des couches supplémentaires est celui réalisé sur un IBM 3704 ou 3705 qui sert de FRONTAL à un IBM 360 ou 370 pour permettre l'accès aux services du 370 à partir des terminaux compatibles 3270 reliés à TRANSPAC (RES-17). En effet, le logiciel du FRONTAL (XMEM) est structuré en couches hiérarchisées. PLusieurs couches ont été rajoutées au logiciel existant qui réalisait un concentrateur pour des terminaux qui lui étaient locaux. Chaque couche rajoutée sert à la gestion d'un protocole donné. Ces couches sont :

- Niveau trames et niveau paquets X 25
- Un protocole de bout en bout (BBX 25) qui gère les échanges entre FRONTAL et TERMINAL.
- Un émulateur BSC 3270 qui assure, pour le compte des terminaux, l'interface entre les messages véhiculés sur le réseau et les messages BSC échangés avec le central IBM.

Une fois l'intégration faite à un niveau donné, on peut valider le produit ainsi obtenu en faisant d'abord des tests similaires à ceux de l'étape précédente en ayant cette fois en plus tous les avantages d'un protocole, à savoir : comptabilité des lettres échangées, contrôle sur le flux, etc... Ceux-ci devraient nous permettre de mettre au point la gestion du protocole (qui nous servira par la suite comme un moyen sûr de transport).

A ce stade là on peut aussi s'occuper du problème de la réalisation d'une gestion de REVEILS qui nous servira pour les réémissions. Tous les systèmes possèdent une gestion élémentaire du temps mais tous n'offrent pas la possibilité d'armer des REVEILS. Cependant, la plupart possède des moyens de suspension d'un processus pendant un délai donné ou des primitives équivalentes (RES-8).

Et ce n'est qu'en dernière étape qu'on s'intéresse aux problèmes de l'interprétation. C'est-à-dire que d'abord on est sûr du moyen de transport avant de s'intéresser au contenu des informations transférées. Le problème d'interprétation étant traité au chapitre suivant (3.2), ici, on se contentera de le citer en situant surtout à quel moment on doit le résoudre. Cependant, on doit quand même tenir compte du fait qu'il y aura une traduction à un moment donné, ne serait-ce que pour déterminer la place fonctionnelle de l'interprète sans s'occuper de sa réalisation.

Une fois la liaison rendue ainsi opérationnelle, on peut chercher à optimiser le rendement de la liaison en utilisant, par exemple, des deux côtés des algorithmes dépendant, de mise en file et de retrait des messages (GEN-8).

2.3.3. LE SYSTEME ENTITE TERMINALE

Logiquement, l'utilisateur accède à un service. Dans la suite, nous préfèrons choisir comme entité terminale : le système.

D'abord, parce qu'un système regroupe un certain nombre de services et en effectuant la connexion du système, l'ensemble de ces services est connecté, donc on évite une duplication de code. C'est le cas du service TRANSIRIS (GEN-2) développé par la CII sur un ordinateur IRIS80 qui offre aux utilisateurs distants l'ensemble des services du système (SIRIS 8).

Ensuite tout nouveau service intégré dans le système est automatiquement connecté (sans aucun frais supplémentaire). C'est-à-dire tant que le J.C.L. (langage de commande) du système n'est pas modifié, aucune modification du logiciel de connexion n'est pas nécessaire.

En outre, le système étant maintenu par le constructeur, des nouvelles versions peuvent apparaître avec des spécifications internes éventuellement modifiées. C'est en effet les interfaces système-service qui peuvent être modifiés tandis que les spécifications externes et le J.C.L. le sont rarement. Donc, en principe, on n'a pas à reconsidérer le problème à chaque nouvelle version.

En effet, le degré d'indépendance d'une application vis-à-vis du système, est plus ou moins grand suivant les systèmes mais souvent fait partie des spécifications internes, moins bien connues et exposées par les constructeurs, que les spécifications externes, et sujettes à modifications sans préavis.

2.3.4. DU POINT DE VUE MATERIEL

Jusqu'ici, on a vu des choix relatifs au logiciel. En fait, le choix à faire lors de la réalisation d'une connexion ne porte pas seulement sur le logiciel mais aussi sur des considérations matérielles. Voyons un exemple :

Le choix du mode de fonctionnement d'un coupleur gérant la ligne peut jouer un rôle important. Pour la première de notre réalisation (voir 4.2.1. transfert de fichiers) on avait choisi le fonctionnement du coupleur sous le control d'un IOP. (il s'agit d'un processeur d'entrées-sorties). Dans ce mode de fonctionnement l'entrée-sortie se déroule sous le contrôle non pas de l'unité centrale mais de l'IOP qui avise cette dernière en fin d'entrée-sortie, donc gain de temps à cause du parallélisme ; par contre, on n'avait pas la possibilité de faire les entrées-sorties sur code d'arrêt.(entrée-sortie sur compte d'octet seulement). Ceci n'était pas gênant pour notre réalisation car comme on voulait transférer aussi des fichiers binaires (texte transparent) les entrées-sorties sur compte d'octet nous convenaient très bien.

Par contre, pour la réalisation de la suite (voir 4.2.1.) la longueur variable (qui peut être ramenée à une longueur fixée maximale en transférant des caractères inutiles) d'une part et l'irrégularité de la fréquence des envois d'autre part faisaient qu'on perdait quelquefois quelques caractères en réception (non mise en réception suffisamment vite), ceci introduisait un décalage et il n'y avait pas de possibilité simple de reprise. Il fallait réinitialiser, la réception mais le problème était de savoir quand ? puisqu' on ne s'apercevait de l'anomalie "qu'en face" et qu'on n'avait aucun moyen d'envoyer des informations "d'en face" car justement la réception "chez nous" fonctionnait mal.

Donc la nécessité d'une information de contrôle indiquant la fin d'un message était évidente. Ceci, afin que toute réception soit terminée en un temps fini,même en cas de perte. Mais les données reçues n'étaient pas toujours valides (manque éventuel de début) donc on voyait la nécessité d'une procédure de transmission ou du moins les fonctions d'une procédure permettant d'envoyer des informations en "transparent" et un contrôle redondant sur l'ensemble du message pour détecter les erreurs.

On voit donc que le choix initial de mettre le coupleur sous le contrôle de l'IOP qui était bien justifié dans un cas précis (transfert de fichiers) s'est avéré complètement inefficace pour le reste.

CHAPITRE III

TECHNIQUE DE CONNEXION D'UN SYSTEME

La nécessité d'un protocole étant ressentie, on examinera après quelques brefs rappels, le protocole "ordinateur connecté" (POC) qui permet de gérer la connexion d'un ordinateur au reste de la communauté. On verra ces deux aspects, connexion à un service et connexion d'un terminal distant. Ce dernier aspect est nécessaire pour la banalisation de tous les terminaux du centre.

Ensuite, on passera en revue le problème des erreurs. Les erreurs de transmission étant inévitables en environnement télé-informatique, le problème de traitement de ces erreurs au niveau du protocole (voire de la procédure) présente un intérêt particulier. Des questions donc de fiabilité et surtout des problèmes de reprises seront traités.

La deuxième partie de ce chapitre est consacrée aux problèmes de l'interprétation. L'existence d'un langage interne unique véhiculé entre les différents éléments du réseau d'une part et la conservation du système du constructeur d'autre part, impliquent une traduction (ou interprétation) à un endroit ou à un autre. La place d'un tel interpréteur n'est pas à priori fixée et on discutera de sa localisation et de sa répartition éventuelle.

3.1. PROTOCOLES

3.1.1. QUELQUES RAPPELS SUR LES PROTOCOLES

Un protocole est l'ensemble des conventions qui règlent les échanges entre plusieurs objets de même niveau. Ces objets échangent des informations par l'intermédiaire d'une interface virtuelle (GEN-3).

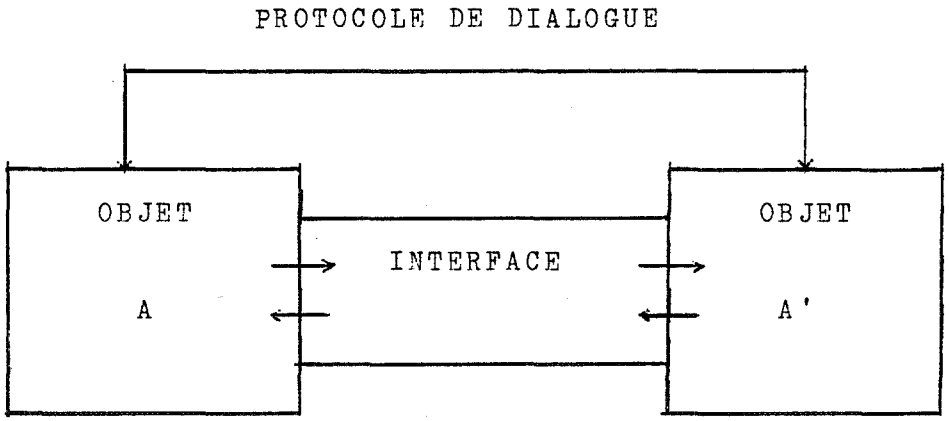


Figure 3.1.

Le rôle de deux objets peut être d'assurer une interface entre deux objets de niveau supérieur. En effet, pour des soucis de modularité, il est préférable de répartir les services en plusieurs niveaux.

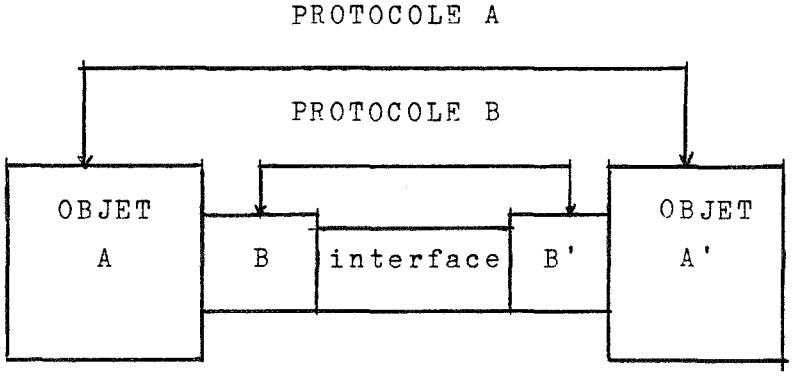


Figure 3.2.

Différents niveaux de protocoles peuvent ainsi se superposer, les niveaux inférieurs déchargeant les niveaux supérieurs d'un certain nombre de problèmes et assurant pour eux des services. Voici un exemple de hiérarchie de protocoles:

<table>
<tr><td colspan="2">Système de banque de données</td><td colspan="2">Langage de commande réseau</td></tr>
<tr><td>Soumission de travaux par lots</td><td colspan="2">Accès à des services de temps partagé</td><td>Protocole d'appareil virtuel</td></tr>
<tr><td colspan="2">Protocole Boîte aux lettres et communication inter-opérateurs</td><td>Lancement de travaux</td><td>Transfert de fichiers</td></tr>
<tr><td colspan="4">Station de transport : permet l'échange de messages</td></tr>
<tr><td colspan="4">Commutation de paquets : permet l'échange de paquets</td></tr>
<tr><td colspan="4">Transmission de données : permet l'échange de suites de bits (suivant une procédure de transmission : BSC, TMM, ...)</td></tr>
<tr><td colspan="4">Réseau téléphonique : permet la transmission de signaux - Protocole imposé par la règlementation des PTT.</td></tr>
</table>

Figure 3.3

Nous allons passer en revue les principaux services que peut réaliser un protocole de dialogue :(GEN-11)

- Négociation d'un échange d'informations : elle a comme but d'établir le contact entre deux objets du même niveau pour engager un dialogue. Elle permet leur synchronisation qui est nécessaire pour effectuer le contrôle d'erreurs et de flux.

- Séquencement des informations : les différentes interfaces utilisées dans les communications posent souvent une limite quant à la taille des messages. On est amené à fragmenter les longs messages, ce qui implique un réassemblage à l'arrivée.

- Contrôle d'erreurs : il s'agit de détecter les pertes, les altérations et les répétitions. Il faut aussi remédier à l'erreur, c'est-à-dire qu'il faut un mécanisme pour redémarrer la transmission à un point sûr. Pour cela, l'émetteur garde une copie d'un certain nombre de messages parmi les derniers envoyés. Pour éviter la conservation indéfinie de ces messages, il faut que le récepteur avertisse l'émetteur de leur bonne réception (en les acquittant).

- Contrôle de flux : c'est l'asservissement de l'émetteur par la capacité du récepteur. C'est un mécanisme qui régit la cadence des messages transférés. Le récepteur alloue de la place à l'émetteur suivant ses capacités de réception.

- Passage d'évènements asynchrones : comme pour les entrées-sorties, on a besoin d' informations de contrôle autres que les données (interruption). De la même manière, il nous faut un mécanisme qui nous permette l'acheminement des informations de contrôle parallèlement aux données transférées.

3.1.2. LE PROTOCOLE ORDINATEUR CONNECTE

Il s'agit d'un protocole suffisamment complet pour gérer toutes les communications entre deux ordinateurs, et simple pour être facilement introduit sous un système d'exploitation donné. Il gère notamment :(CCR-10)

- le séquencement des informations
- le contrôle du flux
- le contrôle d'erreurs
- le passage d'évènements asynchrones.

En fait, il regroupe les deux aspects du protocole suivant :

- le protocole appareil terminal, géré dans les voies logiques de terminaux distants (par rapport au FRONTAL) qui sera traité au paragraphe suivant (3.1.3.)
- le protocole appareil service, géré dans les voies logiques des services.

Les échanges se font sur une voie logique de communication (VLC). Une VLC est le chemin logique reliant un utilisateur ou un service et l'interpréteur (voir 3.2.2.). L'adressage fourni par le Protocole Appareil Service utilise le numéro de SU (unité logique représentant la vision que le correspondant a des VLC) attribué à la voie logique; chaque pli contient ce numéro pour indiquer sur quelle voie se fait l'échange.

Il y a deux types de plis : les lettres (LT) et les acquittements (ACK).

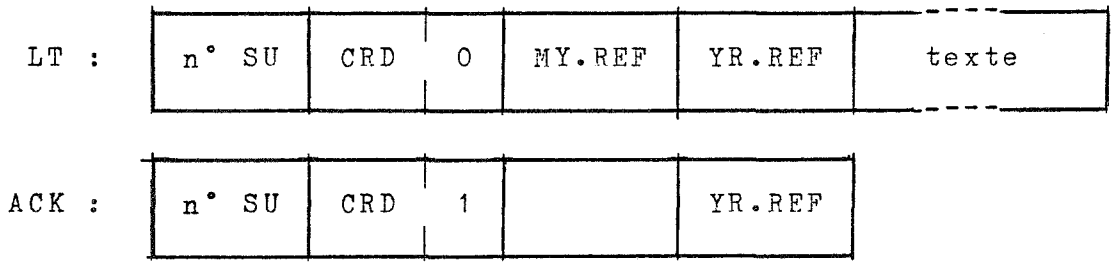


Figure 3.4.

Seules les lettres formées d'un en-tête Protocole et d'un texte transparent au protocole, permettent le transport de données.

Toutes les lettres sont référencées (MY.REF) séquentiellement, ce qui permet :

- le bon séquencement : la lettre n n'est reçue que si la lettre n-1 a été reçue.

- le contrôle d'erreur : lorsqu'une lettre arrive, on l'acquitte en envoyant sa référence (YR.REF) à l'émetteur. Un même acquittement peut ainsi acquitter plusieurs lettres. Un réveil est armé par l'émetteur à l'envoi d'une lettre pour éviter une attente indéfinie de l'acquittement. Si ce réveil arrive à terme, donc que l'acquittement n'est pas venu, la lettre est renvoyée. Au bout d'un certain nombre de renvois non réussis, on déclare l'envoi impossible.

Le contrôle de flux se fait par échange mutuel de crédits (CRD) qui représentent le nombre de lettres que le récepteur autorise l'émetteur à envoyer jusqu'à attribution de nouveaux crédits.

Les crédits et acquittements sont passés dans les plis ACK, mais aussi dans les en-tête de lettre.

Structure de données : les données sont dynamiques pour assurer la réentrance. Notamment les numéros des SU ne sont pas fixés a priori.

Un contexte est associé à chaque SU, il est créé dynamiquement à l'ouverture de la SU et accessible par la mise en place d'un mécanisme standard d'accès en fonction du numéro de SU. A la fermeture de la voie logique le contexte et le moyen d'y accéder sont détruits.

Ce contexte contient les informations nécessaires à l'idenfication de la voie : numéro de SU et type (terminal ou service) tous deux fournis à l'ouverture, et à la gestion du protocole : références des lettres, compteur d'envoi, crédits, etc...

