# Mécanismes de base et réalisation de fonctions pour l'utilisation interactive d'un réseau d'ordinateurs

Amine Zhiri

N° d'ordre

# THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir le titre de

Docteur de 3ème cycle
«informatique»

par

Amine ZHIRI

## MECANISMES DE BASE ET REALISATION DE FONCTIONS POUR L'UTILISATION INTERACTIVE D'UN RESEAU D'ORDINATEURS

Soutenue le 8 décembre 1973, devant la Commission d'examen

| | | |
|---|---|---|
| M. | N. GASTINEL | Président |
| MM. | L. BOLLIET | Examinateurs |
| | S. KRAKOWIAK | |
| | J. DU MASLE | |
| | S. GIRARDI | Invité |

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

---

LISTE DES PROFESSEURS

---

Président : Monsieur Michel SOUTIF
Vice-Président : Monsieur Gabriel CAU

## PROFESSEURS TITULAIRES

| | | |
|---|---|---|
| MM. | ANGLES D'AURIAC Paul | Mécanique des fluides |
| | ARNAUD Georges | Clinique des maladies infectieuses |
| | ARNAUD Paul | Chimie |
| | AUBERT Guy | Physique |
| | AYANT Yves | Physique approfondie |
| Mme | BARBIER Marie-Jeanne | Electrochimie |
| MM. | BARBIER Jean-Claude | Physique expérimentale |
| | BARBIER Reynold | Géologie appliquée |
| | BARJON Robert | Physique nucléaire |
| | BARNOUD Fernand | Biosynthèse de la cellulose |
| | BARRA Jean-René | Statistiques |
| | BARRIE Joseph | Clinique chirurgicale |
| | BENOIT Jean | Radioélectricité |
| | BERNARD Alain | Mathématiques Pures |
| | BESSON Jean | Electrochimie |
| | BEZES Henri | Chirurgie générale |
| | BLAMBERT Maurice | Mathématiques Pures |
| | BOLLIET Louis | Informatique (IUT B) |
| | BONNET Georges | Electrotechnique |
| | BONNET Jean-Louis | Clinique ophtalmologique |
| | BONNET-EYMARD Joseph | Pathologie médicale |
| | BONNIER Etienne | Electrochimie Electrométallurgie |
| | BOUCHERLE André | Chimie et Toxicologie |
| | BOUCHEZ Robert | Physique nucléaire |
| | BOUSSARD Jean-Claude | Mathématiques Appliquées |
| | BRAVARD Yves | Géographie |
| | BRISSONNEAU Pierre | Physique du solide |
| | BUYLE-BODIN Maurice | Electronique |
| | CABANAC Jean | Pathologie chirurgicale |
| | CABANEL Jean | Clinique rhumatologique et hydrologie |
| | CALAS François | Anatomie |
| | CARRAZ Gilbert | Biologie animale et pharmacodynamie |
| | CAU Gabriel | Médecine légale et Toxicologie |
| | CAUQUIS Georges | Chimie organique |
| | CHABAUTY Claude | Mathématiques Pures |
| | CHARACHON Robert | Oto-Rhino-Laryngologie |
| | CHATEAU Robert | Thérapeutique |
| | CHENE Marcel | Chimie papetière |
| | COEUR André | Pharmacie chimique |
| | CONTAMIN Robert | Clinique gynécologique |
| | COUDERC Pierre | Anatomie Pathologique |
| | CRAYA Antoine | Mécanique |

| | | |
|---|---|---|
| Mme | DEBELMAS Anne-Marie | Matière médicale |
| MM. | DEBELMAS Jacques | Géologie générale |
| | DEGRANGE Charles | Zoologie |
| | DESRE Pierre | Métallurgie |
| | DESSAUX Georges | Physiologie animale |
| | DODU Jacques | Mécanique appliquée |
| | DOLIQUE Jean-Michel | Physique des plasmas |
| | DREYFUS Bernard | Thermodynamique |
| | DUCROS Pierre | Cristallographie |
| | DUGOIS Pierre | Clinique de Dermatologie et Syphiligraphie |
| | FAU René | Clinique neuro-psychiatrique |
| | FELICI Noël | Electrostatique |
| | GAGNAIRE Didier | Chimie physique |
| | GALLISSOT François | Mathématiques Pures |
| | GALVANI Octave | Mathématiques Pures |
| | GASTINEL Noël | Analyse numérique |
| | GEINDRE Michel | Electroradiologie |
| | GERBER Robert | Mathématiques Pures |
| | GIRAUD Pierre | Géologie |
| | KLEIN Joseph | Mathématiques Pures |
| Mme | KOFLER Lucie | Botanique et Physilogie végétale |
| MM. | KOSZUL Jean-Louis | Mathématiques Pures |
| | KRAVTCHENKO Julien | Mécanique |
| | KUNTZMANN Jean | Mathématiques appliquées |
| | LACAZE Albert | Thermodynamique |
| | LACHARME Jean | Biologie végétale |
| | LAJZEROWICZ Joseph | Physique |
| | LATREILLE René | Chirurgie générale |
| | LATURAZE Jean | Biochimie pharmaceutique |
| | LAURENT Pierre-Jean | Mathématiques appliquées |
| | LEDRU Jean | Clinique médicale B |
| | LLIBOUTRY Louis | Géophysique |
| | LOUP Jean | Géographie |
| Mle | LUTZ Elisabeth | Mathématiques Pures |
| MM. | MALGRANGE Bernard | Mathématiques Pures |
| | MALINAS Yves | Clinique obstétricale |
| | MARTIN-NOEL Pierre | Seméiologie médicale |
| | MASSEPORT Jean | Géographie |
| | MAZARE Yves | Clinique médicale A |
| | MICHEL Robert | Minéralogie et Pétrographie |
| | MOURIQUAND Claude | Histologie |
| | MOUSSA André | Chimie nucléaire |
| | NEEL Louis | Physique du solide |
| | OZENDA Paul | Botanique |
| | PAUTHENET René | Electrotechnique |
| | PAYAN Jean-Jacques | Mathématiques Pures |
| | PEBAY-PEYROULA Jean-Claude | Physique |
| | PERRET René | Servomécanismes |
| | PILLET Emile | Physique industrielle |
| | RASSAT André | Chimie systématique |
| | RENARD Michel | Thermodynamique |
| | REULOS René | Physique industrielle |
| | RINALDI Renaud | Physique |
| | ROGET Jean | Clinique de pédiatrie et de puériculture |
| | SANTON Lucien | Mécanique |
| | SEIGNEURIN Raymond | Microbiologie et Hygiène |
| | SENGEL Philippe | Zoologie |
| | SILBERT Robert | Mécanique des fluides |
| | SOUTIF Michel | Physique générale |

| | | |
|---|---|---|
| MM. | TANCHE Maurice | Physiologie |
| | TRAYNARD Philippe | Chimie générale |
| | VAILLAND François | Zoologie |
| | VALENTIN Jacques | Physique nucléaire |
| | VAUQUOIS Bernard | Calcul électronique |
| Mme | VERAIN Alice | Pharmacie galénique |
| M. | VERAIN André | Physique |
| Mme | VEYRET Germaine | Géographie |
| MM. | VEYRET Paul | Géographie |
| | VIGNAIS Pierre | Biochimie médicale |
| | YOCCOZ Jean | Physique nucléaire théorique |

## PROFESSEURS ASSOCIES

| | | |
|---|---|---|
| MM. | BULLEMER Bernhard | Physique |
| | HANO JUN-ICHI | Mathématiques Pures |
| | STEPHENS Michaël | Mathématiques appliquées |

## PROFESSEURS SANS CHAIRE

| | | |
|---|---|---|
| MM. | BEAUDOING André | Pédiatrie |
| Mme | BERTRANDIAS Françoise | Mathématiques Pures |
| MM. | BERTRANDIAS Jean-Paul | Mathématiques appliquées |
| | BIAREZ Jean-Pierre | Mécanique |
| | BONNETAIN Lucien | Chimie minérale |
| Mme | BONNIER Jane | Chimie générale |
| MM. | CARLIER Georges | Biologie végétale |
| | COHEN Joseph | Electrotechnique |
| | COUMES André | Radioélectricité |
| | DEPASSEL Roger | Mécanique des fluides |
| | DEPORTES Charles | Chimie minérale |
| | GAUTHIER Yves | Sciences biologiques |
| | GAVEND Michel | Pharmacologie |
| | GERMAIN Jean-Pierre | Mécanique |
| | GIDON Paul | Géologie et Minéralogie |
| | GLENAT René | Chimie organique |
| | HACQUES Gérard | Calcul numérique |
| | JANIN Bernard | Géographie |
| Mme | KAHANE Josette | Physique |
| MM. | MULLER Jean-Michel | Thérapeutique |
| | PERRIAUX Jean-Jacques | Géologie et Minéralogie |
| | POULOUJADOFF Michel | Electrotechnique |
| | REBECQ Jacques | Biologie (CUS) |
| | REVOL Michel | Urologie |
| | REYMOND Jean-Charles | Chirurgie générale |
| | ROBERT André | Chimie papetière |
| | DE ROUGEMONT Jacques | Neurochirurgie |
| | SARRAZIN Roger | Anatomie et chirurgie |
| | SARROT-REYNAULD Jean | Géologie |
| | SIBILLE Robert | Construction mécanique |
| | SIROT Louis | Chirurgie générale |
| Mme | SOUTIF Jeanne | Physique générale |

## MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

| | | |
|---|---|---|
| Mle | AGNIUS-DELORD Claudine | Physique pharmaceutique |
| | ALARY Josette | Chimie analytique |
| MM. | AMBLARD Pierre | Dermatologie |
| | AMBROISE-THOMAS Pierre | Parasitologie |
| | ARMAND Yves | Chimie |
| | BEGUIN Claude | Chimie organique |
| | BELORIZKY Elie | Physique |
| | BENZAKEN Claude | Mathématiques appliquées |
| | BILLET Jean | Géographie |
| | BLIMAN Samuel | Electronique (EIE) |
| | BLOCH Daniel | Electrotechnique |
| Mme | BOUCHE Liane | Mathématiques (CUS) |
| MM. | BOUCHET Yves | Anatomie |
| | BOUVARD Maurice | Mécanique des fluides |
| | BRODEAU François | Mathématiques (IUT B) |
| | BRUGEL Lucien | Energétique |
| | BUISSON Roger | Physique |
| | BUTEL Jean | Orthopédie |
| | CHAMBAZ Edmond | Biochimie médicale |
| | CHAMPETIER Jean | Anatomie et organogénèse |
| | CHIAVERINA Jean | Biologie appliquée (EFP) |
| | CHIBON Pierre | Biologie animale |
| | COHEN-ADDAD Jean-Pierre | Spectrométrie physique |
| | COLOMB Maurice | Biochimie médicale |
| | CONTE René | Physique |
| | COULOMB Max | Radiologie |
| | CROUZET Guy | Radiologie |
| | DURAND Francis | Métallurgie |
| | DUSSAUD René | Mathématiques (CUS) |
| Mme | ETERRADOSSI Jacqueline | Physiologie |
| MM. | FAURE Jacques | Médecine légale |
| | GENSAC Pierre | Botanique |
| | GIDON Maurice | Géologie |
| | GRIFFITHS Michaël | Mathématiques appliquées |
| | GROULADE Joseph | Biochimie médicale |
| | HOLLARD Daniel | Hématologie |
| | HUGONOT Robert | Hygiène et Médecine préventive |
| | IDELMAN Simon | Physiologie animale |
| | IVANES Marcel | Electricité |
| | JALBERT Pierre | Histologie |
| | JOLY Jean-René | Mathématiques Pures |
| | JOUBERT Jean-Claude | Physique du solide |
| | JULLIEN Pierre | Mathématiques Pures |
| | KAHANE André | Physique générale |
| | KUHN Gérard | Physique. |
| | LACOUME Jean-Louis | Physique |
| Mme | LAJZEROWICZ Jeannine | Physique |
| MM. | LANCIA Roland | Physique atomique |
| | LE JUNTER Noël | Electronique |
| | LEROY Philippe | Mathématiques |
| | LOISEAUX Jean-Marie | Physique nucléaire |
| | LONGEQUEUE Jean-Pierre | Physique nucléaire |
| | LUU DUC Cuong | Chimie organique |
| | MACHE Régis | Physiologie végétale |
| | MAGNIN Robert | Hygiène et Médecine préventive |
| | MARECHAL Jean | Mécanique |
| | MARTIN-BOUYER Michel | Chimie (CUS) |

| | | |
|---|---|---|
| MM. | MAYNARD Roger | Physique du solide |
| | MICHOULIER Jean | Physique (IUT A) |
| | MICOUD Max | Maladies infectieuses |
| | MOREAU René | Hydraulique (INP) |
| | NEGRE Robert | Mécanique |
| | PARAMELLE Bernard | Pneumologie |
| | PECCOUD François | Analyse (IUT B) |
| | PEFFEN René | Métallurgie |
| | PELMONT Jean | Physiologie animale |
| | PERRET Jean | Neurologie |
| | PERRIN Louis | Pathologie expérimentale |
| | PFISTER Jean-Claude | Physique du solide |
| | PHELIP Xavier | Rhumatologie |
| Mlle | RIERY Yvette | Biologie animale |
| MM. | RACHAIL Michel | Médecine interne |
| | RACINET Claude | Gynécologie et obstétrique |
| | RENAUD Maurice | Chimie |
| | RICHARD Lucien | Botanique |
| Mme | RINAUDO Marquerite | Chimie macromoléculaire |
| MM. | ROMIER Guy | Mathématiques (IUT B) |
| | SHOM Jean-Claude | Chimie générale |
| | STIEGLITZ Paul | Anesthésiologie |
| | STOEBNER Pierre | Anatomie pathologique |
| | VAN CUTSEM Bernard | Mathématiques appliquées |
| | VEILLON Gérard | Mathématiques appliquées (INP) |
| | VIALON Pierre | Géologie |
| | VOOG Robert | Médecine interne |
| | VROUSSOS Constantin | Radiologie |
| | ZADWORNY François | Electronique |

MAITRES DE CONFERENCES ASSOCIES

| | | |
|---|---|---|
| MM. | BOUDOURIS Georges | Radioélectricité |
| | CHEEKE John | Thermodynamique |
| | GOLDSCHMIDT Hubert | Mathématiques |
| | SIDNEY STUARD | Mathématiques Pures |
| | YACOUD Mahmoud | Médecine légale |

CHARGES DE FONCTIONS DE MAITRES DE CONFERENCES

| | | |
|---|---|---|
| Mme | BERIEL Hélène | Physilogie |
| Mme | RENAUDET Jacqueline | Microbiologie |

Fait le 30 mai 1972.

*Je tiens à exprimer mes remerciements à :*

*Monsieur le professeur Noël GASTINEL, Directeur du Centre Interuniversitaire de Calcul de Grenoble qui a bien voulu me faire l'honneur de présider le jury de cette thèse,*

*ainsi qu'aux autres membres du jury :*

*Monsieur Louis BOLLIET qui a accepté la direction de cette thèse et a montré sa bienveillance pour mon travail,*

*Messieurs Sacha KRAKOWIAK et Jean DUMASLE qui se sont intéressés à mon travail et ont accepté de le juger,*

*Monsieur Serge GIRARDI avec lequel j'ai travaillé profitant ainsi de son expérience et qui n'a ménagé ni son temps ni ses critiques.*

*J'exprime aussi ma reconnaissance à la compagnie IBM-FRANCE pour l'aide que j'ai pu trouver au sein du Centre Scientifique de Grenoble; ce qui m'a permis de mener à bien ce travail.*

*Je remercie également l'équipe système du Centre Interuniversitaire de Calcul de Grenoble pour son aide malgré les difficultés occasionnées par la modification de son système.*

*Je ne saurais oublier le service de Reproduction du laboratoire pour la réalisation matérielle de cet ouvrage.*

TABLE DES MATIERES

# I N T R O D U C T I O N.

La plupart des constructeurs fournissent avec leur matériel un éventail assez large de systèmes d'exploitation qui diffèrent essentiellement par les possibilités offertes à leurs utilisateurs. Celles-ci peuvent être si fondamentalement différentes qu'elles créent parfois des passions. La querelle qui opposa , et continue d'opposer , les partisans des systèmes d'exploitation par lots, appelés généralement systèmes "batch", et ceux des systèmes à partage de temps en est le meilleur exemple.

Etant donné que les travaux soumis dans un centre de calcul ne sont pas exclusivement d'un seul type, la solution adoptée dans quelques centres consiste à utiliser alternativement, selon un emploi du temps les divers systèmes dont ils ont besoin. Cela pose des problèmes de compatibilité entre l'emploi du temps de la machine et celui des utilisateurs surtout quand la présence de ces derniers est nécessaire comme c'est le cas des utilisateurs des systèmes conversationnels. Cette solution comporte plusieurs inconvénients : les changements se font à heures fixes, ce qui peut interrompre des

travaux en cours; le changement de système n'est pas instantané et entraine des opérations manuelles telles que démontage et montage de disques, ou implique l'exécution de programmes d'initialisation ( tambours, disques, imprimantes). Des tentatives d'uniformisation ont amené les concepteurs de systèmes à développer des outils permettant de réaliser les possibilités d'un système donné dans un système différent ; ceci au prix d'une dégradation des principales qualités caractérisant ces systèmes. Ainsi les utilisateurs de l'option interactive d'un système de "batch-processing" doivent se contenter d'un temps de réponse supérieur à celui d'un système à partage de temps pur, tandis que les utilisateurs d'un système interactif auquel ont été rajoutées des possibilités de traitement par lots ne disposent en fait que d'une partie des fonctions d'un vrai système "batch". Les principaux inconvénients d'une telle solution résultent du fait que des outils supplémentaires sont greffés sur des systèmes qui ne sont pas prévus pour cette utilisation et qui ne peuvent donc pas lui assurer de bonnes performances. Ainsi TSO [OS4] qui est une option de l'Operating System (O.S. 360) est contraint d'utiliser des méthodes de "swapping" qui dégradent le temps de réponse.

Plusieurs centres de calcul utilisent les services de deux ou plusieurs systèmes différents le plus souvent situés côte à côte. Très peu ont établi des liaisons entre les calculateurs qui restent spécialisés et communiquent par l'intermédiaire de paquets de cartes, bandes ou disque magnétiques qui doivent être manipulés par l'opérateur pour transmettre des informations d'un système à l'autre. En reliant plusieurs ordinateurs par des lignes de transmission nous obtenons un réseau d'ordinateurs. Les possibilités offertes par un réseau seront d'autant plus intéressantes qu'il

comprendra de systèmes différents.

Les réseaux d'ordinateurs semblent offrir des solutions acceptables à ces différents problèmes. La plupart des projets de réseaux sont encore à l'état d'études. Les plus avancés arrivent aux phases de tests, de mesures et d'évaluation de performance avant de passer à la phase d'exploitation. Nous avons un certain nombre d'exemples de ces évaluations. Certaines sont prometteuses [H1]. D'autres le sont moins [A8].

Le travail que nous présentons içi rentre dans le cadre du Système d'Ordinateurs Connectés (S.O.C), projet réalisé en commun par le Développement Scientifique de la compagnie IBM-FRANCE et certaines universités et organismes gouvernementaux. Le but de ce projet est de réaliser un réseau expérimental d'ordinateurs comprenant des systèmes de traitement par lots et des systèmes à partage de temps. Nous nous sommes plus spécialement intéressés à l'étude de l'emploi de systèmes à partage de temps dans un réseau de calculateurs.

L'utilisation de systèmes à partage de temps à partir de tous les centres du réseau est donc le but à atteindre. Parmi les fonctions de tels systèmes, celles qui consistent à gérer les communications entre les utilisateurs et le système prennent une dimension nouvelle dans un réseau d'ordinateurs. En effet les terminaux, organes d'entrées/sorties essentiels à l'intéraction entre l'utilisateur et le système ainsi que les périphériques tels que les imprimantes, les lecteurs et perforateurs de cartes sont autant de ressources que le système sait allouer aux divers demandeurs tant que ces unités appartiennent à son installation. En

étendant l'accès du système d'un centre à tous les utilisateurs du réseau, nous étendons ainsi les ressources gérées par ce système. La nature des liens entre ces nouvelles ressources et l'ordinateur rend inutilisables les procédures initialement prévues pour cette gestion; d'où la nécessité de développer des procédures nouvelles pour l'utilisation d'unités d'entrées/sorties distribuées à travers le réseau. Pour cela il faut résoudre au préalable les deux problèmes suivants:

- communication entre machines,
- et communication entre systèmes.

Si le premier point consiste à fournir des programmes de gestion de lignes de transmission, le second demande la création de tout un système de gestion de tâches correspondant à des fonctions précises du système. Ce dernier point n'est pas nouveau et constitue en fait une partie des fonctions des systèmes d'exploitation. Par contre la coopération entre des fonctions d'un système distribué à travers plusieurs machines ou entre les fonctions de systèmes implantés sur des machines différentes constitue un élément nouveau dont il faut étudier tous les aspects et les conséquences. Nous étudions dans ce qui suit aussi bien les problèmes de coopération entre les systèmes d'un réseau d'ordinateurs que l'utilisation d'unités éloignées en adoptant l'agencement suivant.

Le premier chapitre traite de généralités sur les réseaux d'ordinateurs et présente deux exemples de projets en cours de réalisation : ARPA et SOC. Nous terminons ce chapitre par la comparaison des méthodes de connexions utilisées dans ces deux réseaux.

Le second chapitre expose les mécanismes de base permettant à deux tâches situées dans des centres différents de

coopérer. Les problèmes de synchronisation que cela pose sont aussi abordés.

L'environnement d'un réseau d'ordinateurs nous amène à réviser certaines notions acquises par la pratique des systèmes actuels. C'est notamment le cas de l'utilisation des unités d'entrées/sorties. L'étude de la réalisation des opérations d'entrées/sorties sur des périphériques dispersés à travers un réseau d'ordinateurs fait l'objet du troisième chapitre.

Enfin dans le dernier chapitre nous décrivons les réalisations effectuées dans le cadre du réseau SOC et du système du Centre Interuniversitaire de Calcul de GRENOBLE pour vérifier les études et les hypothèses faites dans les chapitres précédents.

# C H A P I T R E  1

## GENERALITES ET EXEMPLES DE RESEAUX D'ORDINATEURS.

### I. CLASSIFICATION DES RESEAUX.

Le concept de réseau d'ordinateurs n'est pas nouveau. Il a été utilisé dans plusieurs domaines tels que la météorologie, la réservation aérienne etc.. Ce sont essentiellement des réseaux d'échanges d'informations qui se caractérisent par une spécialisation dans des services bien définis. Les centres de ces réseaux s'échangent soit des données brutes telles que des observations ou des mesures soit des informations plus élaborées constituées par des résultats de prévisions dans le cas d'un réseau météorologique. Les données brutes ne mettent en jeu que des matériels de saisie d'information qui sont des organes d'entrées/sorties. Quant aux secondes, elles nécessitent un traitement qui utilise en plus des machines d'une certaine puissance ce qui diffère des premiers réseaux que l'on appelle des réseaux de télétraitement. La

spécialisation de ces réseaux se manifeste par un traitement unique des informations parvenues dans le sens que ce sont toujours les mêmes programmes en exploitation sur des données constamment renouvelées.