Une émission est réalisée par la fabrication d'une lettre : l'en-tête est formaté en fonction des informations contenues dans le contexte SU, le texte étant les données à transmettre qui se trouvent dans le buffer de l'IOCB (bloc de contrôle relatif à une entrée-sortie). L'émission ne peut être faite (c'est-à-dire le passage du pli à la gestion de la ligne sur laquelle il sera transmis) que si l'on dispose de crédits d'émission (attribués par l'autre extrémité de la voie). Sinon l'IOCB est stocké pendant un certain temps dans le contexte en attendant l'arrivée de nouveaux crédits. On introduit ainsi un niveau de désynchronisation entre l'alimentation de la voie et l'émission sur la ligne, ceci afin de pouvoir contrôler le flux.

L'évènement "fin d'entrée-sortie" relatif à l'IOCB n'est déclenché que lorsqu'on est sûr que la lettre correspondante est bien arrivée (elle a été acquittée) ou qu'elle s'est perdue (plusieurs renvois successifs sans succès).

Un IOCB de réception est stocké dans le contexte SU. C'est le nombre de ces IOCB qui détermine les crédits que la gestion de la voie attribue à son interlocuteur. Une annulation de réception se traduit donc par la récupération d'un IOCB stocké et diminution des crédits de réception.

Arrivées de pli : une lettre est contrôlée à son arrivée : sa référence doit être celle de la lettre précédente plus un (modulo 256) et il y a au moins un IOCB de réception (donc un crédit) pour la recevoir. Elle est alors acquittée et son texte est rangé dans le buffer de l'IOCB de réception en attente dans la voie (fin de réception).

L'arrivée de l'acquittement d'une lettre, par pli ACK ou dans l'en-tête d'une lettre, permet d'acquitter le (ou les) IOCB d'émission correspondant(s) (n'oublions pas que plusieurs lettres peuvent être acquittées à la fois) et l'attribution de nouveaux crédits autorise de faire les émissions en attente (s'il y en a).

3.1.3. TERMINAUX DISTANTS

Les voies logiques des terminaux distants sont gérées par le protocole appareil terminal qui contient toutes les possibilités du protocole appareil service et offre en plus un contrôle du dialogue qui est codé dans un mot de l'en-tête d'une lettre.

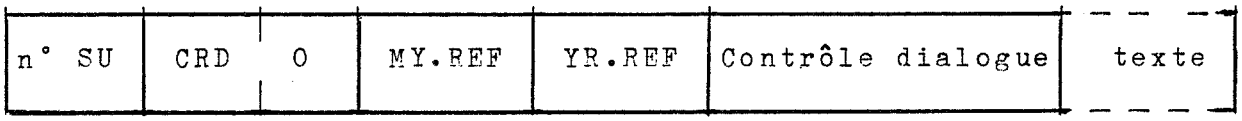


Figure 3.5

Ce mot se présente sous la forme :

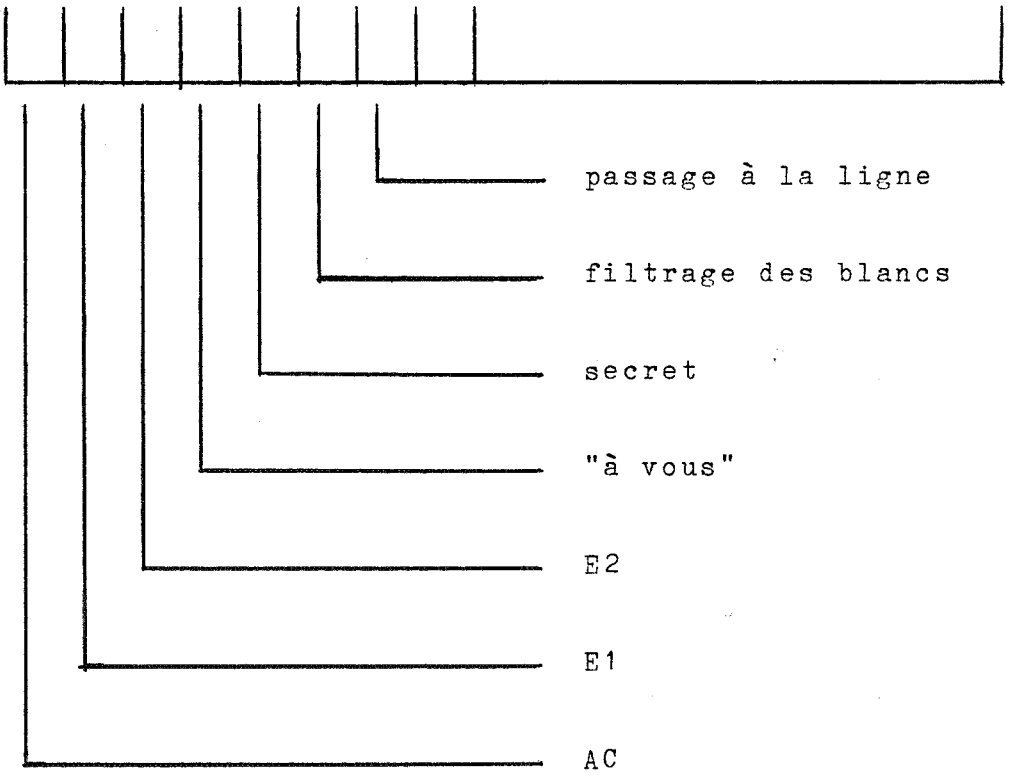


Figure 3.6

Les bits passage à la ligne, filtrage des blancs et secret servent à transmettre à la gestion du terminal distant les fonctions correspondantes qui, dans le cas du terminal local, sont exécutées par la gestion du terminal.

Le passage de la main se fait par le bit "à vous". Pour respecter le contrôle de dialogue implicitement défini par le LCE, la main est donnée au terminal distant lorsque le correspondant utilisateur se met en réception sur la voie, le terminal perdant la main dès qu'il envoie une ligne.

Les bits AC, E1 et E2 permettent la transmission des évènements asynchrones : "appel au correspondant", "attention1", "attention2".

On peut utiliser les lettres sans leur texte, uniquement pour transmettre le contrôle de dialogue (les évènements notamment) mais les crédits ne servent que pour les lettres avec texte.

Remarquons que dans le mot de contrôle de dialogue situé dans l'en-tête d'une lettre les trois premiers bits ne sont utilisés que dans le sens correspondant utilisateur vers terminal. Seul le "à vous" sert dans les deux sens. Ce contrôle de dialogue peut passer aussi bien dans l'en-tête d'une lettre avec texte que dans celui d'une lettre sans texte suivant qu'il peut être associé ou non à une demande d'émission du correspondant ou du terminal.

Les évènements AC, E1, E2 représentés respectivement par les bits 0, 1, 2 du mot de contrôle de dialogue, sont reconnus au niveau protocole et répercutés chez le correspondant utilisateur de la même manière que s'ils venaient d'un terminal local. Si l'évènement arrive dans l'en-tête d'une lettre avec texte, ce dernier est d'abord transmis au correspondant avant la prise en compte de l'évènement.

Pour les fonctions passage à la ligne, filtrage des blancs et secret, on se contente de positionner les bits correspondants dans l'en-tête de la lettre à émettre, c'est à l'interlocuteur de traduire ces fonctions d'une façon ou d'une autre pour qu'elles soient exécutées par la gestion des terminaux de l'ordinateur connecté.

Commandes d'ouverture et de fermeture d'une voie (RES-13) : le protocole ordinateur connecté est un protocole de communication géré sur une voie ouverte. Or, il faut bien ouvrir la SU avant de commencer les échanges. C'est pour cela qu'ont été définies des commandes spéciales, qui ne font pas partie du protocole ordinateur connecté même si elles passent sur la même ligne.

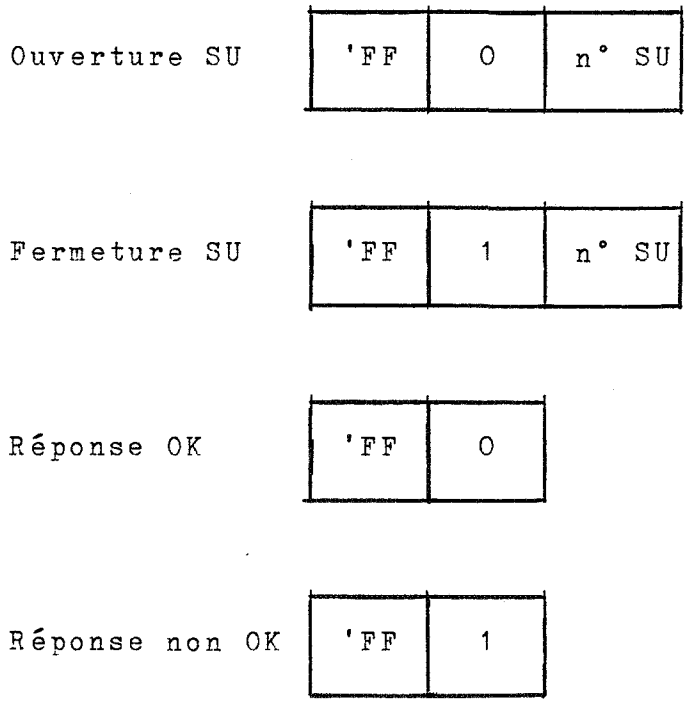


Figure 3.7

L' ordre d'ouverture est toujours à l'initiative de l'opérateur du CCR. L'ordinateur connecté répond par un accord ou par un désaccord. L'ordre de fermeture peut venir de deux côtés mais une réponse est possible seulement dans le cas où la fermeture provient de l'opérateur.

3.1.4. TRAITEMENT DES ERREURS

3.1.4.1. Le protocole et la fiabilité

Un utilisateur du réseau doit voir le service avec lequel il travaille depuis son terminal comme le voit un utilisateur local. Le réseau doit introduire le moins d'erreurs possibles car dans le cas contraire, l'expérience l'a prouvé, les utilisateurs fuient le réseau qui reste l'outil des seuls initiés. Parmi ces erreurs, les plus conséquentes sont celles qui conditionnent la fiabilité et la disponibilité.

Si l'utilisateur accepte que son terminal ne s'utilise pas exactement comme un terminal connecté directement, il tolère difficilement une diminution de la disponibilité du service. Les matériels et les logiciels mis en jeu sont nombreux et la moindre défaillance d'un de ces éléments peut compromettre le travail de l'utilisateur.

Un réseau utilise un ensemble de matériels et de logiciels qui dialoguent entre eux à l'aide de protocoles. Sa fiabilité est conditionnée par celle de chacun de ces composants. On ne s'intéressera pas ici à la fiabilité du matériel qui ne fait pas partie de cette étude, on constatera seulement qu'elle n'est pas totale.

Reste la capacité des logiciels et des protocoles. Les protocoles sont capables de récupérer certaines erreurs, mais ils peuvent en laisser passer d'autres. Les logiciels qui réalisent ces protocoles peuvent aussi apporter des erreurs.

Pour remédier à ces erreurs, on essaye de définir le logiciel le plus modulaire possible (architecture à niveau) avec des interfaces bien définies et réalisées de telle sorte que la défaillance d'un niveau n'en affecte pas les autres. On est ainsi amené à un empilement de protocoles (RES-3).

Voyons un exemple : le logiciel de connexion de deux ordinateurs a la structure suivante :

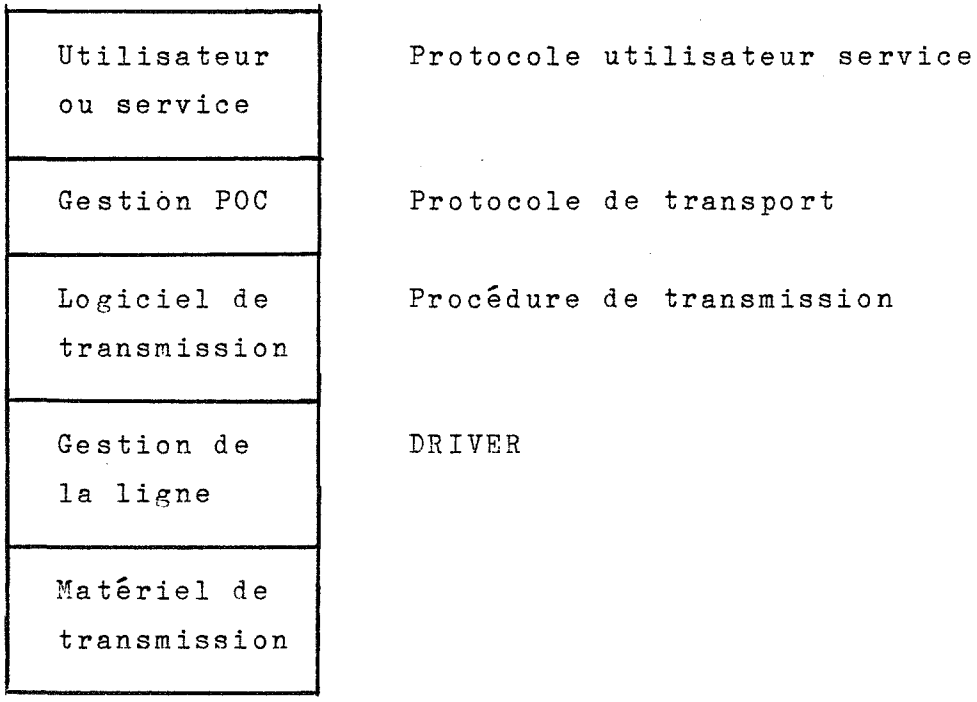


Figure 3.8

Chacun de ces niveaux utilise un protocole pour dialoguer avec son interlocuteur. Il est bien évident qu'on ne peut pas toujours faire en sorte que la défaillance d'un niveau n'affecte pas les autres. Car si on peut imaginer qu'à la défaillance du niveau procédure on peut, dans certain cas, fonctionner sans lui, il n'en est pas de même en ce qui concerne la gestion d'une ligne. Par contre, on peut espérer que la défaillance d'un niveau (procédure par exemple) n'entraînera pas la défaillance de la gestion du même niveau pour la réalisation d'une autre connexion. De même, la défaillance du protocole utilisateur service en ce qui concerne la gestion d'un utilisateur ne doit pas entraîner de perturbations pour les autres.

3.1.4.2. Le problème de reprises

Le matériel n'est pas parfait pas plus que le logiciel, les pannes sont donc inévitables et il faut un moyen d'y remédier si elles se produisent, c'est-à-dire, prévoir des points de reprises à des endroits dont on peut reprendre la communication en étant sûr qu'il n'y ait pas eu de perte, ou au pire être conscient de l'ampleur des pertes s'il y en a.

Pour ce faire, il faut sauvegarder l'état du système pendant le fonctionnement normal. Cette sauvegarde peut être déclenchée, soit à des intervalles réguliers, soit à l'arrivée des évènements susceptibles de changer l'état du système. Les informations sauvegardées doivent se trouver sur le support le plus fiable possible afin qu'une panne ne puisse pas les modifier ni les détruire. Il est bien évident que ce qui risque d'être perdu ce sont les informations traitées entre le moment de la dernière sauvegarde et le moment où survient la panne (RES-10). Essayons de voir ce que ces informations représentent dans le cas de la gestion du protocole ordinateur connecté.

D'abord en ce qui concerne les lettres reçues, deux cas peuvent se présenter :

- La réception a été complètement terminée et la lettre passée au niveau supérieur qui utilise le service de transport. C'est alors à ce niveau supérieur de prévoir ses points de reprise.
- La réception n'a pas été complètement terminée, dans ce cas, la lettre ne sera pas acquittée et elle sera renvoyée. Mais comme elle sera renvoyée un certain nombre de fois, il faut espérer que le redémarrage se fera avant que ce nombre ne soit épuisé.

En ce qui concerne les lettres émises si la panne survient au moment où on est en train de l'émettre en face, on s'en apercevra et on ne va pas l'acquitter. Donc à la reprise on sera amené à la réexpédier (il faut donc que cette lettre fasse partie des informations à sauvegarder).

Si l'émission est complètement terminée la lettre est en attente d'acquittement pour pouvoir la réexpédier éventuellement, donc il faut aussi la sauvegarder.

En ce qui concerne la perte des acquittements, il n'y a pas de problème si un envoi d'acquittement à intervalles réguliers est prévu.

Les lettre qui sont en attente d'acquittement ou en train d'être émises (qu'il faut donc sauvegarder) se trouvent dans le contexte des différentes SU qui sont ouvertes. En outre, ces contextes contiennent toutes les informations nécessaires pour la fabrication des acquittements. Donc il faudrait sauvegarder ces contextes ainsi que le moyen d'y accèder chaque fois qu'ils sont modifiés. Ceci n'est pas possible (sinon on passerait notre temps à faire des sauvegardes !).

Cependant, il faut sauvegarder les tables qui nous permettent d'y accéder chaque fois qu'il y a une ouverture ou une fermeture d'une SU, faute de quoi, au redémarrage on ne saurait pas quelles SU sont ouvertes ou fermées.

Quant au contexte, proprement dit, on pourrait le sauvegarder à intervalles réguliers à condition de pouvoir le faire en même temps sur les deux ordinateurs et encore faudrait il un moyen d'indiquer l'un à l'autre quand il redémarre !