Actuellement nous voyons se développer des réseaux de services qui diffèrent totalement des réseaux de transfert d'information. Si nous voulons établir une hiérarchie entre les différents types de réseaux d'ordinateurs existants ou en cours de réalisation, nous situerons au bas de l'échelle les réseaux de transfert d'information. A un niveau supérieur peuvent être considérés les réseaux effectuant des travaux à distance communément appelés systèmes de 'remote job entry' (RJE). Leur but est de soumettre des travaux à un système éloigné au moyen de terminaux lourds du type IBM 2780 ou 1130 ou de petits calculateurs tels que les IBM 360/20. Les résultats de ces traitements peuvent être récupérés sous forme de listes ou de paquets de cartes perforées. Le principe de ces systèmes consiste à effectuer les opérations de lecture d'impression ou de perforation par des machines satellites et à laisser la totalité de la phase d'exécution à une machine centrale. Nous pouvons citer le programme frontal HASP qui d'une part implémenté sur des terminaux lourds et d'autre part associé au système O.S. MVT permet de soumettre des travaux à ce dernier par l'intermédiaire des premières machines. Dans la même catégorie de réseaux nous trouvons CPREMOTE qui permet de soumettre des travaux à une machine virtuelle gérée par CP67 à partir de stations éloignées. La principale restriction de HASP est que toutes les opérations hormis celle d'exécution qui est en totalité traitée par la machine centrale, sont effectuées par le même calculateur. Ainsi les listes et les fichiers de cartes perforées sont toujours

dirigés vers le calculateur origine du travail. L'interaction dans le réseau se fait exclusivement entre deux centres dont l'un est toujours le centre principal. Une amélioration a été faite au moyen d'une liaison HASP à HASP qui relie plusieurs unités de traitement entre elles [R1]. Parmi les avantages d'une telle liaison nous trouvons un équilibrage des charges des centres de traitement et une extension des centres utilisés par un seul travail.

L'amélioration des réseaux du type RJE tente d'accéder à la catégorie immédiatement supérieure dans notre hiérarchie. C'est celle des réseaux de systèmes de traitement par lots ou systèmes de batch processing. La différence avec les réseaux du type précédent est la levée des restrictions quant aux relations entre les divers centres. Leur utilisation est différente dans la mesure ou les possibilités offertes sont plus vastes allant de l'envoi de travaux et de fichiers jusqu'à des exécutions parallèles de processus ou de tâches faisant partie du même travail. Des ressources telles que des banques de données ou des bibliothèques de programmes sont alors partageables et surtout accessibles à partir de tout centre. Les systèmes de cette catégorie de réseaux devraient être assez élaborés pour permettre un haut degré de parallélisme entre les exécutions des tâches, de synchronisation de ces tâches et surtout de décisions automatiquement prises par le système pour le choix des centres de traitement. Ce dernier point est connu sous le nom de partage de charge.

Au même niveau ou à un niveau supérieur peuvent être classés les réseaux interactifs qui en plus des possibilités des réseaux précédents doivent offrir aux utilisateurs une interaction avec leurs travaux et ce au niveau du réseau. A titre

d'illustration de cette classe de réseaux nous pouvons citer le prototype de réseau interactif CRIC [G1], le TERMINAL IMP [A1] du réseau ARPA et l'aspect conversationnel du projet SOC.

Cette tentative de classification des réseaux d'ordinateurs actuels ou en cours de réalisation se base essentiellement sur le mode d'utilisation et les possibilités de ces réseaux. D'autres classifications ont été faites d'après leurs caractéristiques topologiques ou d'après la nature des calculateurs. Une première classification distingue les réseaux centralisés, décentralisés et distribués. Si l'on prend comme critère les caractéristiques des machines les deux classes obtenues sont alors celles des réseaux homogènes et celle des réseaux hétérogènes. Les premiers regroupent des machines du même type, éventuellement différents modèles d'une même série, alors que les seconds utilisent des modèles différents.

## II. SYSTEMES ET RESEAUX D'ORDINATEURS.

Le rôle des systèmes d'exploitation est d'assurer la gestion des travaux. Ce rôle comprend la sélection des travaux à exécuter, leur ordre de passage et de priorité ainsi que la distribution des ressources de la machine à ces travaux. Dans le cas d'un réseau d'ordinateurs la notion de système a besoin d'être définie sinon au moins d'être précisée. En effet la situation est nouvelle du fait que l'on se trouve en face de plusieurs ordinateurs faisant partie d'un même ensemble. Le problème est de savoir ce qui dans cet ensemble peut représenter un système. Ce qui caractérise un réseau d'ordinateurs n'est pas le fait que l'on dise que telles machines feront désormais partie du réseau, en les reliant par des dispositifs de

communication, mais de définir les relations qui existent entre ces machines.

Les relations entre les ordinateurs d'un réseau comprennent les relations physiques et les relations logiques. Les premières décrivent la structure ou la configuration du réseau en précisant les couples de noeuds qui sont reliés directement par une ligne de transmission. Quant aux secondes, elles englobent l'interaction des systèmes de chacun des centres, les diverses facilités disponibles et leurs modes d'utilisation. C'est dans la notion de système que l'on détermine les relations logiques. Il y a deux manières de voir un système pour un réseau d'ordinateurs donc deux manières de préciser ces relations. La première consiste à considérer séparément chacun des noeuds du réseau et son système alors que la seconde est basée sur une vue plus globale qui conduit à l'idée d'un système pour tout le réseau. On peut dire que dans la première solution le réseau est considéré comme un réseau de systèmes et comme un réseau de machines dans la seconde. Cela conduit aux deux notions suivantes :

-Le réseau existe en soi, indépendamment des participants; cette situation est illustrée par les réseaux ARPA et CYCLADES,

-Le réseau est bâti à l'intérieur des systèmes comme cela est réalisé dans SOC et TSS.

## II.1. Système pour réseau de machines.

Lorsque le réseau est considéré comme un réseau de machines, il est nécessaire de privilégier un des centres en lui attribuant le système maître. Tous les autres centres reçoivent

un système esclave qui dépend du système central. De par sa position privilégiée il revient à ce dernier de prendre les décisions globales laissant ainsi les décisions locales qui relèvent des attributions des systèmes satellites. Il faut ainsi reécrire tout un système maître et des systèmes esclaves en incluant dans chacun d'eux des fonctions spécifiques au centre qui les supporte.

## II.2. Réseau de systèmes.

Les projets de réseaux d'ordinateurs qui se développent actuellement ne partent pas d'un ensemble de machines mais d'un ensemble de systèmes en exploitation. C'est notamment le cas des projets SOC [S1], ARPA [A2] et CYCLADES [C1]. Il est alors question d'adapter les systèmes à la nouvelle situation, ce qui peut représenter un moindre effort. Les problèmes qui se posent alors sont ceux des relations entre les différents systèmes. Ils comprennent les problèmes de communication, d'accès aux systèmes éloignés, de coopération de tâches situées dans des centres différents, de configuration de réseau etc....

## II.3. Configuration de réseaux.

L'un des premiers problèmes qui se posent dans une étude préliminaire pour réaliser un réseau d'ordinateurs est celui du choix de sa configuration. Etant donné un ensemble de centres, lesquels doivent être reliés directement par une ligne, suivant quels critères et quelles caractéristiques de lignes choisir. Les critères qui président au choix de la configuration comprennent les trois suivant: fiabilité, délai d'acheminement de

l'information et coût d'utilisation. Les deux derniers dépendent de la capacité des lignes utilisées.

La topologie du réseau influe sur la fiabilité de son système de communication. Celle-ci est fonction du nombre minimum de lignes en panne pour déconnecter deux centres quelconques. Sous cet angle un réseau centralisé offre la fiabilité minimum car il n' y a qu' un seul chemin d'un centre à un autre. Le choix d'une configuration décentralisée peut être dicté par l'importance que l'on accorde à certains centres et à certaines liaisons à l'intérieur du réseau. Ainsi une liaison entre deux centres sera d'autant plus fiable qu'il y aura de chemins disjoints entre eux.

Les choix deviennent plus difficiles à prendre quand le réseau comprend un grand nombre de centres. Il n'est plus question de déterminer la structure du réseau, la capacité des lignes à utiliser d'une manière empirique en envisageant tous les cas possibles. Une méthode pour étudier les configurations optimales de réseaux d'ordinateurs a été développée par HOWARD FRANK [A3 ; A4] . En se fixant pour but une combinaison acceptable entre le coût d'utilisation et le délai moyen d'acheminement, cette méthode détermine une configuration optimale en faisant varier les liaisons directes et les caractéristiques des lignes qui les équipent.

## II.4. Les problèmes de communication.

Un autre choix fondamental est nécessaire. Il concerne la méthode à utiliser pour effectuer le télétraitement. Deux techniques s'opposent: la technique dite du processeur frontal et la technique des fonctions de télétraitement intégrées au

système.

II.4.1. Technique du Processeur Frontal.

Ce procédé se caractérise par l'adjonction d'un petit calculateur à côté de chacun des ordinateurs principaux de chaque centre. Toutes les fonctions de télétraitement sont alors effectuées dans ce petit calculateur que l'on appelle processeur frontal (P-F). Cette technique est adoptée par les réseaux ARPA [A5] et CYCLADES [C2]. L'ensemble des P-Fs constitue un réseau homogène de calculateurs dont les fonctions consistent à mettre en forme les messages, à gérer les lignes de transmission et aiguiller les messages.

La mise en forme des messages suppose qu'il existe un format standard accepté par tous les P-F. En fait il est nécessaire d'avoir un protocole de dialogue entre ces calculateurs et à un autre niveau un protocole de conversation entre un ordinateur principal et un processeur frontal.

La gestion des lignes de transmission consiste à multiplexer les messages, lancer les opérations d'entrées/sorties, corriger les erreurs de transmission. Le multiplexage des messages est l'opération qui insère dans un même bloc plusieurs messages dont les destinations sont dans la même direction. Le système ajoute des informations afin que le receveur puisse reconnaitre les messages et effectuer l'opération inverse de démultiplexage. Les blocs ainsi préparés sont ensuite transmis. A ce niveau le système de télétraitement gère des queues constituées de blocs en attente d'émission et de blocs déjà émis. Un bloc émis n'est libéré que lorsqu'on est sûr qu'il est bien arrivé. Les blocs reçus avec erreur sont alors émis

plusieurs fois.

En fonction de la destination finale des routines d'aiguillage déterminent le chemin à emprunter. Cette procédure est utilisée exclusivement dans les réseaux décentralisés où l'on peut avoir à choisir un chemin. Plusieurs techniques sont utilisées. La plus simple consiste à choisir le chemin comprenant le nombre minimum d'étapes intermédiaires. Des techniques plus élaborées tiennent compte de la charge globale du réseau et des délais d'attente dans les centres intermédiaires. Un exemple particulièrement représentatif est celui du procédé d'aiguillage adaptatif étudié et simulé pour le projet ARPA [A6].

Il apparait que le système de communication effectué par des Processeurs Frontaux s'impose dans un réseau hétérogène. En effet les unités contrôlant les lignes ne sont pas compatibles avec tous les matériels. Il est alors avantageux de construire un sous-système de télécommunication indépendant des constructeurs participant au réseau. Le prix à payer consiste à construire une interface reliant un type de matériel et le P-F.

### II.4.2. Technique des fonctions intégrées.

Toutes les fonctions qui relèvent du rôle du P-F peuvent être effectuées par un module spécialisé dans le calculateur principal. Ces fonctions sont alors intégrées au système principal qui gère lui même ses lignes de transmission le reliant au réseau. Cette technique peut handicaper ce système car les diverses fonctions peuvent être coûteuses. Ainsi la mise en forme des messages demande beaucoup de mémoire de même que la gestion des queues en attente sur les lignes. La gestion des entrées/sorties des lignes de transmission quant à elles

demandent une grande rapidité dans le traitement des interruptions. L'aiguillage des messages se base sur une recherche dans des tables qui sont mises à jour régulièrement ce qui nécessite de la place pour les tables et du temps pour la mise à jour et le traitement. Evidemment pour un réseau constitué de machines et de systèmes analogues la technique des fonctions intégrées est particulièrement adaptée vu que les modules du système de télétraitement sont les mêmes pour tous les centres du réseau.

## III. EXEMPLES DE RESEAUX D'ORDINATEURS.

Dans ce paragraphe nous nous limiterons à l'étude de quelques aspects des réseaux d'ordinateurs ARPA et SOC en nous efforçant de souligner les caractéristiques de chacun ainsi que leurs différences.