Citons maintenant le minimum parmi les informations du contexte qu'il faut constamment maintenir pour nous permettre de redémarrer correctement :

- Crédit en réception (et file de lettres en réception) : sinon on aurait des réceptions indéfinies pour le niveau supérieur.

- File de lettres en attente d'acquittement : pour la réémission éventuelle.

- La référence de la dernière lettre reçue (YR-REF) doit toujours être à jour. Si au moment de la panne YR-REF vaut n et que la dernière valeur sauvegardée est n-p, après le redémarrage on va recevoir la lettre de référence n+1 et on demandera la réémission de la lettre de référence n-p+1. De l'autre côté, on ne pourra pas la réémettre puisque toutes les lettres jusqu'à la référence n ont été acquittées et sont donc perdues par l'émetteur.

En fait, seul le nombre de crédits d'émission et la référence de la dernière lettre émise et acquittée peuvent ne pas être maintenus constamment. Car ces informations sont reçues périodiquement par le biais des acquittements. Après le démarrage, on les récupèrera. Ceci suppose toutefois que les deux ordinateurs ne tombent pas en panne en même temps.

C'est donc un nombre important d' informations qu'il faut sauvegarder compte tenu de la fréquence des sauvegardes. Ce qui nous a amené dans un premier temps à répondre tout simplement par une fermeture au moment du redémarrage, en se disant que l'utilisateur est bien placé pour savoir où il en était dans son dialogue avec le service et pour reprendre son travail une fois la connexion redemandée.

3.2. INTERPRETATION

3.2.1. NECESSITE DE L'INTERPRETATION

On a vu au chapitre I l'existence d'un langage unique LCE (langage de commande externe) mis à la disposition des utilisateurs, afin qu'ils puissent accéder à n'importe quel service du CCR. Par ailleurs, les ordinateurs du CCR conservent leur langage de commande propre. On a déjà vu les raisons de ce choix au chapitre I.

L'existence de ces deux langages implique donc la transposition de l'un à l'autre (LCE en JCL) à un instant donné. Ainsi, on voit apparaître la nécessité d'un traducteur pour effectuer cette opération. Par la suite, on conservera comme terme, non plus traduire mais interpréter le langage de commande (LCE) car les commandes de l'utilisateur seront partiellement exécutées au fur et à mesure qu'elles seront émises (donc interprétées).

Le travail de l'interpréteur sera plus ou moins complexe suivant les commandes de l'utilisateur. Voyons un exemple: un utilisateur veut travailler avec le service conversationnel APL. Pour ce faire, il se présente devant un terminal et il demande le démarrage d'une session en faisant LOGIN. La commande sera récupérée par l'interpréteur, analysée par lui et si elle est correcte, la main est passée à l'utilisateur avec secret en entrée pour que l'utilisateur tape son mot de passe. Si ceci est exact, l'interpréteur associe l'accès de l'utilisateur (son terminal) à son identification. A partir de ce moment là, l'utilisateur est connu du CCR.

Il peut maintenant essayer de se connecter à APL. Pour cela, il frappe sa demande de connexion qui est récupérée par l'interpréteur et analysée. Si elle est correcte syntaxiquement, il envoie une demande de connexion à APL, ce qui suppose que l'interpréteur connaît la localisation de chaque service.

Le service peut répondre par oui ou par non. Dans le cas positif, une fois la connexion établie, l'interpréteur n'intervient plus. Tout ce qu'envoie l'utilisateur sera transmis à APL et vice versa, et ceci jusqu'à ce qu'une commande de déconnexion provienne de l'utilisateur (ou du service).

Pour finir sa session, l'utilisateur frappe la commande LOGOUT qui est récupérée par l'interpréteur et provoque la dissociation de l'identificateur de l'utilisateur et de son accès.

Exemple de session :

```
? LOGIN

xxxxxxx          pour entrée du mot de passe

? CNEX APL

APL à votre service

.
.
.                Echanges avec le service.
.
.

? DCNX

Déconnexion effectuée

? LOGOUT
```

Figure 3.9

3.2.2. PLACE DE L'INTERPRETE

Une première idée serait de localiser l'interprète sur un seul ordinateur qui connaîtrait tous les ordinateurs du centre et leur langage de commande (JCL) dans lequel il traduirait les requêtes des utilisateurs (CCR-7). L'avantage d'une telle solution est d'avoir une structure simple et de concentrer le logiciel sur le même ordinateur. Par contre, on oblige l'interpréteur à garder en mémoire de nombreuses commandes et la fiabilité de l'ensemble de la communauté diminue. En effet, une panne de l'ordinateur qui supporte l'interprète entraîne l'arrêt de l'ensemble. En plus, l'interprète risque d'être gros : proportionnel au nombre de machines connectées car toute connexion d'un nouvel ordinateur entrainera des modifications importantes pour introduire l'interprétation d'un nouveau langage de commande (JCL).

Une seconde idée, qui nous a paru meilleure, serait de reporter sur chaque machine l'interprétation de son langage de commande. Cependant, une fonction d'aiguillage, nécessaire pour assurer la transparence de la localisation des services, vis-à-vis des utilisateurs, doit être réalisée avant la traduction en langage local. En effet, il faut adresser les requêtes à interpréter vers les ordinateurs concernés. Par conséquent, une première partie de l'interprétation du langage utilisateur doit être faite avant l'accès à l'ordinateur concerné (Figure 3.11.).

On peut donc décomposer en deux la fonction d'interprétation :

- aiguillage des requêtes vers les ordinateurs concernés
- traduction en JCL par des interprètes locaux. Chaque interprète local simule pour le système d'exploitation un appareil réel et lui envoie du JCL. Aucune difficulté pour le système local d'assurer la gestion de ses services, car les commandes répercutées à un service sont les mêmes que si elles provenaient d'un terminal local. (Voir Figure 3.10.)

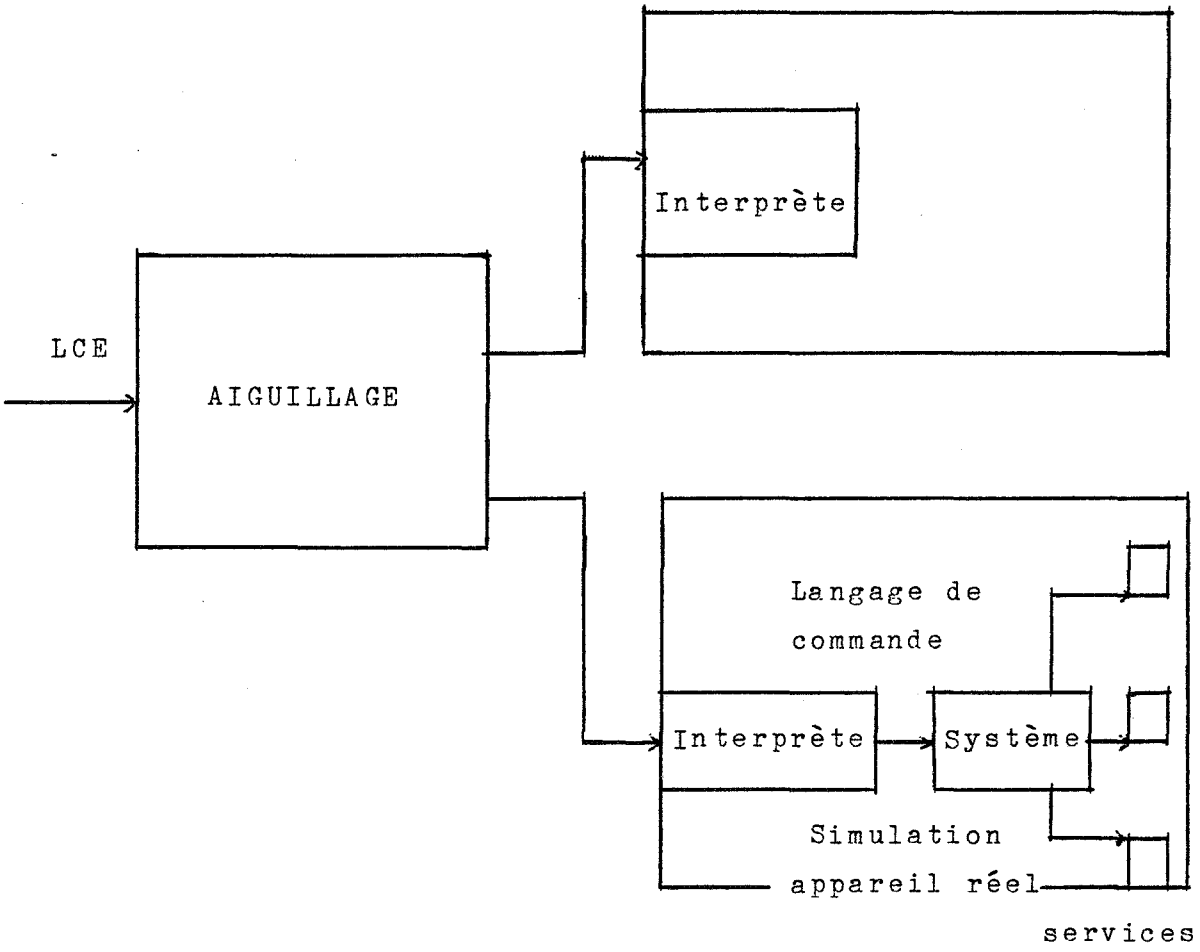


Figure 3.10.

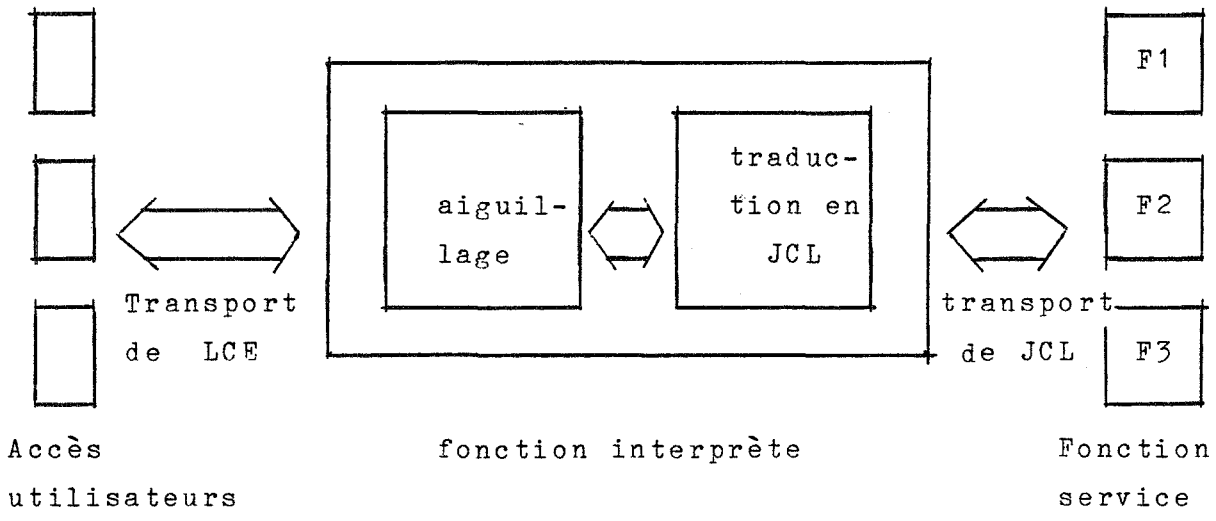


Figure 3.11.

Entre ces deux fonctions de l'interprète une fonction de communication est nécessaire, pour permettre des échanges entre elles. Comme on vient de le voir, ces deux fonctions sont réalisées sur deux ordinateurs différents, la fonction de communication qui les relie va utiliser les moyens de transport fournis par le réseau et il lui faut, en plus, un langage de communication. Ce langage arrive à l'interpréteur local et doit être traduit en JCL par lui.

Une première idée serait de choisir comme tel langage le LCE. Mais du fait que c'est un langage externe, adapté plutôt aux utilisateurs, il risquerait d'obliger les interprètes locaux à une traduction trop complexe et a été abandonné au profit d'un autre langage à la forme et aux fonctions plus primaires mais dont la traduction est plus facile et moins encombrante. On lui a donné le nom de langage interne (LCI). Ainsi, la première fonction de l'interprète fait en plus la traduction du LCE en LCI. Entre les différents éléments du réseau circule du LCI, et chaque interprète local traduit le LCI en langage de commande local (JCL).

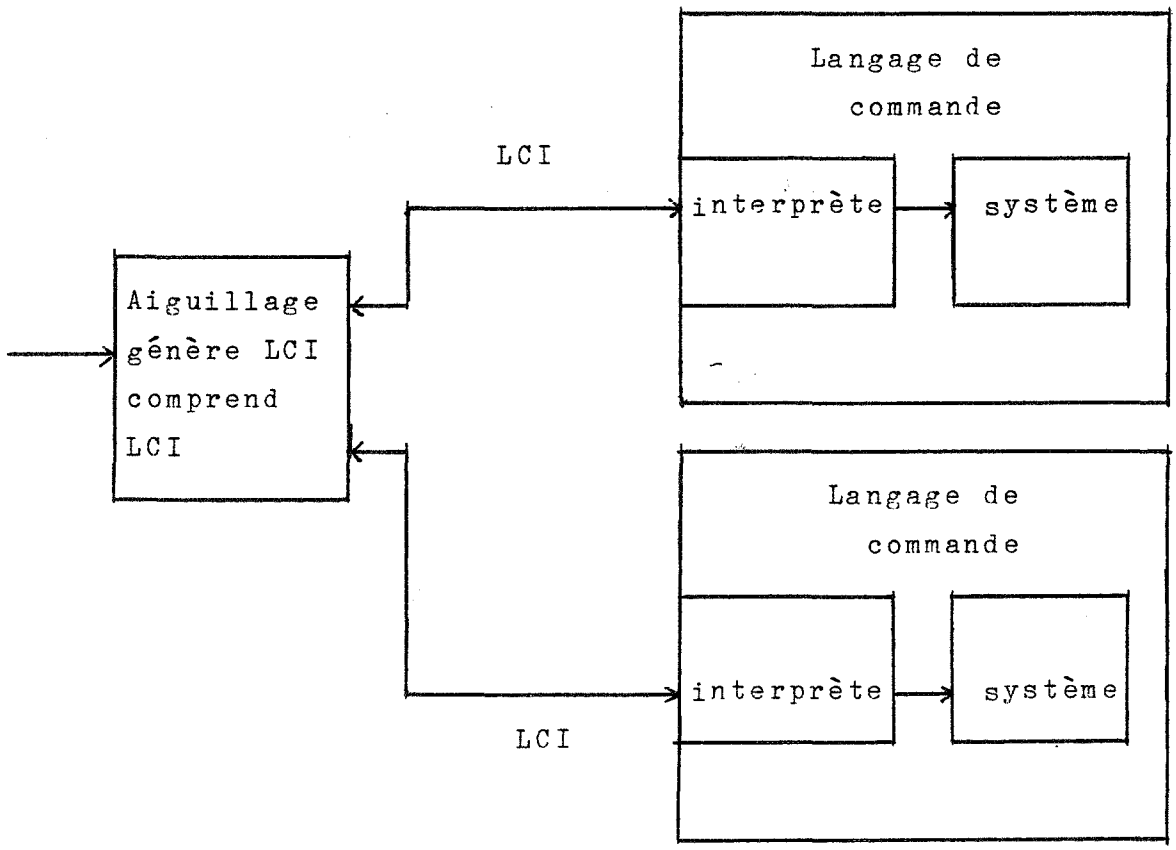


Figure 3.12.

Il apparait intéressant de donner à ce LCI un rôle plus important que celui de simple "code intermédiaire" dans l'interprétation du LCE.

En effet, considérons un service réseau (chapitre I.1.5) et rappelons en quoi cela consiste :

- la traduction LCI-JCL
- la partie du système d'exploitation nécessaire à l'application pour s'exécuter
- l'application (ou service) locale.

Cette entité communique avec l'aiguillage en LCI (c'est nécessaire pour pouvoir envoyer des réponses aux utilisateurs). La fonction aiguillage réalisée pour aller des usagers vers les services peut aussi être utilisée pour communiquer entre services puisqu'elle dispose des données nécessaires (localisation des services).

Cette structure nous permet de résoudre au niveau local les problèmes (justement locaux) de traduction LCI en JCL, problèmes liés au système d'exploitation et à la réalisation particulière des applications. Au niveau global, l'interprète ne s'adresse plus qu'à des entités services comprenant le LCI.

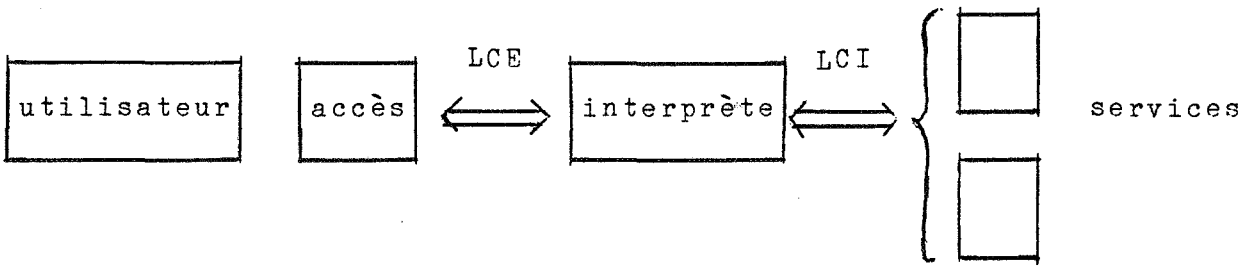


Figure 3.13.