### III.1. Le projet ARPA.

Le réseau d'ordinateurs développé par l'Advanced Research Project Agency (ARPA) est le projet le plus important qui ait été réalisé dans le domaine. Son importance tient aussi bien au nombre de ses participants qu'à ses ambitions. Plusieurs centres (une vingtaine) participent à sa réalisation. Le réseau est supposé founir les fonctions suivantes:

-Accès à des "HOSTS" éloignés à l'aide de terminaux isolés.

-Accès à des sous-systèmes d'un "HOST" éloigné par des utilisateurs d'un autre calculateur.

-Accès à des systèmes de fichiers éloignés.

-Transmission de fichiers de calculateur à calculateur.

-Partage de ressources.

Le matériel des centres de calcul du réseau est trés diversifié. Il comprend différentes marques de calculateurs sur lesquels sont implantés différents systèmes. De manière schématique le réseau ARPA peut être représenté en trois couches (FIG. I.1). Au niveau le plus bas se trouve le réseau des processeurs frontaux appelés "IMP". Le second niveau est constitué de calculateurs principaux ou "HOST". Un niveau intermédiaire est formé par les programmes de contrôle des fonctions du réseau ou N.C.P.

Les calculateurs frontaux sont tous de même nature spécialement conçus pour constituer un sous-réseau de communication. Tous reçoivent le même système qui a pour rôle de relier le ou les ordinateurs principaux (jusqu'à quatre) qui lui sont directement connectés au reste du réseau. Ce rôle comprend la distribution et l'aiguillage des messages, le multiplexage et le démultiplexage des données et les communications d'une part avec les ordinateurs principaux et d'autre part avec les "IMPs" adjacents.

Les "HOSTs" quant à eux se distinguent par leur variété et leur capacité de gérer plusieurs utilisateurs simultanément. La stucture de ce réseau a conduit à développer des conventions que doivent respecter les interlocuteurs dans leurs dialogues. Ces conventions sont appelées protocoles. Elles spécifient donc les méthodes d'échanges entre des "HOSTs" et des "IMPs" et des utilisateurs. Ces protocoles constituent l'un des éléments caractéristiques du réseau ARPA, aussi nous allons les présenter brièvement. Il existe trois niveaux de protocoles:

Niveau 1: protocole HOST-IMP et IMP-HOST.

Niveau 2: protocole HOST-HOST.

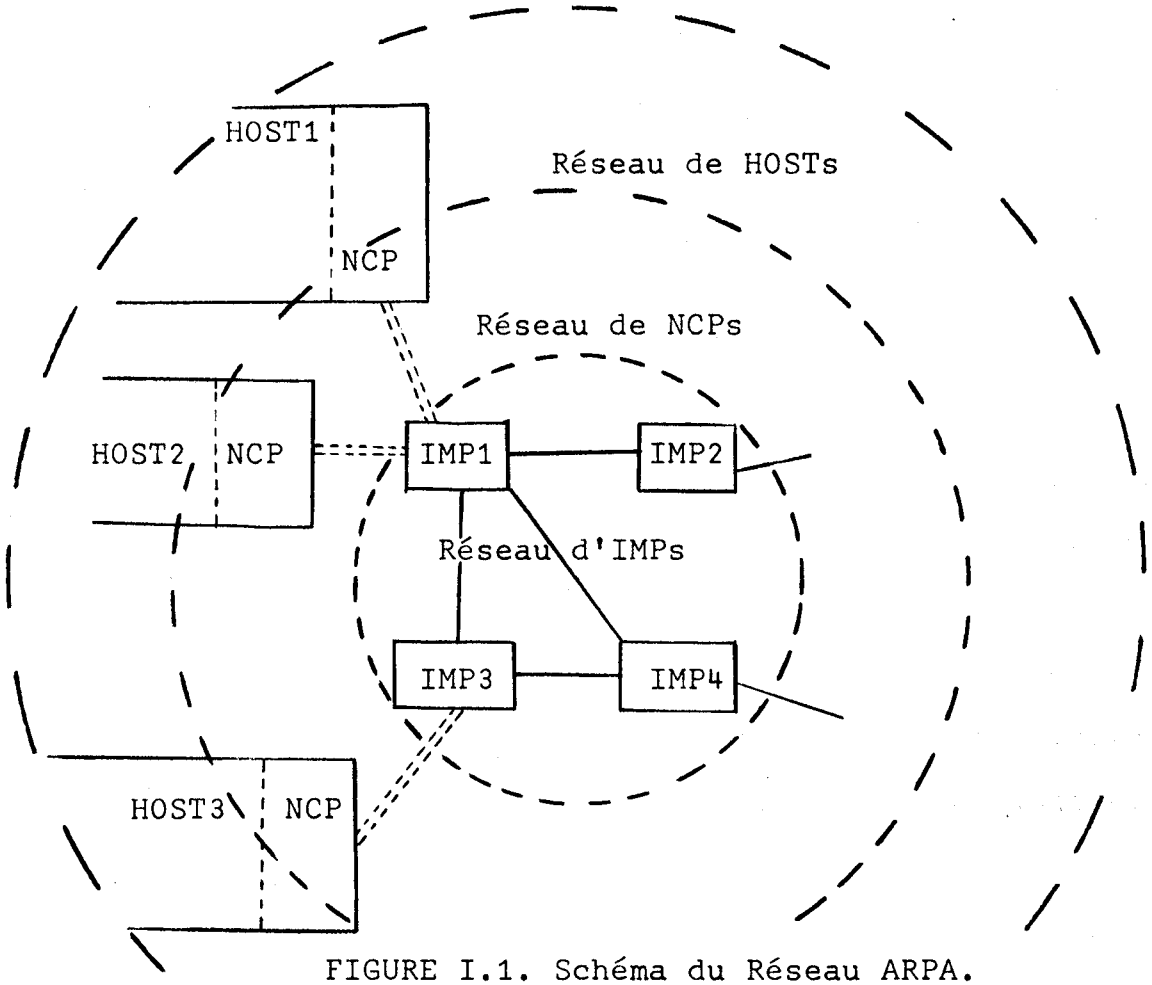



FIGURE I.1. Schéma du Réseau ARPA.

Niveau 3: protocole entre utilisateurs.

III.1.1. Le protocole du niveau 1.

Il sert à codifier les échanges entre un ordinateur principal et son processeur frontal. Les échanges doivent respecter les règles suivantes:

.un calculateur principal et son processeur frontal communiquent par des messages.

.un message est une suite de bits stucturée en deux champs: une en-tête et un corps dont les longueurs sont 32 bits pour l'en-tête et au maximum 8064 bits pour le corps.

.L'IMP gère les voies logiques unidirectionnelles qui

relient deux HOSTs.

.L'en-tête comprend les informations suivantes:

+HOST qui reçoit ou qui envoie suivant que le message est transmis par le calculateur principal au frontal ou inversement.

+Type du message.

+Numéro de la voie logique.

.Le type du message permet de différencier les messages réguliers qui ont pour destination finale un HOST des messages de contrôle utilisés entre le "HOST" et l' "IMP".

Le mécanisme des voies logiques permet d'éviter des engorgements dans le système de communication. Sur une voie logique le système d'acheminement n'accepte qu'un message à la fois. Cela signifie qu'à chaque fois l' "IMP" s'assure que le message qu'il a reçu du "HOST" est bien parvenu à sa destination avant d'envoyer à ce dernier un accusé de réception (RFNM message de contrôle) qui lui permet d'envoyer un autre message. La figure I.2 montre les étapes parcourues par le message du "HOST" A au "HOST" B à travers les "IMPs" IMP1,IMP2, et IMP3; le premier, IMP1, fractionne en deux (Mes1 et Mes2) le message et attend les deux accusés de réception (ack1 et ack2) avant de transmettre au "HOST" A l'accusé de réception global (RFNM).

III.1.2. Le protocole de second niveau.

Le protocole du deuxième niveau gère les échanges entre ordinateurs principaux. Il fournit un certain nombre de fonctions qui seront utilisées pour réaliser celles du protocole du niveau supérieur qui règle les dialogues entre processus. Ces derniers communiquent avec le reste du réseau par l'intermédiaire d'un

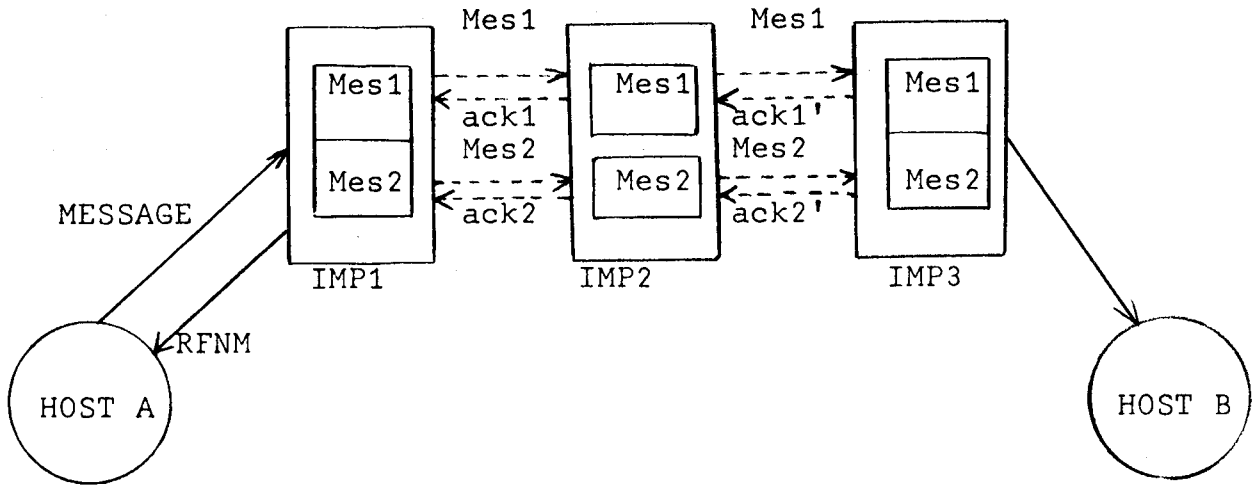



FIGURE I.2.

programme de contrôle (N.C.P.) qui regroupe toutes ces fonctions. Pour communiquer entre eux des processus - que ce soit des composants du système ou des programmes écrits par des utilisateurs- doivent établir des connexions qu'ils utilisent ensuite pour transmettre des données. Celles-ci ne sont soumises à aucune restriction quant à leur taille ou à leur contenu. A partir de là il revient aux NCPs d'effectuer les opérations suivantes:

-Etablir des connexions.

-Détruire des connexions.

-Contrôler le débit des informations à travers ces connexions.

L'établissement des connexions se fait au moyen de deux commandes que doivent émettre chacun des ordinateurs principaux concernés:

STR Prise émettrice Prise réceptrice Taille

RTS Prise réceptrice Prise émettrice Voie

La première commande STR émane du "HOST" émetteur vers le receveur et précise la prise réceptrice et la prise émettrice de la connexion ainsi que la taille de l'unité d'information des messages. La seconde commande RTS est envoyée par le "HOST" récepteur vers l'émetteur et indique en plus des prises la voie qui servira à acheminer les données. Celle-ci a été définie dans le protocole du niveau 1. Le choix de la voie logique est fait par le NCP, tandis que les prises sont données par les processus qui désirent établir la connexion. De cette partie du protocole il ressort qu'une prise est unidirectionnelle (on distingue NCP receveur et NCP émetteur), et qu'elle est établie par les NCPs concernés sur la demande de deux processus correspondants. L'ordre dans lequel ces commandes sont émises n'a aucune importance.

La clôture d'une connexion se fait à n'importe quel moment à l'aide de la commande:

CLS Ma Prise Votre Prise

Cette commande peut avoir deux significations. Elle sert soit à fermer une connexion déjà établie soit à refuser une connexion si elle répond immédiatement à une commande STR ou RTS. Le protocole stipule qu'un NCP qui reçoit une demande de clôture doit répondre par un commande CLS. Ceci permet de libérer les prises de la connexion pour en établir une autre.

La régulation du débit d'une connexion est aussi une

fonction du NCP. Elle permet de freiner les envois du "HOST" émetteur quand le processus destinataire n'est pas en mesure d'absorber assez vite les données qui arrivent.

Ce mécanisme de contrôle est basé sur une préallocation d'espace tampon par le calculateur récepteur et sur une connaissanece par le "HOST" émetteur de la place disponible pour l'émission. Celui-ci maintient à jour deux compteurs: un premier compteur contient le nombre de messages envoyés et le second contient à chaque instant la place disponible dans le "HOST" récepteur. A chaque envoi le premier compteur est incrémenté de 1 et le second diminué de la longueur du message. L'émission n'est autorisée que si le deuxième compteur est positif. En allouant de l'espace supplémentaire l'ordinateur récepteur favorise le débit alors qu'il le diminue en réduisant l'espace tampon. A cet effet les commandes suivantes ont été introduites:

ALL Voie N L

Cette commande est envoyée par l'ordinateur récepteur à l'émetteur pour lui permettre d'envoyer N messages ou L bits supplémentaires.

GVB Voie Fm Fb

En envoyant un ordre GVB le "HOST" récepteur demande à son interlocuteur de restituer de l'espace utilisé par la voie logique spécifiée en paramètre. Ce dernier répond en retournant la commande:

RET Voie N M

III.1.3. Les protocoles du troisième niveau.

Avec les protocoles de troisième niveau les problèmes d'utilisation du réseau par des programmes particuliers sont enfin abordés. La variété et la diversité de ces problèmes expliquent le nombre de ces protocoles. Nous allons citer ces familles de conventions sans entrer dans le détail.

III.1.3.1. Protocole de connexion initiale.

Disons tout simplement qu'il permet d'établir des liaisons entre deux processus dans deux centres différents. Une étude détaillée sera faite lors d'une comparaison entre les procédures de connexion d'ARPA et de SOC.

III.1.3.2. Protocole TELNET.

C'est un ensemble de conventions destinées à des utilisateurs ou à des programmes pour leur permettre de dialoguer avec des systèmes éloignés. Elles régissent le dialogue entre deux programmes correspondant en précisant l'ensemble des codes de caractères à utiliser, des fonctions disponibles ainsi que la manière dont est simulé un terminal hypothétique sur n'importe quel modèle de terminal. Ce protocole utilise le protocole de connexion initiale.

III.1.3.3. Le protocole de transfert de données.

Les conventions de transfert de données ont été introduites afin d'éviter la prolifération de programmes et de

protocoles d'échange de données. Elles servent de base à des protocoles ou à des applications telles que soumission de travaux à distance, transmission de fichiers, applications graphiques, etc..

III.1.3.4. Protocole de transfert de fichiers.

Les objectifs de ce protocole sont: développer le partage de fichiers (programmes ou données), éviter à l'utilisateur d'être confronté avec les problèmes résultant des différences entre les systèmes de fichiers que les centres du réseau utilisent. Des conventions standards sont ainsi arrêtées. Tout utilisateur ou programme qui désire accéder à des fichiers éloignés devra s'y conformer. Le protocole comprend des mécanismes de protection et d'accès sélectif aux données. L'hétérogénéité du réseau a conduit à fournir des méthodes pour définir la structure des données dans un fichier et la manière dont elles ont été rangées. Ceci est indispensable quand on veut retrouver des informations et les utiliser par des machines ou des systèmes dont les caractéristiques diffèrent de ceux qui ont servi à les produire et à les ranger.

III.2. Le projet SOC.

Le réseau du projet SOC se présente comme un réseau homogène d'ordinateurs supportant des systèmes différents. Cependant SOC est de conception hétérogène car la plupart de ses fonctions sont interprétées. Le système de communication fait partie des systèmes d'exploitation, ce qui constitue une différence avec la méthode utilisée par ARPA. La figure I.3.

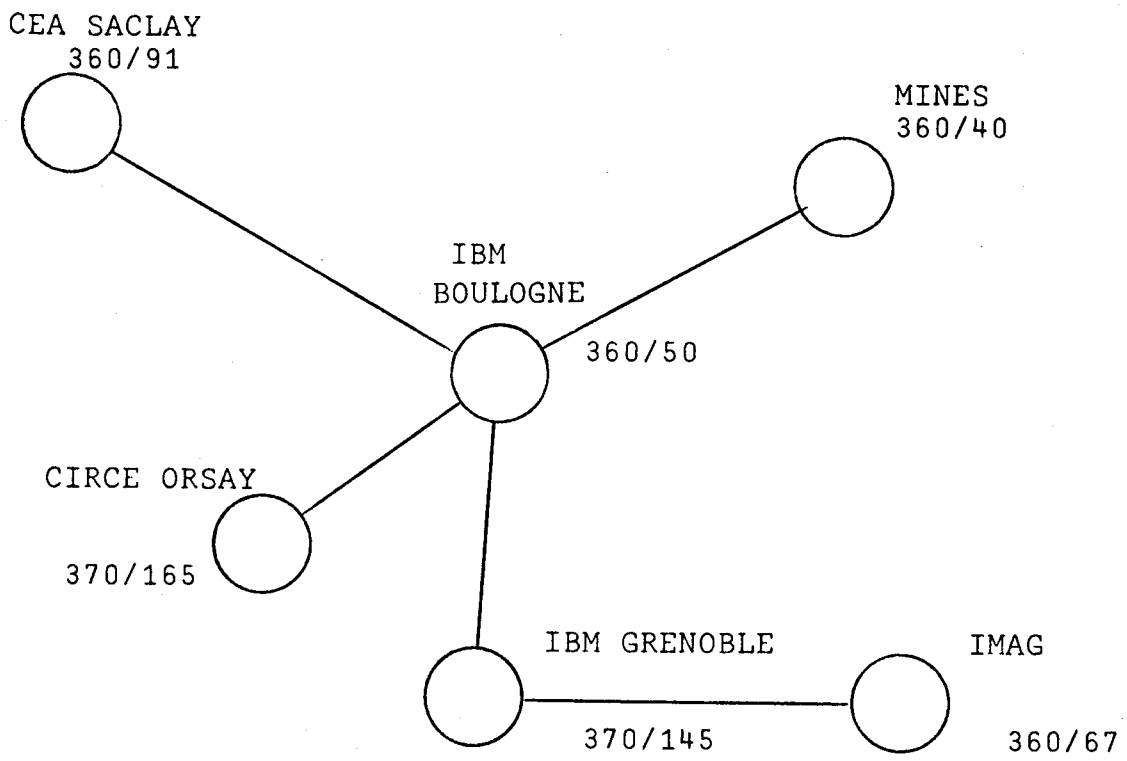



FIGURE I.3. Configuration du réseau SOC.

représente la configuration du réseau. Du point de vue logique le réseau est constitué par un ensemble de systèmes. Dans chaque centre un composant appelé Système-Réseau (S.R.) sert d'interface entre le système local et l'extérieur (FIGURE I.4.). Une solution de facilité a été adoptée en ce qui concerne l'implémentation du système-réseau. Ainsi dans un centre où l'OS/360 est en exploitation il est représenté par un travail (JOB). Dans l'état actuel du projet nous pouvons distinguer deux aspects: l'aspect lié à une utilisation par lots (BATCH) d'une part et l'aspect conversationnel d'autre part. Le premier aspect permet aux utilisateurs de soumettre leurs travaux au réseau par l'intermédiare de commandes. Celles-ci sont bien définies et constituent le langage de commande du réseau ou langage externe (LE). Une application écrite en langage externe et soumise au

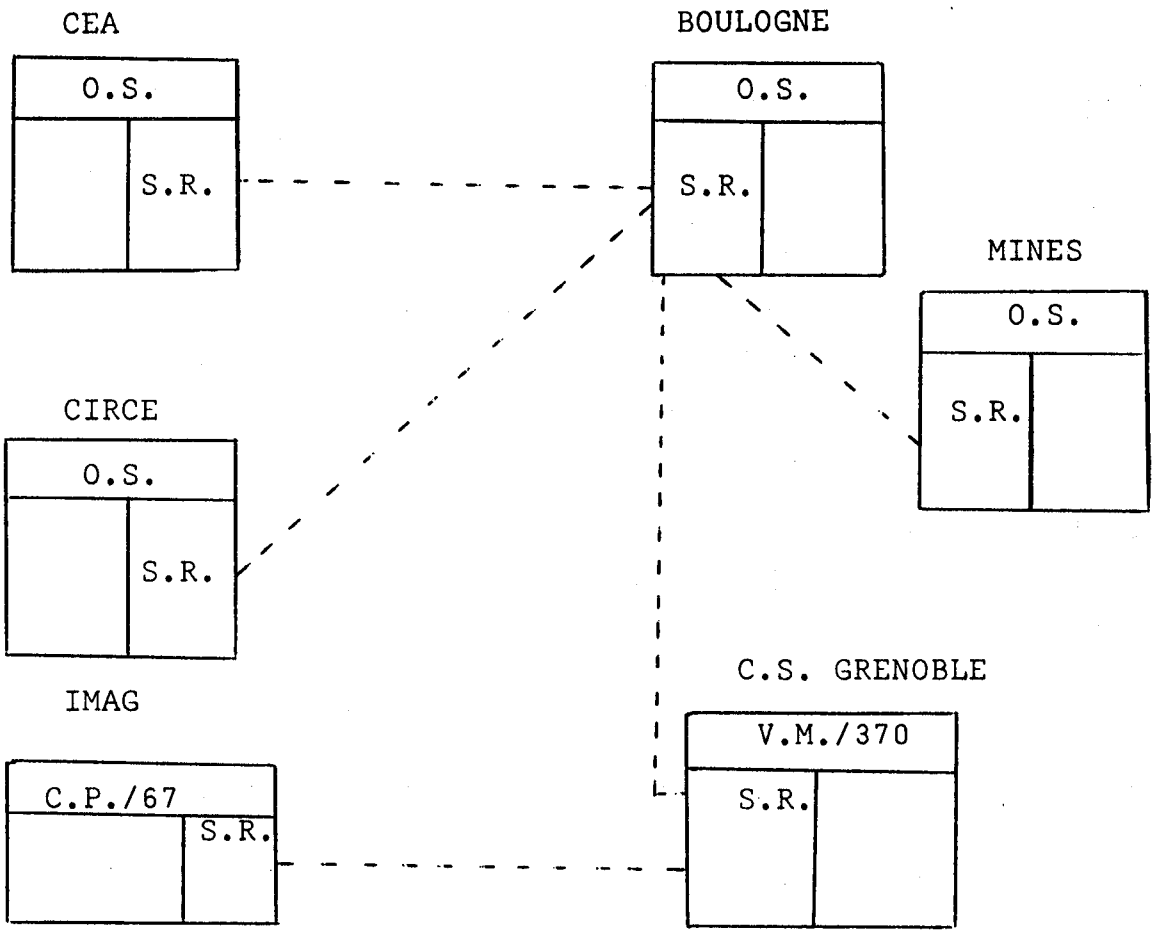



FIGURE I.4. Schéma logique du réseau SOC utilisé pour l'aspect conversationnel.

réseau est appelée "travail-réseau". Le système-réseau transforme le "travail-réseau" exprimé en langage externe en une série de travaux qui sont dirigés vers les centres d'exécution. Ces travaux sont exprimés en langage interne qui lui est directement interprétable localement par chacun des centres concernés par le "travail-réseau". Quant au second aspect, il permet l'interaction d'un utilisateur avec des systèmes éloignés à l'aide de terminaux. Cet aspect qui représente une grande partie de ce travail est présenté dans les chapitres 3 et 4.