Le LCE ne permet pas de distinguer les points d'accès (terminaux) ; il faut donc que chaque terminal ait un point d'entrée chez l'interprète ; on appellera voie logique de communication (VLC), le chemin logique utilisateur - point d'entrée chez l'interprète. C'est l'ensemble des VLC qui réalise la fonction de communication entre utilisateur et interprète.

Quant à la fonction de communication entre interprète et services, elle pourrait être incluse dans le LCI : la distinction entre applications serait alors faite au niveau LCI. Cette solution oblige l'interprète à connaître la localisation des services sur les différentes machines et à s'adresser à l'interprète du LCI sur chaque machine et a l'inconvénient de figer la structure des applications.

Une solution semblable à celle utilisée pour la communication utilisateur-interprète est préférable, c'est-à-dire que chaque service possède un point d'entrée chez l'interprète. Le chemin logique interprète-service est aussi appelé voie logique de communication.

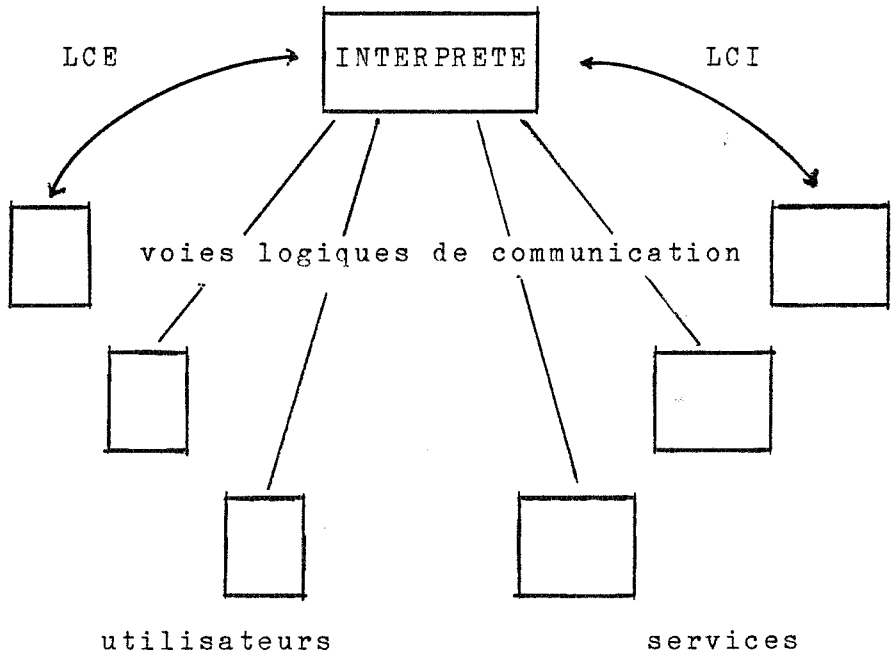


Figure 3.14.

En conclusion, on voit que pour intégrer un système d'un nouvel ordinateur, c'est-à-dire l'intégration de l'ensemble de ses services, au sein de notre CCR, outre l'élaboration d'un moyen de transport, il faut réaliser l'interpréteur du LCI en JCL. Cet interpréteur fait donc partie du logiciel de connexion et doit être utilisé par tous les services qui seront mis à la disposition du réseau. Autrement dit un service réseau pour effectuer la traduction LCI-JCL qui lui est nécessaire, fait appel à des routines de cet interpréteur.

3.2.3. AUTRES REALISATIONS DANS LE DOMAINE

1) SOC : (RES-12) Système d'ordinateurs connectés. Il s'agit d'un réseau homogène :les ordinateurs participants sont tous des IBM 360 ou 370. Le système réseau est physiquement réparti sous forme d'une tâche utilisateur sur chacun des calculateurs connectés.

Les utilisateurs communiquent avec le réseau par des commandes exprimées dans un langage de commande réseau (NCL) (LAN-3). Ces commandes sont soumises à un ordinateur local par le biais d'un travail réseau. Le système réalise les tâche suivantes :

a) prise en compte des travaux réseau
b) traduction et interprétation des commandes réseau
c) contrôle de l'exécution des requêtes et des échanges entre ordinateurs.

Un langage de commande interne (NIL) est utilisé pour faciliter l'interprétation des commandes réseau et les dialogues entre les différents ordinateurs.

2) IGOR :(LAN-4) Le but de cette étude était de doter le réseau CYCLADES d'un système réseau. Comme pour SOC, deux langages ont été développés : un langage externe (LE) fourni aux utilisateurs et un langage interne (LI) (LAN-2) transportable, puisque chaque ordinateur connecté doit disposer d'un interprète LI pour comprendre les ordres qu'on lui envoie. IGOR est le nom de cet interprète du LI.

3) UNIX :(RES-7) UNIX est un système d'exploitation (réalisé sur PDP 11) dans lequel on se propose de développer un langage de commande réseau en ajoutant des requêtes réseau au JCL. Le système remplit deux rôles : processeur local et machine d'accès au réseau. Les commandes réseaux respectent la forme des autres commandes du JCL, ce qui simplifie la vision de l'utilisateur. Cette solution reste liée au choix de la machine : le changement de machine implique le changement de langage de commande.

3.2.4. NOTION D'ENVELOPPE RESEAU ET SA LOCALISATION

L'importance en taille d'une enveloppe réseau est fonction de la partie de l'interpréteur qui effectue l'aiguillage et la traduction du LCE en LCI. En effet, quand une requête LCE concernant un service lui parvient, il effectue son analyse syntaxique et ensuite :

- soit il réalise en plus le contrôle sémantique de telle sorte que les commandes en LCI parvenant au service peuvent être exécutées immédiatement avec une garantie d'exactitude (car l'analyse sémantique aurait éliminé toute commande erronée). Dans ce cas, l'enveloppe réseau n'est pas importante car elle est déchargée de toutes les fonctions de l'analyse sémantique. Par contre, l'interprète se voit augmenté d'une certaine quantité d'informations lui permettant d'effectuer l'analyse sémantique.

- soit il se contente de laisser passer des informations sémantiques non contrôlées, en considérant que le service est mieux placé pour savoir si une commande est correcte ou non. Dans ce cas, l'enveloppe réseau est plus importante car elle effectue le traitement sémantique en plus mais on gagne de la place au niveau de l'interpréteur.

On serait tenté de dire que la première solution est meilleure car l'interpréteur est unique tandis que les enveloppes réseau sont multiples. Cependant, la seconde est plus souple pour l'adjonction d'une nouvelle application. En effet on n'a pas à modifier l'interpréteur chaque fois qu'une telle adjonction est en vue.

En plus, comme l'introduction d'un service se traduit en général par l'écriture de son enveloppe réseau, peu importe qu'elle soit un peu plus longue ou non car c'est un travail qu'il faut faire de toute façon.

Donc, d'une manière générale, les enveloppes réseau prendront en charge les problèmes sémantiques et elles se trouvent sur le même ordinateur que les services qu'elles englobent.

Cependant, la déportation des enveloppes réseau, dans certains cas particuliers, est envisageable à l'aide de procédures cataloguées. Dans ce cas là, ce n'est plus du LCI qui circule entre les deux ordinateurs mais du JCL et on peut toujours considérer que le service réseau "déborde" sur le FRONTAL. C'est le cas des services accessibles par l'intermédiaire d'un réseau externe (réseau CYCLADES) : on ne peut introduire quoique ce soit sur les ordinateurs de sites distants.

CHAPITRE IV

APPLICATION A UN CAS REEL

Dans ce chapitre on verra une application réelle à savoir la connexion d'un SOLAR-65 au sein de notre Centre de Calcul Réparti. On a pris comme système de référence, sur cette machine, le système en temps partagé TSM16 car il est le plus intéressant du point de vue de l'exploitation et tout ce qui a été fait pour lui reste valable pour la connexion de n'importe quel autre système pouvant être exploité sur un SOLAR. En particulier, la connexion du système RBOS/D a pu être réalisée très facilement suivant les mêmes principes.

Dans un premier temps, on examinera le système TSM16 et son moniteur d'entrées-sorties (IOCS) en mettant l'accent sur celles de ses fonctions qui nous ont paru intéressantes, pour notre réalisation, et sur les fonctions qu'il a fallu rajouter.

Ensuite, on verra comment ces fonctions ont été réalisées et en particulier un peu plus en détail la réalisation :

- de la gestion du protocole ordinateur connecté
- de la fonction d'interprétation
- de la gestion des terminaux distants

Enfin, nous formulerons un certain nombre de réflexions critiques à propos de cette réalisation et nous évoquerons les nouvelles démarches qui, grâce à l'expérience acquise, simplifieront les connexions ultérieures.

4.1. LE SYSTEME T S M

4.1.1. PRESENTATION

TSM16 (Time Sharing Monitor) (TECH-3) est un des systèmes fournis par le constructeur du SOLAR (SEMS) (TECH-1) en tant que système d'exploitation de celui-ci. C'est un système en temps partagé pouvant gérer actuellement jusqu'à 32 utilisateurs simultanément.

En outre, il dispose d'un moniteur d'entrées-sorties qui lui permet de gérer des périphériques du style lecteur de cartes, lecteur de rubans, imprimante, perforateur de rubans, bandes magnétiques, disques ... en plus des terminaux interactifs classiques. Un langage de commande est à la disposition des utilisateurs :

- pour se faire connaître du système (LOGIN)
- pour utiliser différents processeurs (il s'agit de processeurs logiciels)
- pour s'attacher (et libérer) les périphériques dont ils ont besoin
- pour se déconnecter à la fin d'une session (LOGOUT).

Un sous-ensemble du langage de commande (ensemble des commandes dites privilégiées) réservé exclusivement à un utilisateur particulier dit gérant du système ou encore opérateur, permet de contrôler l'état du système et son environnement et éventuellement de les modifier pour les besoins de l'exploitation.

En outre, TSM16 dispose d'un ensemble de requêtes accessibles par programme permettant de faire des entrées-sorties sur les périphériques de la configuration, de gérer des fichiers sur disques, de segmenter éventuellement des programmes importants en taille, etc...

D'un point de vue interne, TSM16 gère les différentes ressources physiques telles que mémoire centrale, unité centrale, périphériques, espace disques ... et logiques telles que processeurs, fichiers ... au profit des utilisateurs en assurant pour eux un partage de ces ressources avec un maximum de sécurité.

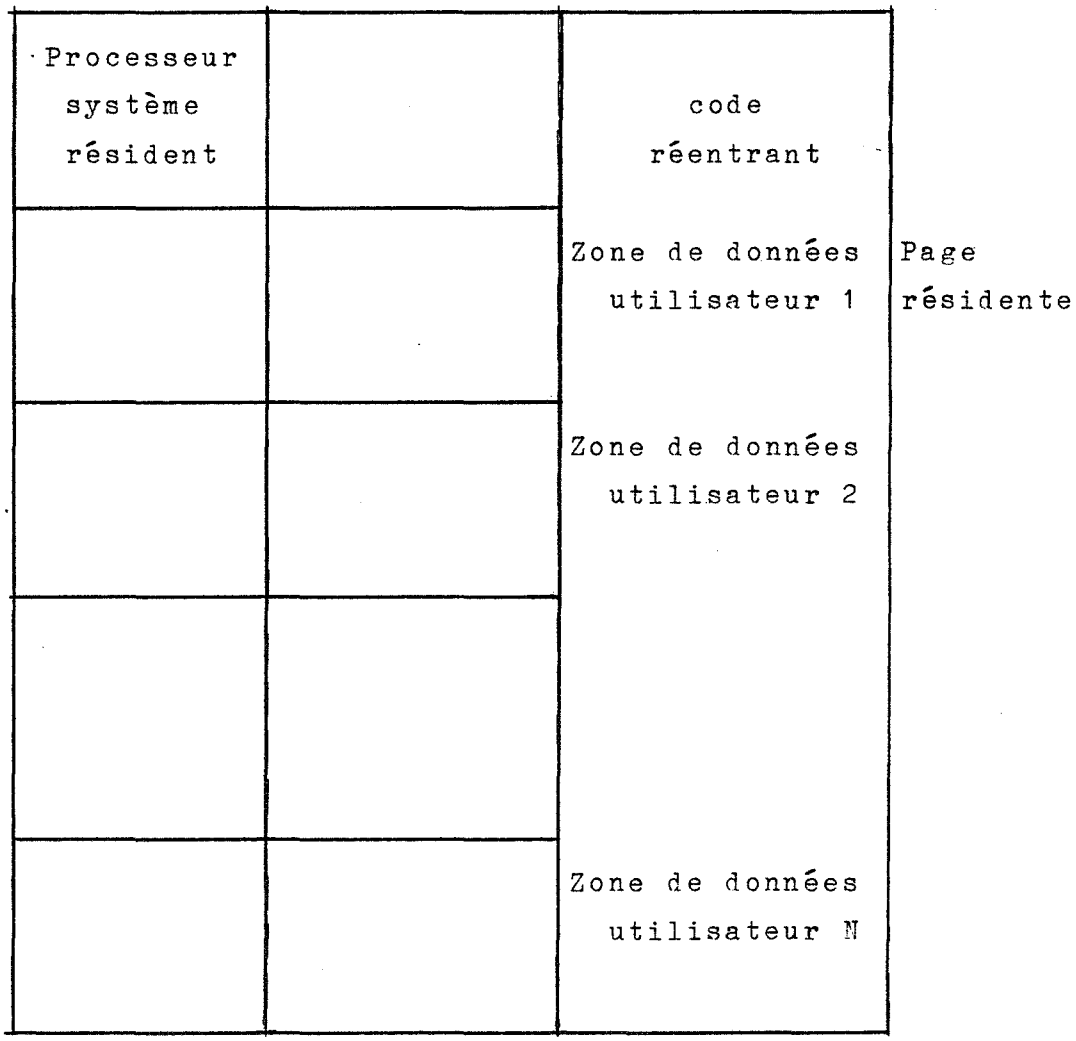
Voyons maintenant comment TSM16 assure le partage des différentes ressources de la machine.

Le partage de la mémoire d'abord est assuré par le gestionnaire de mémoire qui a comme unité de partage la page (zone de blocs de mémoire consécutifs). On distingue deux types de pages :

les pages résidentes allouées statiquement à la génération et de taille variable et les pages non résidentes qui ont toutes la même taille (définie à la génération).

Une page résidente contient un (ou une partie de) processeur et autant de zones de données qu'il est possible d'avoir d' utilisateurs simultanés utilisant ce processeur (Figure 4.1.).

Une page non résidente est allouée à un utilisateur, à un instant donné et contient son programme (utilisateur ou système) et ses données. Les pages de ce type font du va-et-vient entre la mémoire centrale et un disque (mécanisme du SWAP) si le nombre des utilisateurs actifs est supérieur au nombre de pages non résidentes (Figure 4.2.).



Exemple de page résidente de TSM16

Figure 4.1.

<table>
<tr><td>image
mémoire
processeur
système</td><td></td><td>Code
réentrant</td><td rowspan="2">Page
n° 1</td></tr>
<tr><td>Zone de
données</td><td></td><td>Swap</td></tr>
<tr><td>i.m. processeur
utilisateur</td><td></td><td>Swap</td><td rowspan="3">Page
n° 2</td></tr>
<tr><td>Zone de
données</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Zone
inutilisée</td></tr>
<tr><td>i.m. processeur
système</td><td></td><td>Code
réentrant</td><td rowspan="3">Page
n° 3</td></tr>
<tr><td>Zone de
données</td><td></td><td>Swap</td></tr>
<tr><td></td><td></td><td>Zone
inutilisée</td></tr>
</table>

Exemple de pages non résidentes de TSM16

Figure 4.2.

Le partage du temps de l'unité centrale se fait suivant un algorithme à deux niveaux. Il existe, en effet, deux files d'attente, une pour les utilisateurs en mode conversationnel (avec une tranche de temps allouée relativement petite), et une pour les utilisateurs en mode exécution (une tranche de temps beaucoup plus grande). Dans la première file sont mis les utilisateurs qui font une entrée-sortie console avant que leur tranche de temps ne soit épuisée, dans la seconde, les autres.

Tout utilisateur venant de se connecter se trouve sur la première file et y reste jusqu'à ce qu'il épuise sa tranche de temps sans faire d'entrée-sortie console. Inversement, dès qu'un utilisateur se trouvant dans la deuxième file fait une entrée-sortie console il passe dans la la première file.

Notons enfin qu'on ne choisit quelqu'un dans la deuxième file que si la première est vide.