III.2.1. Le langage externe. [S2]

C'est un langage de commande qui permet à l'utilisateur de spécifier ses besoins au système-réseau. Il offre la possibilité d'exprimer dans un langage général de haut niveau des opérations de type utilitaire telles que manipulation de fichiers (création, suppression, mouvement, recopie) et exécution de travaux. Ces derniers restent exprimés dans le langage de contrôle du système sur lequel ils doivent s'exécuter. Les déclarations dans un programme en langage externe associent des identificateurs à des entités telles que "data-set", ordinateurs, volumes, etc.. et permettent d'utiliser ces identificateurs dans les instructions du programme pour préciser au système l'utilisation que l'on veut en faire. L'influence du "JOB CONTROL LANGUAGE" de l'OS/360 qui est le système de la plupart des centres du réseau se ressent fortement sur la partie déclarative du langage qui comprend des déclarations de types: Data-set, space, membre de data-set partitionné, taille de bloc physique, d'organisation des données dans un fichier (PS: séquentiel, DA: accès direct, IS: séquentiel indexé, PO: partitionné).

Quant aux instructions exécutables du langage, directement inspirées de MTS, elles portent essentiellement sur des manipulations d'ensembles de données (EMPTY, DELETE, CREATE, COPY, ADD, CATALG, UNCATALG, LIST) et comprennent une commande d'exécution RUN et des opérations de calcul d'expressions arithmétiques. Pour donner une idée des possibilités du langage prenons des exemples de programmes en langage externe:

```
NETIN(6,6400,106),(AMINE.ZHIRI,PROG.RESEAU);
INP ALPHA:=
ZINZIN;
//JOB   JOB
   :
   :
// SYSIN DD *
   :
   :
/*
ZINZIN;
RUN ALPHA AT CIRCE.75 LISTON IMAG.67;
NETOUT;
```

Ce programme débute et finit par deux mots réservés (NETIN et NETOUT). Il déclare une variable ALPHA du type INPUTSTREAM qui est initialisée avec un programme en JCL OS/360 et commande son exécution au centre CIRCE avec des résultats produits au centre IMAG.

```
NETIN;
COMMENT 'LE FICHIER A EXISTE; ON LE DEFINIT';
DSN:=S5161.P0580.DECALUWE.IN/DC0010/2314/IMAG.67;
COMMENT 'LE FICHIER B EST A DEFINIR ET A CREER';
DSN B:=S9999.CALU.OUT/SAC005/2314/SAC91;
SPACE SPA:=§USED A;
CREATE B(SPA PS FB 800 80);
COPY A TO B;
NETOUT;
```

Ces deux exemples sont tirés de la thèse de R. DECALUWE [ S3 ].

Il est à noter que l'ambition du langage externe de SOC reste limitée vu l'absence d'instructions conditionnelles ou structurées, d'ordres d'exécutions parallèles et conditionnelles qui doivent faire leur apparition dans de tels langages afin de permettre des coopérations entre des travaux distribués dans le réseau.

### III.2.2. Architecture du Système-Réseau.

Le système-réseau est constitué d'un superviseur

(S.S.R.) et d'un ensemble de processus. Le superviseur est composé de modules qui ont pour rôle de gérer les processus. Cela comprend leur création leur activation et leur suppression et se fait au moyen des primitives suivantes:

a) Primitives de gestion.

CREER (TYP). C'est l'opération qui permet de créer un processus dont le type est donné en paramètre. Cela consiste à construire un PCB (bloc de contrôle du processus)
qui sera initialisé en fonction du type indiqué et d'inserer ce bloc dans la queue des processus activables.

MOURIR. Lorsqu'un processus a terminé ses fonctions il émet une macro instuction ( MOURIR). Le S.R. entreprend alors d'enlever le bloc qui représente la sous-tâche de la queue du système et de libérer la mémoire qu'il occupait.

b) Primitives de synchronisation.

WAIT(ECB,MASQUE). Par cette instruction le processus indique au système qu'il est en attente d'un événement. Le paramètre ECB sert à recevoir l'indication selon laquelle l'événement attendu s'est produit. Il contiendra alors une information que le paramètre MASQUE permet de reconnaitre.

POST(ECB,MASQUE). C'est l'opération complémentaire de la précédente. Elle est effectuée quand un événement se produit.

Ce sont là les opérations fondamentales pour la gestion des processus. Elles permettent de créer et de supprimer des

sous-tâches aussi bien que de les synchroniser sur des événements.

Les processus sont des sous-tâches de la tâche S.R. Chacune est un ensemble de programmes formant un tout. A chaque processus est attaché un PCB déstiné à décrire ce dernier pour le système. La notion de processus est à la base de tout travail soumis au réseau. A chaque activité correspond une sous-tâche. Le PCB contient un nombre d'informations qui comprennent :

-le type. Le type d'un processus dépend des fonctions qu'il effectue.

-La priorité. C'est un paramètre qui sert à la sélection du processus à activer.

-L'état. Un processus peut être dans l'un des états suivants:

+en attente d'un événement, donc non activable.

+actif, il a alors le contrôle de l'unité centrale.

+activable. Il est candidat au contrôle de l'unité centrale.

-son identificateur. Chaque processus peut être identifié grâce à un numéro à l'intérieur d'un système. Pour l'identification globale dans le réseau il suffit de préfixer l'identificateur local par le numéro du centre dans le réseau.

Afin de soumettre le "travail-réseau" au S.R. , d'effectuer la transformation du langage externe en langage interne et d'interpréter les programmes produits, le système-réseau dispose de processus spécialisés dans ces fonctions. Le Lecteur-Compilateur lit les commandes du "travail-réseau". Les programmes en langage interne qu'il produit sont traités par un Interpréteur qui lui même est créé par un

autre processus: l'Initiateur. La figure I.5. montre un exemple de "travail-réseau" soumis dans le centre A où il est lu par le Lecteur-Compilateur a1 qui produit trois programmes en langage interne: LI1, LI2 et LI3 qui sont rangés sur disque. L'Interpréteur a2 prend en charge LI1 demande la création des Interpréteurs b1 et c1 dans les centres B et C auxquels il envoie respectivement les programmes LI2 et LI3. Ces derniers sont interprétés localement par chacun des Interpréteurs.

Comme dans tout système le S.R. comporte un certain nombre de composants. Nous ne nous attarderons pas sur les composants classiques tels que le DISPATCHER ou la gestion de la mémoire. Le DISPATCHER peut adopter une politique d'activation des tâches se basant sur des allocations de tranches de temps ou de tranches de travail. La gestion des processus a été abordée à travers quelques unes de ses fonctions. Dans un environnement de réseau d'ordinateurs le plus interressant est d'étudier des composants nouveaux apportés au système par l'environnement même. Ces composants sont appelés processus-système.

III.2.2.1. Les processus-système.

Les processus système sont déstinés à servir toutes les activités du réseau. Ils servent le plus souvent d'interface entre les autres processus et les systèmes des autres centres. Ce sont des processus spécialisés dans l'utilisation de certaines ressources ou type d'activité aussi mettent-ils à la disposition des processus utilisateur des fonctions qui leur permettent d'accéder aux ressources sous le contrôle des processus système. Parmi ces derniers nous pouvons citer :

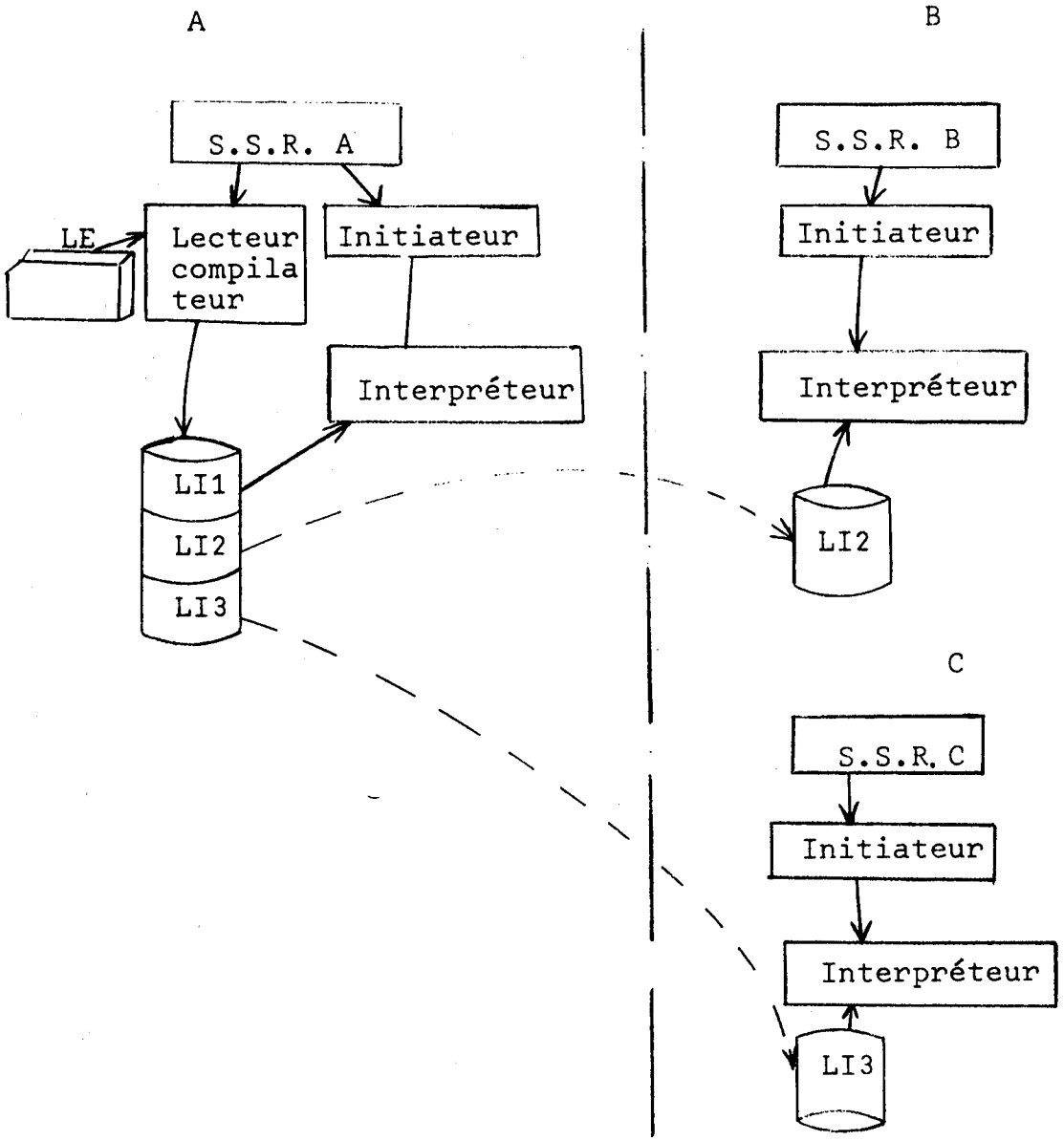



FIGURE I.5.

III.2.2.1. P0.

C'est le lien permanent entre les divers centres du réseau. Dans chaque centre il existe un processus P0 actif qui a des liaisons directes avec tous les autres P0 du réseau. Ces liaisons sont sous le contrôle de P0 et jouent un rôle fondamental car elles sont les seules existantes lors de

l'initialisation du système. Elles sont à la base de toutes celles qui seront établies au cours de l'activité du système. Ces liaisons sont constituées de la manière suivante:

Les centres sont numérotés globalement de 1 à N. Chaque P0 est affecté de N-1 liaisons qui sont: i--j pour j=1 à N et j≠i, i est le numéro du centre considéré. Les processus P0 utilisent les primitives de transmission de l'interface de communication au même titre que les autres processus. Toutefois ils sont dispensés des procédures d'initialisation des voies logiques. Les messages qu'ils s'échangent résultent en général de fonctions appelées par les autres processus afin d'établir des liaisons ou de créer des sous-tâches correspondantes. Les fonctions auxquelles font appel les processus et qui utilisent les liaisons permanentes entres les P0 sont RFP et RFL et seront détaillées plus loin.