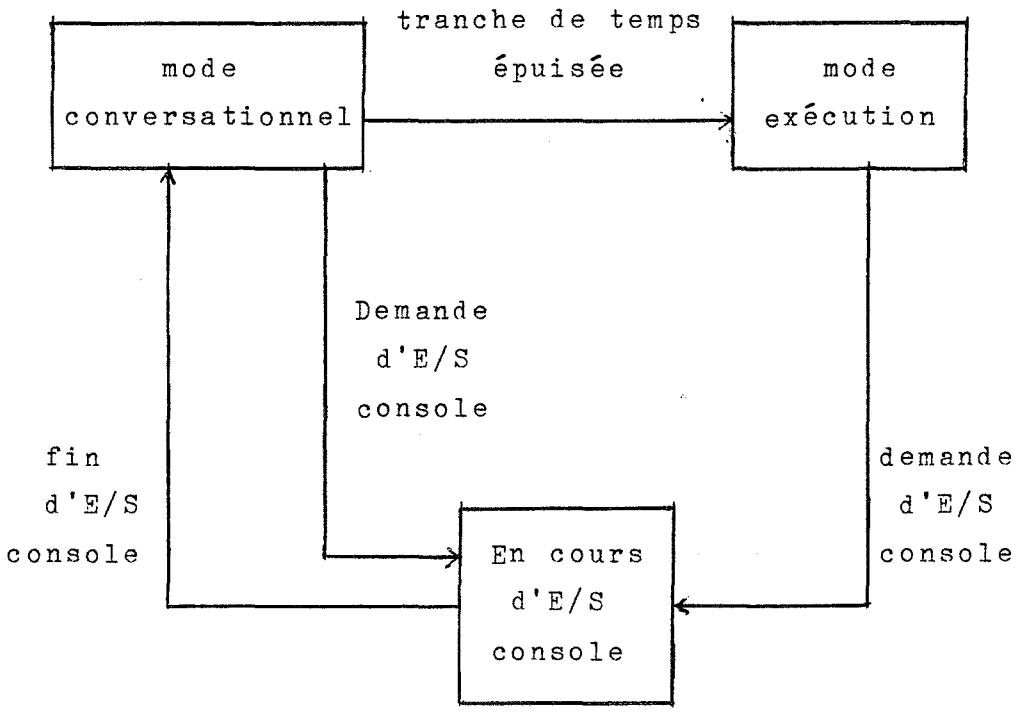


Figure 4.3.

Le partage des différents périphériques se fait à l'aide du moniteur d'entrée-sortie (voir 4.1.3.).

Enfin, le partage des fichiers est effectué à l'aide d'un moniteur de gestion de fichiers (FMS) (TECH-7, TECH-8) en utilisant, éventuellement, des catalogues (faisant partie du nom du fichier) permettant d'attribuer tel fichier à tel utilisateur.

Les appels par TSM16 aussi bien au moniteur d'entrée-sortie qu'au moniteur de gestion de fichiers se font par des appels au superviseur (instruction SVC) en passant comme paramètre un bloc de contrôle relatif à une entrée-sortie ou à une opération sur un fichier.

4.1.2. DECOMPOSITION FONCTIONNELLE

D'un point de vue fonctionnel, TSM16 est réalisé à l'aide d'un ensemble de processus qu'on peut classer dans deux catégorics :

- processus utilisateur ou processus console. A chaque utilisateur est associé au départ un tel processus qui le gère à travers son terminal.

- processus service ou processus système.

La règle de priorité entre les processus du système TSM16 est illustrée par la figure ci-dessous :

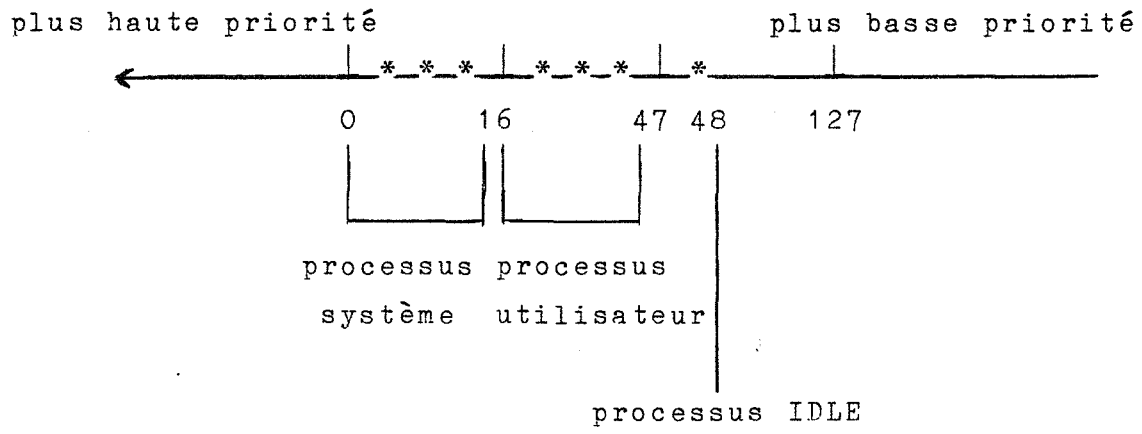


Figure 4.4.

Passons maintenant en revue les différents processus système :

INIT : sert à la réinitialisation du système en cas d'erreur grave.

SWAPOUT : sert à envoyer sur disque les données d'un utilisateur.

SWAPIN : opération inverse, dans le sens disque mémoire centrale.

SWAPRO : il s'occupe du va-et-vient entre le disque et la mémoire centrale en ce qui concerne les processeurs.

OVERLAY : il effectue les recouvrements (chargement des branches).

LOAD : il s'occupe du chargement d'un processeur pour le compte d'un utilisateur.

FINECH : c'est lui qui est réveillé à la fin d'une entrée-sortie. Il permet les entrées-sorties chainées : deux appels au moniteur d'entrée-sortie sans qu'il y ait de va-et-vient (SWAPIN).

SCHED : c'est lui qui s'occupe du "complément de scheduling" de telle sorte que tous les utilisateurs paraissent avoir la même priorité.

BREAK : il récupère les différentes break faites sur les consoles et initialise, s'il y'a lieu, un processus utilisateur.

SPOOL : il s'occupe des flots de sortie destinés à l'imprimante.

IDLE : c'est le processus de fonds de priorité moins élevé que tous les autres processus du système (activé quand il n'y a rien d'autre à faire).

Quant aux processus utilisateurs, on en associe un à chaque utilisateur au départ et il gère son terminal. On l'appelera aussi processus console.

Tous les processus du système, quand il s'agit de faire des entrées-sorties, que ce soit pour leur compte ou pour le compte d'un utilisateur (processus utilisateur), font appel au moniteur d'entrée-sortie (IOCS) qui gère le périphérique concerné à l'aide d'un DRIVER.

On remarque donc une certaine architecture en couche comme le montre le schéma ci-dessous :

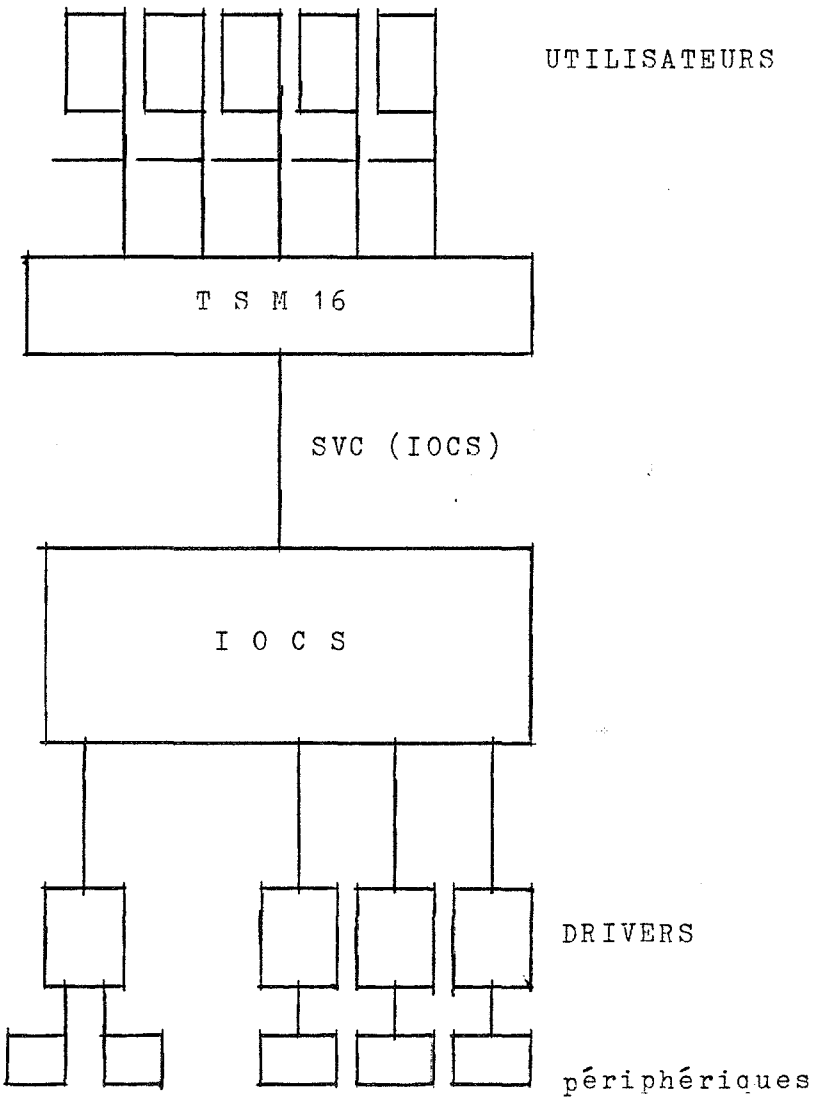


Figure 4.5.

4.1.3. MONITEUR D'ENTREES-SORTIES

IOCS (INPOUT OUTPUT CONTROL SYSTEM) (TECH-5,TECH-6) est le moniteur d'entrées-sorties utilisé par tous les systèmes pouvant être exploités sur un SOLAR. Il effectue l'initialisation et contrôle le déroulement de toutes les entrées-sorties effectuées par un système donné. Pour gérer les différents périphériques et lignes, il dispose des différents DRIVERS. Il s'agit de programmes (réentrants en général) nécessaires à la gestion des coupleurs (ligne ou périphérique). Un DRIVER peut, évidemment, dans certains cas, être utilisé pour gérer plusieurs coupleurs de même type.

L'interface de IOCS avec les différents DRIVERS est standardisé, à savoir qu'un DRIVER dispose d'un certain nombre de points d'entrée et de tables, et rend des compte-rendus bien définis. Ceci est valable pour tous les DRIVERS même si certaines de ces entrées sont fictives et ne sont jamais appelées.

De la même manière l'interface de chaque système qui utilise IOCS est standardisé. L'appel à IOCS se fait par appel au superviseur (instruction SVC). Le passage de paramètres se fait à l'aide d'un bloc appelé IOCB (INPOUT OUTPUT CONTROL BLOC) où on trouve notamment des informations comme :

* entrée ou sortie
* n° de l'unité sur laquelle se fait l'échange (unité fonctionnelle ou unité symbolique, on verra plus loin leur signification)
* l'adresse de la zone de stockage des informations lues ou écrites
* les comptes d'octet à émettre ou à recevoir ou l'adresse d'une table des codes d'arrêt si l'échange se fait sur code d'arrêt
* le compte-rendu du déroulement de l'échange (positionné par IOCS)

* le mode de retour du moniteur à l'appelant. Il en existe trois :

- retour en fin d'échange. L'appelant est suspendu jusqu'à la fin de l'entrée-sortie.

- retour après initialisation. L'échange est initialisé et le retour s'effectue tout de suite après, à la charge de l'appelant, d'attendre ou non la fin de l'entrée-sortie.

- retour après initialisation avec utilisation d'un pool de buffers.

- il existe un quatrième mode dit spécial qui n'est utilisé que par TSM16.

Après une initialisation de l'entrée-sortie, IOCS fait un retour pour que la mise en attente soit faite chez l'appelant. Ainsi, à la fin de l'entrée-sortie, le processus de l'utilisateur n'est pas réveillé automatiquement pour ne pas perturber le scheduling. A la fin de l'entrée-sortie, c'est toujours le même processus qui est réveillé (FINECH).

Pour rendre les programmes indépendant des opérations d'entrées-sorties, les notions d'unité physique, d'unité fonctionnelle et d'unité symbolique ont été introduites. Un périphérique est désigné par une unité physique. Une unité fonctionnelle désigne un sous-ensemble d'un périphérique tandis qu'une unité symbolique représente un flot d'informations en entrée ou en sortie et elle n'a pas une signification "physique".

Aux unités symboliques, on associe par programme des unités fonctionnelles. Ces liaisons sont dynamiques et permettent une indépendance des flots d'informations en entrée ou en sortie et des périphériques sur lesquels ils sont effectivement réalisés. On voit déjà une propriété intéressante du moniteur à savoir qu'un programme système ou un programme utilisateur,

quand ils font des entrées-sorties n'ont pas à connaître à priori les périphériques sur lesquels ils effectueront les échanges; ainsi le choix des périphériques peut être fait au dernier moment (avant l'exécution) par l'affectation SU-FU.

Outre les entrées-sorties proprement dites, IOCS effectue un certain nombre de fonctions, dites spéciales, qui ne s'accompagnent pas d'un transfert de données. Il en existe deux types :

- fonctions spéciales moniteur : ce sont des fonctions d'intérêt général dites standards qui nous permettent de réinitialiser complètement le système d'entrées-sorties, de lire le mot d'état d'un périphérique, de s'attacher un périphérique ou de s'en détacher, d'arrêter un échange avant sa fin normale, ou de réinitialiser une unité physique. En plus, des fonctions standards on a la possibilité de rajouter d'autres fonctions spéciales.

- Fonctions spéciales de positionnement : ce sont des fonctions particulières à un périphérique donné et réalisées au niveau du DRIVER. Chaque DRIVER dispose d'une entrée spéciale pour les fonctions de positionnement. Le numéro de la fonction demandée est interprété d'une façon particulière par chaque DRIVER.

En ce qui concerne notre application, le DRIVER réalisé possède, bien entendu, une entrée fonction spéciale qui nous a permis, entre autres, la récupération d'un IOCB de réception avant que l'entrée-sortie ne soit terminée. Ceci est nécessaire car si une réception est demandée par un utilisateur, elle ne se termine que si on a une réception valide de données venant de l'ordinateur d'en face, il faut donc pouvoir, éventuellement, l'arrêter avant. Autrement dit il s'agit d'annuler une réception en cours.

Une autre fonction spéciale moniteur qui nous a intéressés particulièrement est la réinitialisation complète d'une unité physique. Quand IOCS détecte une telle fonction, il fait une réinitialisation de ses tables concernant cette unité physique et ensuite appelle le DRIVER par une entrée spéciale pour qu'une réinitialisation de l'unité physique soit faite par lui. On s'est servi de cette entrée pour effectuer la réinitialisation du processus du DRIVER qu'on a réalisé (voir 4.2.1.).

Enfin, la fonction spéciale moniteur KILL, qui met fin à un échange se traduit par l'appel de l'entrée correspondante du DRIVER. Pour notre application, cette entrée regroupe les travaux suivants :

Retirer du contexte un IOCB en attente d'acquittement s'il s'agit d'une sortie ou récupérer l'IOCB de réception s'il s'agit d'une entrée. Dans les deux cas, un compte-rendu négatif est positionné dans l'IOCB.

4.2. INTEGRATION DU LOGICIEL DE CONNEXION

Au chapitre II, on a examiné une certaine méthodologie à suivre quand il s'agit de connecter deux ordinateurs. Ici, on verra de quelle manière on l'a appliquée pour connecter le SOLAR (son système TSM16 en l'occurence) à l'ordinateur FRONTAL de notre Centre de Calcul Réparti.

4.2.1. REALISATION

Le problème a été abordé par étapes.

La première qui a été réalisée, était la gestion du protocole, ordinateur connecté sur le frontal. Après les premiers tests effectués en bouclage et dès l'arrivée du SOLAR, on a voulu faire une première validation du logiciel de connexion dans un cas particulier, à savoir la création d'un utilitaire de transfert de fichiers entre T1600 et SOLAR. Chose d'autant plus intéressante, d'un point de vue pratique, dans la mesure où le SOLAR n'ayant aucun périphérique d'entrée d'informations (pas de lecteur de cartes ni de rubans) en dehors d'une console opérateur et des disques à cartouche (non compatibles avec les cartouches du T1600) cela devait nous permettre d'avoir une manière rapide de lui introduire des programmes : soit rentrés par le lecteur du T1600 et ensuite transférés, soit se trouvant sur la machine dorsale du centre (PHILIPS) en faisant deux transferts (un utilitaire de transfert PHILIPS-T1600 existait déjà). Voir configuration du Centre de Calcul en Annexe.

Pour réaliser donc cet utilitaire de transfert, on a choisi comme système de base le système RBOS/D car c'est un système conçu pour la production des programmes et il présente plus de facilités pour leur mise au point. D'autre part, son aspect monoutilisateur simplifiait certains problèmes de synchronisation dans un premier temps.

Sur le T1600 (TECH-2) on est parti d'un système RBOS/D (TECH-4) également pour la réalisation de cet utilitaire, car la version à l'époque du logiciel du frontal ne disposait pas des primitives concernant les fichiers.

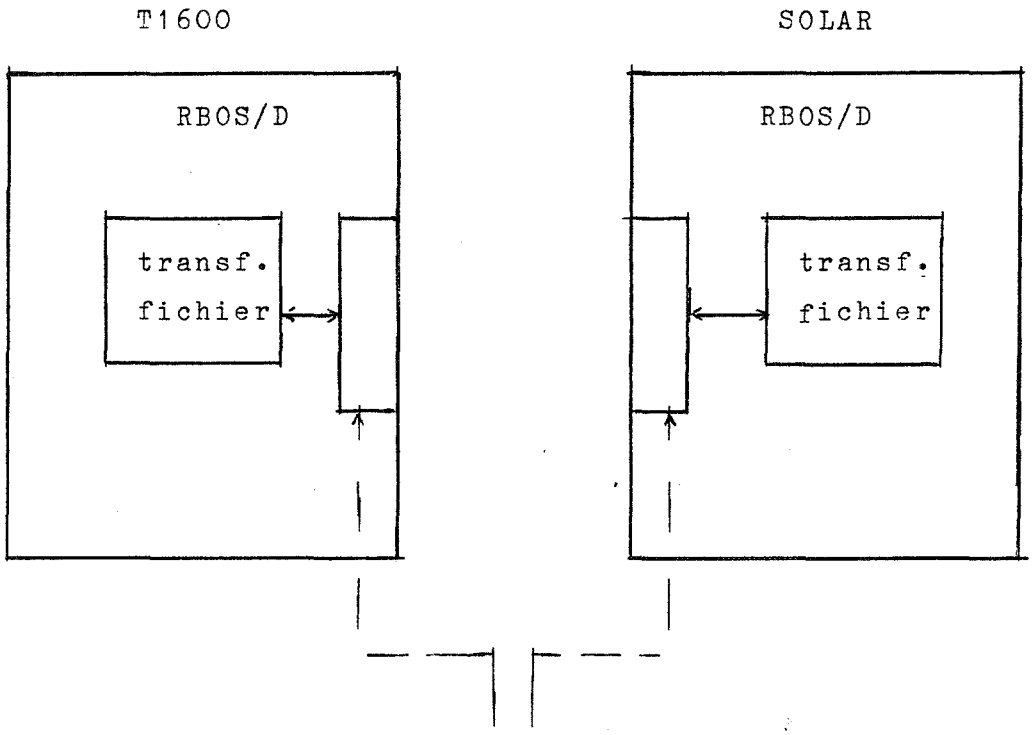

T1600
SOLAR
RBOS/D
RBOS/D
transf.
fichier
transf.
fichier
Logiciel de connexion

Figure 4.6.

Une première version de cet utilitaire de transfert des fichiers a vu le jour avec une gestion du protocole ordinateur connecté fortement inspiré du logiciel réalisé déjà sur le frontal. D'autre part, les deux processus de transfert de fichiers proprement dits étaient les mêmes sur les deux machines (Figure 4.6.).

Dans cette première version, le logiciel de connexion était introduit en tant que tâche temps réel d'application utilisateur utilisant des processus utilisateurs.

L'intérêt de cette façon de faire était double : d'abord on n'a pas du tout à modifier le système d'exploitation, donc la mise au point a pu être faite à l'aide d'un système dont on était sûr du fonctionnement. Ensuite, le fait que l'utilitaire utilisait des processus utilisateurs qui travaillaient en parallèle avec le processus du superviseur nous permettait de faire en même temps les traitements habituels qu'on faisait sous RBOS/D. Notons aussi que du fait que les deux systèmes avaient le même "J.C.L.", aucune traduction n'était nécessaire.

La deuxième étape a été la définition des spécifications de réalisation d'un utilitaire "ECHO" très simple qui avait plusieurs actions :

- -renvoi de n copies des messages reçus (n étant un paramètre de commande),
- -possibilité de blocage et déblocage par émission de "BREAK".

Il a été réalisé, sur le SOLAR toujours sous RBOS/D, en tant que processeur utilisateur, qu'on appelera désormais service parce qu'il s'agissait du premier "service réseau" réalisé au sens du chapitre I.

L'intérêt de cette réalisation reposait surtout sur le fait que des problèmes d'interprétation faisaient leur apparition et qu'ils ont été résolus au niveau du service par le service lui-même. C'est à dire que le service "comprenait" le LCI donc l'enveloppe réseau était vide.

Tandis que les messages transitaient de la même manière que pour le transfert des fichiers, une partie de leur contenu supportait le LCI qui arrivait jusqu'au service qui l'interprétait. C'est-à-dire que le logiciel de connexion était utilisé en tant que service de transport et la signification des messages transportés étaient explorés par le service lui-même.

L'étape finale a été l'intégration du logiciel de connexion dans TSM16 non plus comme processeur utilisateur mais en tant que partie fonctionnelle du système. L'aspect du problème était, ici, différent que précédemment en ce sens que la connaissance du système (TSM16) d'un point de vue fonctionnel devenait indispensable car, en lui rajoutant des éléments, on était amené à le modifier.

A ce propos, il serait bon de souligner les discussions très intéressantes et constructives qu'on a eues avec les gens du département du développement du constructeur et qui ont abouti à une modification légère de la version suivante de leur système, qui consiste à laisser un processus très prioritaire disponible, dans le pool de processus et qui était à la fois indispensable à notre réalisation,et pour considérer la possibilité d'un frontal ou d'un réseau pour le constructeur.

Voyons maintenant, rapidement, les différents problèmes que l'on a eu à résoudre. Le système TSM16 n'étant pas un système de production de programmes n'avait pas de moyen de créa tion et de destuction dynamique de processus.

Il a fallu donc créer de toutes pièces la gestion d'un pool de processus qui utilisait le processus de priorité laissé "libre" par le système, pour pouvoir créer les processus dont on avait besoin pour notre réalisation.

D'autre part, on a vu au paragraphe 4.1.1. que TSM16 gère la mémoire centrale en ayant comme unité de mémoire la page, quantité trop grande à notre niveau. Notre unité étant de quelques dizaines de mots servant pour des tempons et des contextes dynamiques.

On a donc été amené à réaliser une gestion mémoire de petits blocs. Pour ce faire on s'est réservé une certaine quantité de mémoires pouvant contenir, à la fois, la mémoire nécessaire pour un certain nombre d'utilisateurs en se disant qu'il n'y a qu'un certain nombre d'utilisateurs actifs à un instant donné. D'autre part, cette mémoire doit être prise en mémoire centrale et doit être résidente en permanence puisqu'elle contient les tempons d'entrées-sorties.

Une autre difficulté rencontrée lors de l'introduction de notre logiciel sous TSM16 a été le fait que pour réaliser la gestion des REVEILS on ne pouvait pas prendre celle existante sous RBOS/D qui utilisait un processus de gestion d'horloge permettant la suspension d'un processus pour un certain délai, car elle n'existait pas sous TSM16.

On verra, par la suite, comment on a résolu ces problèmes. Disons simplement que dans un premier temps si le nombre d'utilisateurs distants est réduit, on pouvait fonctionner sans REVEIL car la probabilité de perte des paquets était quasiment nulle, il n'y avait donc pas à faire de réémission des paquets.

Cas de visualisation de l'espace mémoire (DUMP). Chaque processus utilisateur s'exécutant en mode esclave, on n'a pas la possibilité de visualiser de la mémoire en dehors de celle qui appartient à son espace d'exécution. Entre autre, on n'a pas la possibilité de visualiser des zones mémoire "système".

Pour palier cet inconvénient, on a été obligé d'introduire deux nouveaux programmes superviseurs (nouveaux SVC) qui permettent le passage du mode esclave en mode maître et l'inverse. Il est bien évident qu'une possibilité comme celle-là ne sera pas à la disposition de l'utilisateur courant et qu'elle servira seulement au "sysman" pour des essais en "ligne". D'ailleurs, dès la fin de la mise au point, elle disparaîtra probablement.

Enfin, une trace a été fabriquée pour tracer tous les IOCB et les buffers qui sont passés au DRIVER DRVSP, ce qui nous permet une trace au niveau protocole. La réalisation sur le FRONTAL de la même trace existe mais en plus on en dispose d'une au niveau des octets.

Voyons maintenant à quel niveau hiérarchique (voir 2.1.) on a introduit notre logiciel de connexion. On a vu que TSM16 a une vision standardisée de tous les périphériques et que, pour faire des entrées-sorties, il s'adresse au moniteur d'entrées-sorties (IOCS) d'une façon unique. En particulier, pour effectuer des entrées-sorties pour le compte des utilisateurs distants il suffisait d'introduire un nouveau DRIVER (nommé DRVSP) qui gérait les voies des utilisateurs distants.

Ainsi, les interfaces TSM16 et IOCS, d'une part et IOCS et DRIVER, d'autre part, n'étaient pas du tout affectées (voir 2.1.), à la charge du DRIVER d'effectuer le multiplexage et le démultiplexage des utilisateurs distants sur la ligne qui les reliaient au reste de la communauté en utilisant, entre autres, le DRIVER constructeur conçu pour gérer la ligne.

Pour ce faire, il a fallu autant d'unités physiques "fictives" que d'utilisateurs distants potentiels, chacune de ces unités étant gérée par DRVSP. On a repris une structure similaire à celle utilisée pour gérer un coupleur multiplexé (MUX), où on a une unité réelle (RACINE) et des unités fictives pour chaque voie, avec un ensemble de tables pour chaque élément.

Cependant, la similitude n'est pas complète : toutes les unités physiques sont sur un niveau d'entrée-sortie fictif (il n'y aura pas d'interruptions réelles d'entrées-sorties) tandis que pour un MUX la racine correspond à un niveau d'entrée-sortie réel. En fait, la fin d'une entrée-sortie survient à l'arrivée de l'acquittement d'un message émis et à la fin d'une réception valide sur la ligne. Donc, pour le niveau supérieur, le mécanisme des acquittements et la fin d'une réception valide sont pris comme les véritables interruptions de fin d'entrées-sorties.

4.2.1.1. Gestion du protocole ordinateur connecté

Les spécifications internes et externes ayant été déjà données au paragraphe 3.1.2. du chapitre précédent, ici, on verra l'aspect fonctionnel défini lors de la réalisation.

Une partie du traitement (les émissions) pouvant se faire au coup par coup, c'est-à-dire à la demande, il existe une autre partie qu'on est obligé d'effectuer en différé. C'est le cas de traitement de REVEILS et de fin de réception sur la ligne. Pour ce traitement en différé, on est donc obligé d'utiliser un processus spécial qui ne fait que cela.

En plus, il faut qu'il se mette en réception sur la ligne, le plus rapidement possible, afin de minimiser la perte de caractères venant de la ligne. Donc, le traitement effectué par celui-ci doit être très court d'une part et d'autre part, avoir une priorité assez élevée par rapport à la plupart des autres processus du système.

C'est la priorité de ce processus qui a été fixée en accord avec le constructeur (Ch. 2.1.). Ce processus traite aussi les commandes (voir 3.1.2.) d'ouverture et de fermeture.

Une fois l'ouverture du service effectuée, les utilisateurs peuvent envoyer et recevoir des informations à l'aide des IOCB. Un IOCB d'émission est émis tout de suite, en ajoutant l'en-tête relatif au protocole, si des crédits d'émission le permettent. Ensuite, il est stocké dans le contexte, dans une file en attente d'acquittement et un REVEIL est armé.

Si l'acquittement arrive avant le déclenchement du REVEIL, le REVEIL est désarmé et l'émission terminée (IOCB retiré de la file et évènement fin d'émission enclenché), sinon une réémission aura lieu, suivie d'un nouvel armement de REVEIL. Si l'acquittement n'arrive pas à nouveau, on recommence un certain nombre de fois avant de déclarer l'envoi impossible, par compte-rendu négatif au niveau supérieur.

Si l'on ne dispose pas de crédits d'émission, l'IOCB est passé directement (sans être effectivement émis) dans la file d'attente d'acquittement et un REVEIL est armé afin qu'il soit émis par le jeu de réémission en espérant que des crédits seront alloués, entretemps.

Les crédits et les acquittements sont inclus soit, dans des paquets acquittement, soit dans l'en-tête de lettre (ch. 3).

La fin d'une entrée-sortie, pour un utilisateur donné, survient :

- en ce qui concerne les émissions, lors de la réception d'un acquittement signalant que le paquet est bien arrivé (fin d'émission normale) ou après un certain nombre de tentatives d'envois sans réponses positives, c'est-à-dire pas d'acquittement (fin d'émission anormale).

- en ce qui concerne la réception, lors de la réception valide d'un paquet et destiné à un utilisateur donné. En fait, la réception fonctionne de la façon suivante. Un utilisateur qui veut recevoir fait une demande de réception à l'aide d'un IOCB noté IOCBU. L'IOCB en question est stocké dans le contexte et des crédits sont alloués :

  dès réception chez lui et d'émission en face.

  Par ailleurs le processus de réception est toujours en réception sur la ligne à l'aide d'un IOCB noté IOCBR. A la fin d'une réception, sur la ligne, et après diverses vérifications, les informations reçues sont recopiées du buffer de l'IOCBR dans celui de l'IOCBU. Et à ce moment, la réception est déclarée "terminée pour l'utilisateur".

Voyons maintenant comment a été résolu le problème des REVEILS. Quand un processus (A) arme un REVEIL, il souhaite être avertie après un certain délai sans obligatoirement être contraint de s'interrompre pendant ce temps là. Voyons maintenant comment on peut réaliser cela à l'aide d'un processus (B) de traitement de REVEILS et d'une primitive de suspension d'un processus pendant un délai déterminé.

Le processus qui veut armer un REVEIL appelle une procédure qui active le processus (B) en lui passant comme paramètres : l'adresse du semaphore sur lequel il veut être averti, un paramètre (P) éventuellement et le délai (T). Ces informations sont stockées dans une table des REVEILS et au retour un compterendu à l'appelant est l'index dans la table de REVEILS et qui servira pour le désarmement éventuel de ce REVEIL.

Le processus (B) se suspend pendant le délai (T) à l'aide de la primitive de suspension. Passé ce délai, il va regarder dans la table de REVEILS à l'aide d'un indicateur si le REVEIL n'a pas été désarmé entretemps, si c'est le cas, il active le processus (A) avec, éventuellement, comme paramètre (T).

Avec ce système, il est clair que si un deuxième REVEIL est armé par un autre processus avant la fin du premier, celui-ci aura comme durée le délai désiré plus une partie du délai du premier car pour prendre en compte le deuxième, il faut que le temps du premier soit écoulé. Cet inconvénient n'est pas d'une importance majeure car ce qu'il faut surtout c'est d'être averti au bout d'un certain temps. Peu importe si ce temps est un peu plus long que ce que l'on a prévu au départ ; en plus, s'il n'y a qu'un seul processus qui arme un REVEIL, le problème ne se pose pas.

TSM16 ne dispose pas d'une requête qui permette à un processus de se suspendre pendant un délai donné. Par contre, il dispose d'une requête qui permet d'effectuer des entrées avec en paramètre un délai au bout duquel, si l'entrée n'est pas terminée, elle est interrompue.

Le processus de traitement des REVEILS se sert de cette requête pour être suspendu pendant un certain délai en demandant une entrée sur une unité fictive. Ainsi, l'entrée en question ne peut être terminée que sur délai épuisé, et le processus se trouve suspendu pendant ce délai.

L'inconvénient de cette solution est que l'unité fictive doit être connue d'IOCS et de TSM16 donc un certain nombre de tables lui sont allouées, ce qui implique une perte de place. Un avantage cependant, non négligeable, est le fait que l'on n'a pas eu à modifier le processus horloge de TSM16.

4.2.1.2. Interprétation

Pour se faire connaître du système TSM16, un utilisateur se présente devant un terminal (connecté physiquement au SOLAR) et fait un BREAK. Ceci provoque l'initialisation et le lancement d'un processus utilisateur par le système. Il est bien évident que pour un terminal local, le BREAK est directement capté par le DRIVER gérant le coupleur de ce terminal et répercuté par celui-ci au système TSM16. Pour un terminal qui ne lui est pas connecté physiquement, les choses devraient se passer d'une manière similaire.

Dans ce cas, comme le système n'a pas un contrôle direct du terminal, il faut que le logiciel de connexion lui fasse parvenir la même action que celle que le DRIVER lui aurait répercuté, s'il s'agissait d'un terminal local.