III.2.2.1.2. L'interface de communication. [S4]

L'interface de communication est le processus système chargé de gérer les lignes de transmission. Sa fonction est d'organiser les opérations sur ces lignes et les données qui partent. Cette organisation consiste à faire des paquets, c'est à dire à multiplexer les messages avant de les envoyer et à les démultiplexer dès la réception. Les procédures de reconnaissance des messages, leur mise en ordre et la vérification des données reçues relèvent aussi du travail de l'interface de communication. Avant de présenter les fonctions de ce processus-système nous allons définir la notion de voie logique qui est à la base des communications entre processus.

C'est un numéro associé à un processus à sa demande quand il a besoin de transmettre ou de recevoir. Une voie

logique est bi-directionnelle, elle sert indifféremment pour envoyer ou recevoir, et ne contient en elle même aucune information sur la destination ou l'origine des données qu'elle va acheminer à partir ou vers le processus auquel elle appartient avant que celui-ci n'ait fait appel à certaines fonctions de l'interface de communication.

Les fonctions suivantes permettent à n'importe quel processus d'envoyer des données et de les recevoir en se servant des lignes :

a) LNKGET(LPNO).

Cette commande est exécutée par un processus pour obtenir un numéro de voie logique. Ce numéro est attaché au processus LPNO et lui permettra de transmettre et de recevoir sur la voie logique que le numéro reçu identifie.

b) LNKOPN(LNKNO,RSITE,RPNO,RLNKNO).

Par cette commande le processus qui l'émet demande à l'interface d'initialiser la voie logique désignée par LNKNO. Ceci est fait à partir des paramètres qui sont:

-LNKNO est le numéro de la voie logique à initialiser.

-RSITE est le centre éloigné avec lequel le processus émetteur se met en communication.

-RPNO est le numéro du processus éloigné correspondant.

-RLNKNO est le numéro de la voie logique que ce dernier a obtenue (par LNKGET) en vue de communiquer avec le processus local.

c) SEND (LNKNO,BUFAD,BUFLG).

C'est la commande permettant d'envoyer des données contenues dans une zone dont l'adresse est BUFAD et dont la longueur est BUFLG. L'émission se fait sur la voie logique dont le numéro est le premier paramètre de la commande: LNKNO.

d) RCV (LNKNO,BUFAD,BUFLG,WAITIME,WAITECF,MSK).

Lorsqu'un processus s'attend à recevoir des informations d'un autre processus avec lequel il est en communication, il émet un RCV avec les paramètres suivants:

-LNKNO est le numéro de voie logique de communication entre le processus local et son correspondant.
-BUFAD est l'adresse de la zone destinée à recevoir les données.
-BUFLG représente la longueur de la zone.
-WAITIME est le temps pendant lequel le processus peut attendre les données.
-WAITECF est un mot qui recevra l'indication de l'arrivée effective des informations attendues.
-MSK est le masque qui servira à poster la zone WAITECF.

e) LNKCLS (LNKNO).

Lorsqu'un processus estime ne plus avoir besoin d'une voie logique, il la ferme pour indiquer à l'interface de

communication qu'elle n'est plus opérationnelle.

III.2.2.1.3.Le TIMER.

La gestion du temps est nécessaire pour certains processus tels que l'interface de communication. Celui-ci a besoin d'être averti de la progression du temps à intervalles réguliers. Tous les processus qui à un moment de leur activité désirent être avertis de l'écoulement d'une tranche de temps à partir de cet instant font appel au processus TIMER.

III.2.3.Protocole d'établissement de connexion.

Le protocole d'établissement de connexion entre deux centres est basé sur deux fonctions: RFP et RFL. Celles-ci ont été développées pour implémenter les modules d'utilisation des terminaux et de connexion à des systèmes conversationnels. Elles feront l'objet d'une étude détaillée dans le chapitre suivant. La fonction RFP (Request For Process ou demande de processus correspondant ) sert à créer un processus correspondant et a pour paramètres: le processus local, le type du processus éloigné à créer ainsi que le centre où il doit être créé. Quant à la fonction RFL (Request For Link ou demande de liaison), elle sert à mettre en correspondance deux voies logiques, chacune appartenant à un processus afin d'autoriser la communication entre eux.

Le protocole d'établissement d'une connexion peut se schématiser de la façon suivante:

| Pi/A | Pj/B |
| --- | --- |
| Call RFP(A,Pi,B,TYPj) | |
| retour=Pj | |
| Call LNKGET | Call LNKGET |
| retour=LNKi | retour=LNKj |
| Call RFL(A,Pi,LNKi,B,Pj) | Call RFL(B,Pj,LNKj,A,Pi) |
| retour=0 | retour=0 |

## IV. COMPARAISON DES METHODES DE CONNEXION DE SOC ET ARPA.

Le protocole de troisième niveau du projet ARPA fournit un grand nombre de prises aux utilisateurs rattachés à un centre. De même l'interface de communication de SOC permet aux processus d'un centre donné d'utiliser ses services à un niveau logique en leur fournissant des voies logiques. Mais dans un système comme dans l'autre aucune hypothèse n'est faite sur l'utilisation des prises ou des voies logiques. Faire des restrictions sur ces moyens de communication c'est figer le système logique de communication et détruire son caractère dynamique.

Si l'utilisation des voies de communication est souple, elle ne permet pas à elle seule d'établir les connexions entre processus. Pour cela il est nécessaire de définir quelques conventions. Celles-ci constituent le protocole à respecter de part et d'autre pour établir une liaison entre deux centres. Dans le projet ARPA l'I.C.P (Initial Connection Protocol) est utilisé à cette fin, tandis que SOC fournit pour cela un ensemble de fonctions.

### IV.1. L'ICP d'ARPA.

Rappelons que le réseau ARPA fournit trois familles de protocoles. Chaque HOST communique avec le reste du réseau par l'intermédiaire d'un petit calculateur appelé IMP (Interface Message Processor) qui est chargé de la manipulation des messages, de leur fractionnement en messages de longueur acceptable par les programmes de transmission, de diriger un message vers le centre correspondant à la première étape du

meilleur chemin possible à un instant donné, etc... Cette hiérarchie entre les divers composants du système et leurs fonctions a conduit à définir des protocoles à plusieurs niveaux:

-Le protocole du premier niveau gère les transactions HOST-IMP et inversement.

-Les échanges HOST-HOST sont soumis aux règles du protocole de second niveau.

-Quant à l'I.C.P. qui nous interresse directement, c'est un des constituants des protocoles de troisième niveau.

L'I.C.P. se situe au niveau de l'utilisateur et sert à connecter deux processus de deux centres différents: un processus usager (P-U) et un processus serveur (P-S). Il y a d'ailleurs plusieurs protocoles de troisième niveau ayant en commun la séquence d'instructions suivante:

| Processus serveur | Processus usager |
|---|---|
| (1) LISTEN (L,32) | (1') INIT (U,L,32) |
| (2) SEND (L,S) | (2') RECEIVE (U,S) |
| (3) CLOSE (L) | (3') CLOSE (U) |
| (4) INIT (S,U+1,tu) | (4') INIT (U+1,S,tu) |
| (5) INIT (S+1,U,ts) | (5') INIT(U,S+1,ts) |

L'instruction (1) indique que le processus serveur attend qu'un processus usager initialise une connexion entre la prise L et une prise du centre usager, la longueur du byte étant de 32 bits. Ceci est fait par l'instruction (1') , la prise réceptrice U est alors couplée à la prise émettrice L. Le P-S

envoie au P-U un numéro de prise S par l'instruction (2), l'instruction (2') permet alors au P-U de recevoir ce numéro. Les prises L et U ne servant plus à rien, la liaison L-U est alors close par l'émission simultanée de (3) et (3'). Les deux couples d'instructions (4)-(4') et (5)-(5') établissent alors la connexion demandée dans les deux sens: S-U+1 et S+1-U. Les paramètres tu et ts indiquent pour leur part les tailles des bytes dont il faut tenir compte dans chaque sens.

Un protocole I.P.C. est alors défini par un triplet L,tu ts et par une suite d'instructions. Le triplet sert à caractériser une activité. Ainsi deux processus interactifs pour l'utilisation de terminaux devront se conformer au protocole défini par le triplet:

L=1 ts=tu=8.

Quant à la séquence d'instructions elle sert à transmettre à chaque processus la prise du correspondant afin d'émettre la commande INIT. Ainsi le processus usager a transmis le numéro U au processus serveur qui à son tour a envoyé le numéro S. Le protocole comporte aussi la convention suivante: en même temps que l'on envoie un numéro on transmet le numéro suivant pour établir la connexion dans l'autre sens. Ces numéros sont obtenus dynamiquement et doivent satisfaire une seule condition: un numéro correspondant à une prise de réception est pair alors que celui d'une prise d'émission est impair. Dans la séquence d'instructions donnée plus haut U et S sont pairs alors que L est impair.

## IV.2.Etablissement de connexions dans SOC.

Le système du projet SOC se caractérise par le fait que chaque activité est représentée par un processus. Une connexion entre deux centres se fait alors nécessairement entre deux processus en associant deux numéros de voie logique, chacun étant fourni par un processus. Une voie logique étant bi-directionnelle, les processus peuvent l'utiliser aussi bien en émission qu'en réception de telle sorte que si vi et vj sont les voies logiques d'une connexion entre les processus Pi et Pj , les données émises par Pi sur vi seront reçues sur vj par Pj et inversement toute information envoyée par Pj sur la voie logique vj sera reçue par Pi par l'intermédiaire de la voie vi. Les voies logiques sont obtenues du système au moyen d'une instruction LNKGET émise par un processus, ce qui peut être fait à tout instant. Une seule restriction est faite à l'heure actuelle: par processus le nombre de voies logiques est limité à 256.

Dans le système un processus est privilégié. Appelé P0, il sert d'intermédiaire entre les différents centres dans toute opération de connexion. A cette fin dès l'initialisation du système de chaque nõeud du réseau un processus P0 est créé. Si n est le nombre de centres actifs dans le réseau, pour chaque centre le système établit automatiquement n-1 liaisons avec les n-1 centres restants.

Une connexion entre deux centres se fait en deux temps:

La première étape consiste à créer un processus dans le centre avec lequel on désire communiquer. Ceci est fait à l'aide d'une commande: RFP. Les processus se différencient par leurs types et par le numéro que le système leur attribue lors de leur

création. Les paramètres de RFP sont: le numéro du centre local, le numéro du processus qui émet l'instruction, le centre éloigné ainsi que le type du processus demandé. La commande RFP est adressée à PO qui la transmet à son tour au PO du centre demandé. Le résultat de cette commande est soit un numéro, celui du processus créé à distance, soit un code négatif indiquant que la requête n'a pas été satisfaite.

La seconde étape consiste à connecter deux processus et s'exécute à l'initiative de ces derniers qui agissent simultanément:

-D'abord chaque processus obtient du système un numéro de voie logique en émettant la commande LNKGET,

-Puis chaque processus émet la commande RFL qui a pour but d'obtenir le numéro de voie logique de son correspondant afin de préciser à l'interface de communication les éléments de la liaison qui sont: les deux numéros de processus et les deux numéros de voie logique. Ceci est fait par l'instruction LNKOPN qui initialise ainsi la connexion.

Les commandes RFL et RFP sont indépendantes. Mais elles se suivent lors de la première connexion entre deux processus. Si l'activité de ces derniers requiert une autre connexion, c'est à dire l'association de deux voies logiques, la suite d'instructions à émettre est la même pour les deux processus et comprend: LNKGET, RFL.

## IV.3. Conclusion.

De la comparaison des deux méthodes utilisées par ARPA

et SOC, il apparait que pour assurer une certaine liberté dans l'établissement de connexions, il faut passer par des connexions préétablies d'un niveau supérieur. Celles-ci se traduisent par des prises caractérisant les protocoles de troisième niveau dans le système d'ARPA. Pour le système SOC les liens existant entres les différents PO des noeuds du réseau constituent des connexions permanentes couvrant tout le réseau, permettant ainsi des connexions acquises dynamiquement.