Dans le contexte CCR, l'évènement qui provoque cette action est la réception d'une demande de connexion. On a vu l'existence d'un processus qui était en permanence à "l'écoute" de la ligne. C'est lui qui, en analysant ce qui vient de la ligne, détectera une demande de connexion. C'est donc à lui de prévenir le système TSM16. Voyons maintenant comment cela est réalisé :

A la réception d'une demande de connexion, on choisit une voie fictive parmi celles destinées aux utilisateurs distants potentiels (voir 4.2.1.) et on la marque occupée Ensuite, on avertit TSM16, en particulier le processus qui est en attente de BREAK en lui passant un mot d'état avec le bit BREAK positionné, concernant la voie fictive choisie. Donc, pour TSM16, tout se passe comme si un BREAK lui était parvenu par un terminal local, c'est-à-dire lui ne fait pas la différence d'un BREAK provenant d'un terminal local ou distant, il connaît un ensemble de terminaux qui est constitué indifféremment de terminaux locaux ou non.

Enfin, on dispose d'une table (VOIDUTI) de correspondance entre le numéro de la voie fictive et l'identification de l'utilisateur connecté sur cette voie. L'identification de l'utilisateur est introduite dans la table au moment du choix de la voie fictive. On verra par la suite l'utilité de cette table, pour l'interprétation.

TSM16 ne connaît que des voies et il fait ses entrées-sorties à l'aide des IOCB d'émission et de réception sur ces voies, en particulier, il ne connaît pas l'identification de l'utilisateur (au sens LCI).

Par contre, du côté du FRONTAL, on ne connaît que des utilisateurs identifiables à l'aide d'un numéro et pour chacun d'eux, l'émission de messages vers un service se traduit par la fabrication d'un message LCI qui, dans son en-tête contient, entre autres, l'identification d'un utilisateur (voir annexe concernant LCI). Bien entendu, la notion de voie au sens TSM16, est ici inconnue.

La traduction donc de l'identification, en numéro de voie, et l'inverse est indispensable. On a choisi de la faire sur le SOLAR pour ne pas remettre en cause le choix de la localisation de l'interpréteur.

L'interpréteur, fait donc partie du DRIVER spécial (DRVSP) et a deux fonctions différentes :

- interpréter le LCI quand on a une réception valide qui se termine. En particulier, repèrer la voie fictive qui est associée à un utilisateur à partir de son identification et à l'aide de la table VOIDUTI.

- fabriquer le LCI lorsqu'on initialise une émission. Pour ce faire, on se sert de la table VOIDUTI pour récupérer l'identification de l'utilisateur à partir du numéro de la voie fictive qui lui est associée et d'une autre table qui nous donne, à partir du numéro d'identification, le numéro de la SU (ce qui correspond à un numéro d'accès utilisateur) associé à son terminal. Cette table est remplie au moment de l'arrivée d'une demande de connexion (identification utilisateur et SU origine faisant partie du LCI).

Du point de vue code opération de la commande (champs CAT et FONCT) on peut dire qu'un service peut envoyer :

- du point de vue contrôle de connection : une demande de déconnexion ou une demande de suspension ou encore un détachement d'appareil. En effet, la connexion à un service se fait toujours à l'initiative d'un utilisateur.

- du point de vue contrôle de dialogue : un service n'envoie pas des attention. Le "à vous" fait partie du contrôle de dialogue mais il est passé dans le champ FONCT des commandes d'échanges.

- du point de vue commandes d'échanges : c'est la quasi totalité des messages envoyés par un service. En effet, le passage de l'information se fait par les commandes d'échanges. Le champ FONCT, dans ce cas, détermine le "à vous", un bit secret (pour pouvoir masquer les entrées éventuellement) et le niveau.

Résumons maintenant les opérations à faire à la demande d'une connexion :

- on choisit une voie fictive et la marque occupée.

- on remplit les tables de correspondance numéro de voie - identification utilisateur - SU origine.
- on simule un BREAK pour la voie fictive.
- on attend le message d'initialisation de TSM16 sur la voie, en refusant tout ce qui vient de l'utilisateur hormis une demande de déconnexion.
- "la main" est passée à l'utilisateur avec l'envoi du message.

Enfin, à propos de l'ouverture de la SU (voir chapitre 3.1.2) elle se fait au moment de la mise en route du service (par la commande d'ouverture de la SU du service en provenance de "l'opérateur du CCR").

4.2.2. BANALISATION DES TERMINAUX

On a vu au chapitre I qu'un des objectifs du centre de calcul réparti était de dégager l'utilisateur des soucis "d'appartenance" des terminaux à tel ou tel ordinateur. Il est souhaitable pour un utilisateur de s'installer devant un terminal et de demander la connexion à un service accessible quelconque sans qu'il ait à se déplacer physiquement (changer de terminal) chaque fois qu'il veut changer de service sous prétexte que ce dernier se déroule sur un autre ordinateur. En particulier, on souhaite qu'un utilisateur voulant travailler avec TSM16 ne soit pas obligé de chercher un terminal qui soit connecté physiquement au SOLAR mais qu'il puisse travailler à partir de n'importe quel terminal du centre.

Inversement, on souhaite qu'à partir d'un terminal connecté physiquement au SOLAR, on puisse travailler avec n'importe quel service offert par le centre de calcul réparti. En particulier, on doit pouvoir demander la connexion à TSM16 une fois connecté au C.C.R. (on ne se connecte pas directement à TSM16 à l'aide d'un BREAK) car on souhaiterait que l'utilisateur n'ait pas à connaître s'il s'agit d'un terminal local ou non au SOLAR. On voudrait qu'il se présente devant son terminal, qu'il fasse Login (au sens CCR) et qu'il demande la connexion à TSM16 dans tous les cas.

Bien entendu, on prévoit, dans le cas où à partir d'un terminal local au SOLAR on demande la connexion à TSM16, de faire passer les données directement du terminal à TSM16 afin d'éviter que les données transitent vers l'extérieur inutilement.

Les terminaux connectés physiquement au SOLAR deviennent donc des terminaux distants pour le C.C.R. au sens du paragraphe 3.1.3. On a vu qu'à chaque terminal distant on associait une SU de la même manière que pour un service. L'ouverture d'une telle SU (au sens 3.1.2.) se fait à l'initiative de l'opérateur du CCR.

Un terminal local physiquement connecté au SOLAR le reste aussi logiquement à TSM16 tant qu'une telle ouverture n'est pas faite. Ensuite, une fois qu'il a été ouvert comme terminal distant par commande de l'opérateur CCR, il faut le déconnecter logiquement de TSM16 (en empêchant les BREAK émis par lui d'arriver directement à TSM16). Par contre, à la fermeture d'un terminal distant pour le C.C.R. il faut le reconnecter logiquement à TSM16.

En outre, un système d'indicateurs doit nous permettre l'interdiction de l'ouverture en tant que terminal distant d'un terminal qui est en train de travailler en tant que terminal local.

Cet aspect de banalisation des terminaux, qu'il s'agisse des terminaux locaux ou distants, est tout à fait vrai si tous les terminaux ont les mêmes fonctions physiques, condition rarement remplie si les différents terminaux ne proviennent pas du même constructeur. Par contre, la vision que le système a de tous les terminaux qui lui sont logiquement connectés, qu'il s'agisse de terminaux locaux ou de terminaux distants, doit être la même. Ce qui nous amène à introduire la notion du terminal virtuel (CCR-5,RES-2).

Enfin, en ce qui concerne notre application, pour fixer les idées, on peut donner 20 comme nombre de terminaux connectables logiquement à TSM16, compte tenu de la configuration actuelle du SOLAR. Les 12 sont localement connectés à SOLAR et les 8 parmi les terminaux connectés extérieurs (distants).

Par ailleurs, le nombre des utilisateurs simultanés de TSM16 (configurable) est, pour le moment, fixé à 12 car s'il y a plus de 12 utilisateurs le temps de réponse devient inacceptable. Cela veut dire qu'on aura comme utilisateurs maximum :

- soit 12 locaux et aucun distants.

- soit 8 distants plus 4 locaux.

- soit toute combinaison intermédiaire.

4.3. SI TOUT ETAIT A REFAIRE

Comme on l'a vu au paragraphe 2.1 un de nos soucis principal était de modifier le moins possible le système à connecter. Sans véritablement mettre en cause ce principe, l'expérience acquise lors de la mise au point nous a montré qu'en réalisant des modifications un peu plus profondes du système, sans pour autant changer sa logique, on gagnerait en performance.

Pour faire de telles modifications, la connaissance profonde du système en entier est indispensable et malheureusement cette connaissance est souvent acquise une fois que la première réalisation est effectuée.

Par exemple, en ce qui concerne notre réalisation, l'intégration dans le système d'une gestion dynamique de la mémoire, et en particulier la possibilité de modifier dynamiquement le nombre de pages non résidentes en mémoire centrale nous auraient permis :

- d'abord de ne charger le logiciel de connexion qu'à son démarrage (ouverture de la SU), en réalisant une structure de recouvrement. Ainsi, le système voyait ses performances augmenter grâce à l'augmentation de la mémoire disponible, si la liaison n'était pas en service pour une raison quelconque.

- Ensuite, et surtout, le fait que la mémoire pour les utilisateurs (accédant à distance) était acquise dynamiquement, pourrait nous permettre, dans le cas où l'on a peu d'utilisateurs de ce type actifs, d'avoir une page non résidente supplémentaire à la disposition du système.

Une autre modification du système qui nous semble bénéfique au niveau des performances est l'intégration de la gestion des REVEILS dans le processus horloge de TSM16. On aurait gagné les activités d'un processus (des REVEILS) et les tables associées à une unité fictive (voir 4.2.1.1). En plus, on aurait offert aux utilisateurs et aux processeurs une possibilité supplémentaire : l'accessibilité des réveils.

Voyons maintenant quelques autres modifications ou ajouts qu'on aurait pu effectuer au départ et qui nous auraient aidés aussi bien à la conception qu'à la mise au point.

On aurait du reprendre le DRIVER qui gérait la ligne (DRVVT) vers le reste de la communauté, en lui ajoutant la gestion d'une procédure, ou au moins les fonctions d'une procédure dont on avait besoin : contrôle d'erreurs sur un message, possibilité de transmettre des textes transparents. En effet, ainsi on éliminait d'emblée le problème de rapidité des réceptions (voir 4.2.1.1) et en plus on était sûr de l'exactitude du contenu des messages transportés.

On a vu que le DRIVER DRVSP qui gérait les utilisateurs non locaux à TSM16 se servait du DRIVER DRVVT pour envoyer et recevoir sur la ligne. En déportant les fonctions de procédure sur ce dernier, on confiait les problèmes dûs au transfert sur la ligne à la gestion de cette dernière.

La modification du DRIVER DRVVT aurait pu constituer la première étape de notre réalisation. Pour la mise au point de cette étape, on peut citer comme outil indispensable une trace au niveau de l'octet. Sa conception devrait se faire d'ailleurs en même temps que l'intégration de la procédure au DRIVER car celui-ci est le mieux placé pour réaliser une trace au niveau de l'octet. Une telle trace pourrait nous être très utile pour la suite de notre réalisation.

CHAPITRE V

CONCLUSION

Notre étude s'inscrit dans le domaine de connexion d'un ordinateur à un réseau, avec la particularité de nous limiter volontairement à des réseaux locaux. Plusieurs problèmes pouvant ainsi être particularisés (routage par exemple) entrainant une réduction du logiciel de connexion et la possibilité donc de sa réalisation sur des mini-ordinateurs.

Tout au long de cette étude, on a essayé de démontrer que le problème d'intégration d'un ordinateur, au sein d'un réseau local, peut être résolu d'une manière presque systématique en suivant des étapes bien déterminées.

En plus de la méthodologie ainsi définie, on a examiné les techniques utilisées pour réaliser les différentes étapes. Enfin, on présente l'application de ces principes, dans un cas concret, à savoir la réalisation de la connexion d'un nouvel ordinateur au sein d'un réseau local. Reste à voir le prolongement futur de notre réalisation.

Pour ce faire, on commencera d'abord par examiner l'état actuel des travaux, de l'ensemble du Centre de Calcul Réparti car, ne l'oublions pas, notre étude se place dans le cadre de ce projet.

Actuellement, l'architecture du FRONTAL, telle qu'elle a été présentée au chapitre I, est opérationnelle. En particulier, les différents accès utilisateurs, le correspondant et l'interprète sont fonctionnels afin qu'un certain sous-ensemble de commandes LCE soit réalisé (le reste des commandes se réalisera facilement à cause de la modularité de l'interpréteur).

En ce qui concerne les différentes liaisons du FRONTAL avec les autres utilisateurs du centre de calcul réparti, on peut dire que, outre la liaison avec le SOLAR, les liaisons suivantes sont fonctionnelles :

- T1600 - PH1175 : liaison en mode synchrone avec une procédure ECMA Philips. Un protocole de transfert de fichier existe.

- T1600 - PDP : liaison en mode asynchrone avec une procédure TMM. Un service ECHO fonctionne sur le PDP.

- T1600 CYCLADES : liaison qui nous permet l'accès à tous les services CYCLADES.

Mais revenons sur la connexion T1600 SOLAR. D'abord, un protocole de transfert de fichier nous permet le transfert des fichiers (simultanément dans les deux sens). Ensuite, le système RBOS/D du SOLAR est accessible par le T1600 et vis versa. En particulier, l'initialisation à distance d'un des deux systèmes est possible. Citons aussi un utilitaire ECHO qui fonctionne également sur le SOLAR aussi bien sous le système RBOS/D que sous TSM16.

Enfin, la mise en exploitation du système TSM16 via le CCR, actuellement en fin de la phase de mise au point, nous permettra, entre autre, la mise en exploitation du service APL via le CCR.

Reste la mise en exploitation des terminaux distants (par rapport au CCR donc locaux au SOLAR) dont la gestion bien qu'existante, n'a pas pu encore être testée. Il s'agit donc d'une perspective à très court terme.

Voyons maintenant quelques autres perspectives et développements ultérieurs qui nous semblent bénéfiques non seulement à notre réalisation mais aussi à l'ensemble du centre de calcul réparti.

D'abord, en ce qui concerne le transfert des fichiers, l'utilitaire actuel nous permet le transfert d'un seul fichier à la fois dans un sens donné (et un autre dans l'autre sens). Le transfert de plusieurs fichiers, simultanément, pourrait être envisagé moyennant de légères modifications, car la structure du réseau de transport le permet (existence d'un multiplexage/démultiplexage).

Ceci est nécessaire à un développement qui se fait actuellement pour le CCR et qui consiste à la réalisation d'un système de coopération de systèmes de gestion de fichiers (CCR-9) et qui entre dans un des domaines les plus importants de l'informatique répartie à savoir la gestion des bases de données réparties.

L'utilisation aussi de cet utilitaire sous TSM16 sera possible car bien qu'ayant été conçue pour travailler sous RBOS/D, les interfaces de RBOS/D et de TSM16 vis-à-vis du moniteur d'entrées-sorties, étant identiques, on pourra l'utiliser dans les deux cas.

Une autre possibilité qu'il faudrait envisager pour le futur est le fait de pouvoir envoyer des informations issues du FRONTAL et à destination d'une autre machine connectée au SOLAR via ce dernier. Pour cette réalisation, il n'y a pas des difficultés majeures car l'adressage nous permettrait de désigner la destination. Une table supplémentaire (n° d'accès au service - localisation du service) nous permettrait d'envoyer les messages au SOLAR ou ailleurs.

Enfin, des calculs statistiques, par comptabilité de la perte de caractères de réception (ce qui entraine la perte de messages), pourraient nous amener à changer soit la distribution logique des traitements à faire (par introduction d'autre(s) processus) soit le matériel de transmission (pour augmenter la vitesse de la ligne).

On vient de voir quelques uns des futurs développements de notre réalisation. Cependant, on a surtout essayé de montrer, par cette étude, le besoin d'un effort de la part des concepteurs de système constructeur afin que leur futur système soit plus ouvert. Car la conception d'un système réparti est une idée séduisante et sa réalisation le serait d'autant plus si le logiciel de différents composants était conçu convenablement.

ANNEXES

Description du LCI

Le format défini est le suivant :

< origine >	< destination >	< ident utilisateur >	longueur	< CAT >	< FONCT >

{< param >} *	< information >

< origine > : numéro SU Origine (1 octet)

< destination > : numéro SU destination (1 octet)

< ident utilisateur > : identificateur de l'utilisateur pour le compte duquel s'effectue l'échange (2 octets)

Ces trois informations définissent l'adressage CCR.

Longueur (2 octets)

< CAT >

détermine le code opération de la commande

< FONCT >

Liste des commandes

1°) Contrôle de connexion

. CNEX : permet l'établissement d'une liaison entre un utilisateur et un service. Cette commande peut comporter des paramètres qui sont les appareils mis en jeu.

. DCNX : rompt une liaison utilisateur-service

. SUSP : suspend les échanges sur une liaison

. RACT : réactive une liaison

. ATTACH: permet d'attacher un appareil

. DTCH : détachement d'un appareil

2°) Contrôle de dialogue

. ATTN : attention

. PRTY : priority

3°) Commande d'échange

. EXCH : transmission d'information entre un utilisateur et un service ou entre deux services.

Syntaxe du LCE

a) Vocabulaire :

- < caractère> :: tout caractère ASCII. Le caractère "blanc" est considéré comme un séparateur.

Il existe des caractères spéciaux appelés caractères de contrôle.

Ce sont :

AC	Appel correspondant
E1	Evènement 1
E2	Evènement 2
FL	Fin de ligne
SC	Suppression de caractère
SL	Suppression de ligne

Ces caractères sont modifiables par les utilisateurs locaux et il en existe un jeu standard :

AC	ESCAPE
E1	CONTROLE A

E2	CONTROLE Y
FL	CARRIAGE RETURN
SC	BACK SPACE
SL	

- <lettre> ::= toute lettre de l'alphabet minuscule ou majuscule
- <chaîne> ::= <lettre>{<lettre>}*
- <chiffre>::= 0/1/.../9
- <nombre> ::= <chiffre> { <chiffre>}*
- <letchif>::= <lettre>/<chiffre>

b) Une session de travail aura la forme suivante :

<session> ::= <début session><corps session><fin session>

<début session> ::= (AC) L[OGIN] <identificateur> (FL) <mot de passe> FL

<identificateur> ::=<nombre>

<mot de passe> ::= <chaîne>

<fin session> ::= (AC) L[OGOUT][: (FL) <mot de passe>] (FL)

<corps session> ::= {<ligne>}*

<ligne> ::= (AC) <ligne AC> / <ligne E> / (E1) / (E2)

<ligne E> ::= <data> (FL) : il s'agit d'une requête de transmission de données à un service.

<data>::= {<caractère>}*

Ce sont les données qui sont transmises par le correspondant au service duquel l'utilisateur est connecté.

<ligne AC> ::= <commande> (AC) <ligne AC> / <commande> (FL)

<commande> ::= <fonction> / <appel service> / <appel procédure>

<fonction> ::= <nom fonction> [∪ { <PP> ∪ }* <PP>] [∪ { <PMC> ∪ }* <PMC>] [∪ { <PP> ∪ }* <PP>]

<nom fonction> ::= <lettre> <lettre> [{ <lettre> }*]

<appel service> ::= <nom service> [{ <PP> ∪ }* <PP>] [∪ { <PMC> ∪ }* <PMC>] [∪ { <PP> ∪ }* <PP>]

<nom service> ::= <lettre> { <letchif> }*

<appel procédure> ::= <nom procédure> [{ <PP> ∪ }* <PP>]

<nom procédure> ::= <lettre> { <letchif> }*

<PP> ::= <paramètre>

<PMC> ::= <mot clé> = <paramètre>

<mot clé> ::= <lettre> <letchif>

<paramètre> ::= <chaîne>

Fonctions du LCE

- Liste des mots clés :

AC	AC
E1	E1
E2	E2
SC	Suppression de caractère
SL	Suppression de ligne
LL	Longueur de ligne
LP	Imprimante
LC	Lecteur
FI	Fichier
CL	Classe

- Liste des fonctions :

CS(ERV) Appel d'un service
Paramètre : nom du service, LP, LC, FI

DS(ERV) Fin d'appel d'un service

DA(TE) Demande ou changement de la date
Paramètre : date

HE(LP) Provoque l'impression de la liste de toutes les fonctions et de tous les services accessibles.

ET(AT) Donne un certain nombre de renseignements contenus dans l'environnement utilisateur

MS(SAGE) Envoi d'un message à un autre utilisateur
Paramètre : identificateur de l'utilisateur destinataire, suite de caractères constituant le message.

MO(DIF) Modification des caractères de contrôle
Paramètre : AC, E1, E2, SC, SL, LL
N'est accessible que d'un terminal du Frontal

IU(TIL) Introduction d'un utilisateur
Paramètre : Identificateur de l'utilisateur, mot de passe, CL

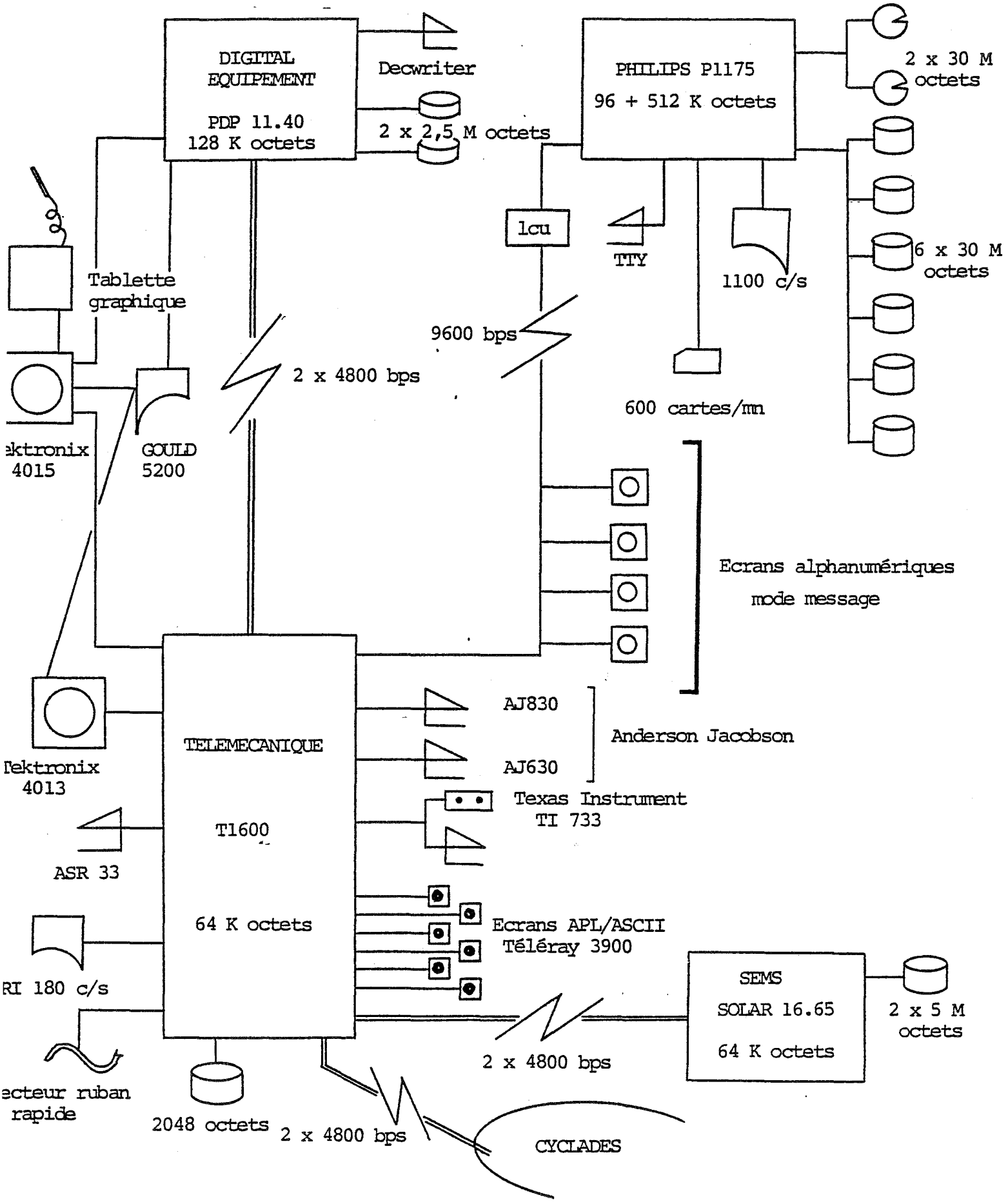
SU(TIL) Suppression d'un utilisateur
Paramètre : Identificateur de l'utilisateur à supprimer

MU(TIL) Modification des caractéristiques d'un utilisateur
Paramètre : Identificateur de l'utilisateur, CL

KU(TIL) Tuer un utilisateur (Logout forcé)
Paramètre : Identificateur de l'utilisateur

KA(NY) Déconnexion forcée de tous les utilisateurs ayant appelé un service donné
Paramètre : nom du service.

Moyens matériels du centre de calcul



BIBLIOGRAPHIE

SYSTEMES ET RESEAUX : GENERALITES

GEN-1 ANGELIDES J., CART M., CHAMBON J.F. etc...
Cours d'Informatique Industrielle
Ecole des Mines de Saint-Etienne (Décembre 78)

GEN-2 BOULLE C., THIERY L.
"Architecture des systèmes distribués"
Convention Inf. 1975, pp.82-84.

GEN-3 CHAMBON J.F., LE BIHAN B.
"Architecture d'un Frontal en environnement télé-informatique (application au réseau Cyclades)"
Thèse de Docteur Ingénieur, Ecole des Mines de St Etienne (Octobre 76)

GEN-4 CHAMBON J.F., LE BIHAN B.
"Architecture d'un calculateur Frontal en environnement télé-informatique"
Congrès AFCET, Gif-sur-Yvette (Novembre 76) pp. 881-890

GEN-5 CROCUS
"Systèmes d'exploitation des ordinateurs"
DUNOD (1975)

GEN-6 GUIBOUD-RIBAUD S.
"Mécanisme d'adressage et de protection dans les systèmes informatiques. Application au noyau GEMAU"
Thèse de Docteur d'Etat, Grenoble I (Juin 75)

GEN-7 MAC QUILLAN J.M.
"Strategies for implementation of Multi-Host computer network"
Computer Communication Review, vol. 6, n°4 (Octobre 76) pp. 19-24

GEN-8 MAEKAWA M.
"Queueign Models for Computer Systems Connected by a Communication Line"
J. Assoc. Comput. Mach.(U.S.A.), vol. 24, no.4, pp.566-582, Oct.77.

GEN-9 MAHL R.
"Téléinformatique et réseaux d'ordinateurs"
Ecole d'été AFCET, Tarbes (Juil. 74)

GEN-10 POUZIN L.
"Les réseaux informatiques"
SI N°01 B.P. 53 - 38041 GRENOBLE CEDEX. Mars 77

GEN-11 POUZIN L.
"Network Protocols"
Document Cyclades SCH 517 (Septembre 73)

LANGAGES DE COMMANDE

LAN-1 CHUPIN J.C.
"Command Languages and heterogeneous networks"
in Unger ed, IFIP Working conference on command languages, Suède (Juillet 74) pp. 311-318

LAN-2 DANG N.X.,SERGEANT G.
"Système et langage portable pour le traitement d'applications réparties sur une réseau hétérogène"
Séminaire IRIA : Langages et traducteurs (1977)

LAN-3 DU MASLE J.
"An evaluation of the LE/1 network command language designed for the SOC network
in Unger ed, IFIP Working conference on command languages, Suède (Juillet 74) pp. 319-326

LAN-4 SERGEANT G., FARZA M.
"Machine interprétative pour la mise en oeuvre d'un langage de commande sur le réseau Cyclades"
Thèse de Docteur de Spécialité, Toulouse Paul-Sabatier (Octobre 74)

RESEAUX

RES-1 BEAUFILS R., BACKMANN S., CABANEL J.P., LAGASSE J.P.
"ARAMIS - Conception d'un réseau de mini-ordinateurs pour l'accès à des services spécialisés"
Contrat SESORI n° 74-98, Lot n° 1

RES-2 BARBER D.L.A.
"The role and nature of a virtual terminal"
Computer Communication review, Vol. 7, No.3 (Juil.77)

RES-3 BOELL H.-P.
"Netzanschlüsse - Zugangsmöglichkeiten zu Netzwerken"
Nachrichtentech. Z. (NTZ) (Germany), vol.31, no.5, pp.333-338, Mai 1978.

RES-4 FRANCHI P.
"Distribution of functions and control in RPCNET"
in IEEE Computer Society ed, 3rd annual symposium on computer architecture (Janvier 76) pp. 130-135

RES-5 HEBENSTREIT J.
"Projet M2 - Conception et réalisation d'un système multi-mini-ordinateur"
Congrès AFCET, Gif-sur-Yvette (Novembre 76) pp. 725-731

RES-6 KOHN M.
"Raccordement d'un ordinateur XDS 94 de télésystèmes au réseau expérimental de commutation par paquets RCP".
Convention inf. 1975, pp 35,38.

RES-7 MANNING E., HOWARD R., O'DONNELL C., PAMMETT K., CHANG
"A UNIX-based local processor and network access machine".
Computer Networks, vol. 1, n° 2 (Sept.76) pp.139-142.

RES-8 PAPACRISTODOULOU Z.
"TELCOM. Un support de terminaux lourds sous un système à temps partagé".
Thèse de Docteur Ingénieur Grenoble (18.10.1974)

RES-9 QUINT V.
"Complément de spécifications pour l'utilisation du Protocol Appareil Virtuel"
Document Cyclades TER 540.1 (Mai 77)

RES-10 QUINT V.
"protection logicielle contre les erreurs dans un réseau d'ordinateurs hétérogène. Application au 360/67 du réseau Cyclades"
Thèse de Docteur Ingénieur Grenoble (20.12.1976)

RES-11 SCHREIBER F.A., MARTUCCI R.
"COGELA : COLLEGAMENTO SPERIMENTALE TRA DUE ELABORATORI".
Telecommunicazioni (Italie), no.57, pp.65-74, Dec.1975

RES-12 SOC Project - Report 1.
Project departement, IBM France (Juin 71).

RES-13 SUSHINE C.A., DALAL Y.K.
"Connection Management in Transport Protocols".
Comput. Networks, Vol. 2, no.6, pp.454-73, Dec. 78.

RES-14 WEBER S.
"Concentrateur Cyclades - Introduction aux concepts et à l'usage"
Support cours formation Cyclades (1977)

RES-15 ZIMMERMANN H.
"Proposal for a virtual terminal protocol (VTP)"
Document Cyclades TER 533.1 (Juillet 76)

RES-16 "Is there any reason why these two computers cannot be joined together ?"
Data System (GB), 23, Feb. 1979.

RES-17 "R T X 25 un outil de raccordement réseau à interface X 25"
Document CAP/SOGETI/SYSTEMES.

CENTRE DE CALCUL REPARTI

CCR-1 ANGELIDES J., CART M., CHAMBON J.F.
"Spécifications externes et internes de la machine gérant les demandes d'entrées-sorties M:DM"
Note interne, EMSE (Avril 77)

CCR-2 ANGELIDES J., CART M., CHAMBON J.F.
"Spécifications externes de maintenance de la machine M:DM"
Note interne, EMSE (Mai 77)

CCR-3 ANGELIDES J., CART M., CHAMBON J.F., TOSAN Y.
"Architecture globale du Frontal - Premières définitions"
Note interne, EMSE (Novembre 76)

CCR-4 CART M.
"Développement d'un logiciel de Frontal pour un centre de calcul réparti"
Rapport DEA, EMSE (Octobre 76)

CCR-5 CART M.
"Fonctions frontales dans un environnement réparti bâti sur un réseau local".
Thèse de Docteur Ingénieur, Ecole des Mines de Saint-Etienne (11.12.1979)

CCR-6 CHAMBON J.F., GUIBOUD-RIBAUD S., LE BIHAN B., TOSAN Y.
"Intérêt et faisabilité d'un centre de calcul fondé sur un réseau de mini-ordinateurs"
Note interne, EMSE (Octobre 76)

CCR-7 J.F. CHAMBON, J.J. GIRARDOT, S. GUIBOUD-RIBAUD
"Langage interactif et accès à des réseaux"
Saint-Pierre de Chartreuse (Oct. 75)

CCR-8 S. GUIBOUD-RIBAUD
"Système de réseaux de mini-ordinateurs spécialisés"
Note interne (Déc. 75)

CCR-9 JUREMA M., NIANG A.
"Système de Coopération de Systèmes de Gestion de Fichiers".
Note interne, EMSE (novembre 79).

CCR-10 TOSAN Y.
"Langages et protocoles dans un centre de calcul réparti"
Thèse de Docteur de 3ème cycle, Ecole des Mines de ST ETIENNE (15 Mars 78)

REFERENCES TECHNIQUES

TECH-1 Manuel de présentation du calculateur Solar16
SEMS

TECH-2 Manuel de présentation du calculateur T1600
TELEMECANIQUE

TECH-3 TSM16
Manuel de référence (Juillet 1977)
SEMS

TECH-4 BOS-D
Manuel de référence (Novembre 1977)
SEMS

TECH-5 IOCS
Manuel d'utilisation (Mai 1978)
SEMS

TECH-6 IOCS
Manuel de référence (Mars 1978)
SEMS

TECH-7 FMS
Manuel d'utilisation (Octobre 1978)
SEMS

TECH-8 FMS
Manuel de référence (Février 1978)
SEMS

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 avril 1974,

VU le rapport de présentation de M. J.F. CHAMBON,

Monsieur Jean ANGELIDES

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR de 3e CYCLE, spécialité Systèmes et réseaux informatiques

Fait à Saint-Etienne, le 6 décembre 1979.

Le Président de l'I.N.P.G.,

Ph. TRAYNARD.

Le Directeur de l'EMSE,

G. ARNOUIL.