# C H A P I T R E  2

## COOPERATION ENTRE PROCESSUS ELOIGNES.

La plupart des études et travaux de recherche sur les systèmes d'exploitation ont conduit à parler de processus. Ces derniers sont définis par rapport à une implémentation. Toutefois on retrouve toujours des références aux relations entre les processus et la fonction du système qui les gère d'une part, et d'autre part entre les processus eux mêmes. Cela amène donc à étudier des problèmes de coopérations entre les processus. La littérature dans ce domaine est assez abondante et traite de problèmes variés: synchronisation, coopération, parallélismes, allocation de ressources, blocage, gestion de processus avec des solutions aussi diverses: sémaphores, événements, algorithmes d'allocation de ressources pour éviter les blocages etc.. [B1-01-D1-H2] . La nature du matériel influe parfois sur les définitions de certaines notions. Ainsi dans une machine mono-processeur la définition du parallélisme entre deux processus peut être exprimée en termes d'indépendance de

leurs algorithmes malgrè un recouvrement éventuel de leurs exécutions dans le temps [H3] . Les problèmes d'identification ou de connaissance mutuelle de processus sont résolus par des méthodes diverses: variables situées dans des segments partageables dans le cas de MULTICS [O1] , nom d'un processus qui est simplement l'adresse en mémoire de son descripteur dans GMS [B1] .

Quant aux problèmes relatifs aux communications entre processus leurs solutions comprennent:

-l'accès à des informations partageables,

-la fourniture de paramètres à un processus par un autre; ce qui suppose un protocole entre ces derniers,

-une méthode de "boîte aux lettres" partagée par deux processus, remplie par l'un, vidée par l'autre, nécessitant un mécanisme de réveil d'attente et de recherche dans la "boîte aux lettres".

Dans un environnement de réseau d'ordinateurs tous ces problèmes se posent; parfois sous un angle différent. Ainsi le parallélisme logique défini pour une machine mono-processeur se transforme en parallélisme réel quand les deux processus s'exécutent sur deux calculateurs distincts. La gestion des processus prend un autre aspect du fait que dans un réseau de systèmes chaque système est responsable de ses processus locaux. Il faut cependant pouvoir nommer et atteindre des processus dans un centre éloigné. Mais les problèmes les plus délicats sont certainement ceux de l'allocation de ressources, de communication et de synchronisation. Des études commencent à aborder ces problèmes dans ce nouveau contexte. Mais aussi bien les problèmes d'allocation de ressources [S5] , que les problèmes de communication [A7] se réfèrent en général à une implémentation donnée. Cependant une étude plus globale [M1] généralise la méthode de HABERMAN [H2] pour résoudre le problème de

blocage mutuel de processus utilisant des ressources réparties sur les ordinateurs d'un réseau.

Dans ce chapitre, nous nous attaquons spécialement aux problèmes de communication entre deux centres, problèmes nouveaux qui mènent à la communication entre deux processus et à leur synchronisation. Bien que nous présentions ici une implémentation tenant compte des spécifications du projet SOC, nous nous efforçons de relever au passage les éléments indépendants du système choisi, donc transportables dans d'autres, et de donner des exemples d'utilisation.

## I. PROCEDURE DE COMMUNICATION ENTRE DEUX CENTRES.

L'utilisation effective d'un réseau d'ordinateurs met en jeu, dans une application donnée, au moins deux centres. Un travail soumis au réseau peut donc faire référence à deux centres de calcul, soit de manière explicite quand l'utilisateur spécifie explicitement les centres qu'il désire utiliser, soit de manière implicite quand il laisse au système la responsabilité du choix des centres suivant des critères de ressources ou de charge.

Une fois les deux centres déterminés, l'exécution du travail nécessite des échanges entre ces derniers et plus précisément entre les programmes qui représentent l'application dans chacun des deux centres. Pour permettre ces échanges les systèmes fournissent des moyens de communication entre deux programmes ou processus situés dans deux centres différents.

I.1. Création de processus correspondant.

La possibilité de créer des processus à distance ou d'obtenir un correspondant dans un centre éloigné est un des mécanismes de base dans un réseau d'ordinateurs. En effet il n'est pas possible de concevoir une coopération entre deux centres que ce soit au niveau des systèmes ou au niveau des processus sans un tel mécanisme. Plusieurs types de correspondances peuvent être envisagées suivant les besoins. On peut avoir à faire appel à la gestion des processus pour en créer un. Le processus dont on a besoin peut être déjà créé auquel cas on désire uniquement son identificateur. Les deux méthodes d'allocation de processus sont réalisées dans le système SOC par la même fonction notée RFP. Selon la méthode choisie elle sera appelée avec différents paramètres. L'appel à cette fonction doit avoir la forme suivante:

CALL RFP (RSITE,RTYPE,TYPCALL,WAITIME).

RSITE est le numéro du centre correspondant.
RTYPE est le type du processus demandé.
TYPCALL est le type de l'appel. Il peut prendre trois valeurs :
CREATE (création) quand on veut créer le correspondant.
NOCREATE (sans création) dans le cas où l'on désire le numéro du correspondant s'il est déjà créé sinon rien.
et LISTEN (écoute) pour indiquer que le processus est déjà créé et qu'il attend des appels du type NOCREATE.
WAITIME est le temps maximum que désire attendre l'appelant.

Dans tous les cas d'appel le résultat que délivre la

fonction est soit le numéro du processus correspondant soit un code indiquant la raison pour laquelle le processus correspondant n'a pas été alloué.

Dans chaque centre sont dressées deux tables RFPLTAB et RFPCNTAB déstinées à contenir respectivement les demandes du type LISTEN et les demandes du type CREATE et NOCREATE. Chaque entrée à un instant donné correspond à des demandes non satisfaites. Pour satisfaire ces demandes l'algorithme suivant est exécuté :

I.1.1. Fonctionnement interne de RFP.

Si l'appel est du type LISTEN on va au PAS 1, sinon au PAS 2.

PAS 1 : On remplit une entrée de la table RFPLTAB avec les données suivantes:

RFPLPNO : identificateur du processus demandeur,

RFPLRSITE: numéro du centre demandé,

RFPLLTYP: type du processus appelant,

RFPLANS: champ réservé pour le numéro du processus faisant une demande du type NOCREATE qui satisfera la demande en cours.

RFPLECB: ECB sur lequel le processus sera en attente jusqu'à satisfaction de sa demande.

Le processus appelant est alors mis en attente jusqu'à ce qu'une requête correspondante parvienne au centre local. A ce moment l'algorithme se poursuit au PAS 3.

PAS 2 : Si aucune entrée n'est libre dans RFPCNTAB le processus appelant est mis en attente sur deux événements: le premier se produira quand le temps limite spécifié par le paramètre WAITIME expire. Quant au second il correspond à la libération d'une entrée de RFPCNTAB. Le premier événement arrivé réveillera le processus qui continue au PAS 3.

PAS 21: Si une entrée est libre elle reçoit alors les mêmes informations que celles du PAS 1. Un message est alors envoyé au P0 du centre identifié par le paramètre RSITE de l'appel avec les données suivantes:

-RQSTRFP code x'00' indiquant que c'est une demande,

-LPNO numéro du processus appelant,

-RTYP type du processus demandé,

-LSITE numéro du centre demandeur,

CALLTYP type de l'appel: CREATE ou NOCREATE.

Après cela le processus est mis en attente, comme au PAS 2 sur le temps limite qu'il a précisé ou ce qu'il en reste (voir PAS 3) et sur la réponse à sa demande quand le centre RSITE l'enverra. L'algorithme se poursuit alors au PAS 3.

PAS 3 : Le processus appelant était en attente. Trois cas peuvent se présenter: Si c'est le temps limite qui a expiré l'entrée correspondante est libérée ce qui a pour effet de débloquer tous les processus en attente d'une entrée au PAS 2. Ensuite un code indiquant que la fonction RFP n'a pas pu être terminée faute de temps est transmis

au processus appelant. L'algorithme est terminé. Si le processus est réveillé par la libération d'une entrée l'algorithme reprend au PAS 21. Le troisième événement qui peut se produire au PAS 3 est la réponse à la requête envoyée au PAS 21. Dans ce cas on transmet le résultat de la réponse au processus appelant (soit le numéro du correspondant soit zéro si la demande a été refusée par le centre éloigné). L'entrée de la table RFPCNTAB est libérée et le contrôle rendu à l'appelant.

La partie de l'algorithme qui va être décrite dans la suite concerne les opérations entreprises dans le centre correspondant lorsqu'un message envoyé au PAS 21 parvient à sa destination.

Si le champ CALLTYP du message contient NOCREATE le PAS 1' est alors exécuté sinon c'est le PAS 2'.

PAS 1' : La table RFPLTAB est explorée en vue de trouver une entrée dont les éléments correspondent à ceux du message arrivé. Si une telle entrée n'est pas trouvée cela veut dire qu'aucun processus local n'a émis un appel du type LISTEN. Un message analogue est retourné au centre origine précisant que la demande n'est pas satisfaite:

-ANSRFP code x'01'

-LPNO numéro du processus demandeur,

-RTYP type du processus demandé,

-RSITE numéro du centre demandé,

-RPNO zéro indiquant: demande non satisfaite.

Lorsque l'entrée est trouvée le même message que celui qui

vient d'être décrit est envoyé avec RPNO contenant la valeur du champ RFPLPNO de l'entrée traitée. Le champ RFPLANS de celle-ci est rempli avec le contenu du LPNO du message parvenu. Ceci permet donc aux deux processus, celui qui a émis RFP du type LISTEN et celui qui a émis RFP du type NOCREATE d'échanger leurs identificateurs. Le premier reprendra l'algorithme au PAS 3.

PAS 2' : Le message parvenu correspond à un appel du type CREATE. Le module de gestion des processus est alors sollicité pour la création d'un processus dont le type est la valeur du champ RTYPE du message arrivé. Le nouveau processus reçoit le numéro de celui qui a demandé sa création. De même par l'intermédiaire d'un message du même type que celui du PAS 1' on fait parvenir au processus demandeur soit le numéro du nouveau processus soit zéro si le module de gestion ne l'a pas créé (type invalide ou nombre maximum de processus atteint .. etc ...)

Parmi les éléments utilisés dans cet algorithme, l'identification des processus de manière globale dans le réseau ainsi que les divers types d'appel peuvent être retenus comme essentiels dans toute méthode de création de processus à distance. Une fonction du système tenant le rôle de P0 s'avère aussi nécessaire comme le prouvent les solutions adoptées par ARPA et CYCLADES qui utilisent respectivement une méthode de "logger" et d'"abonné".

## I.2. Liaisons entre processus.

Parallèlement aux liaisons physiques existant

entre deux centres on définit des liaisons logiques. Les liaisons physiques sont matérialisées par les lignes de transmissions tandis que les liaisons logiques, elles, sont représentées par des numéros.

Par définition une liaison entre deux processus de deux centres différents est un couple de voies logiques. La liaison est réalisée en associant les deux numéros représentant les voies, chacun de ces numéros appartient à un processus. Ceci est effectué par la fonction LNKOPN.

Une liaison entre deux processus est un moyen d'effectuer un partage de ressource. En effet les lignes de transmission servent à acheminer les informations d'un noeud à un autre du réseau. Plusieurs travaux peuvent avoir à se servir de ces lignes en même temps. Les travaux s'exécutant dans un centre sont considérés par le système comme des processus. En attribuant des voies logiques à ces processus et en définissant des liaisons logiques entre eux on permet ainsi à plusieurs programmes de se servir d'une ligne qu'ils ont en partage. Un exemple de partage peut être illustré par la figure II.1.

Le réseau d'ordinateurs qui y est représenté est constitué des centres A,B,C,D,E,F. Les processus actifs sont a1,a2,b1,c1,d1,e1,f1,f2. On réunit en couples les processus en communication:

a1-e1 (1)

a2-d1 (2)

c2-f1 (3)

b1-f2 (4)

Les liaisons physiques sont :

A--B B--C B--D D--E et D--F

la liaison B--D est partagée entre les couples: (1),(2),(3) et

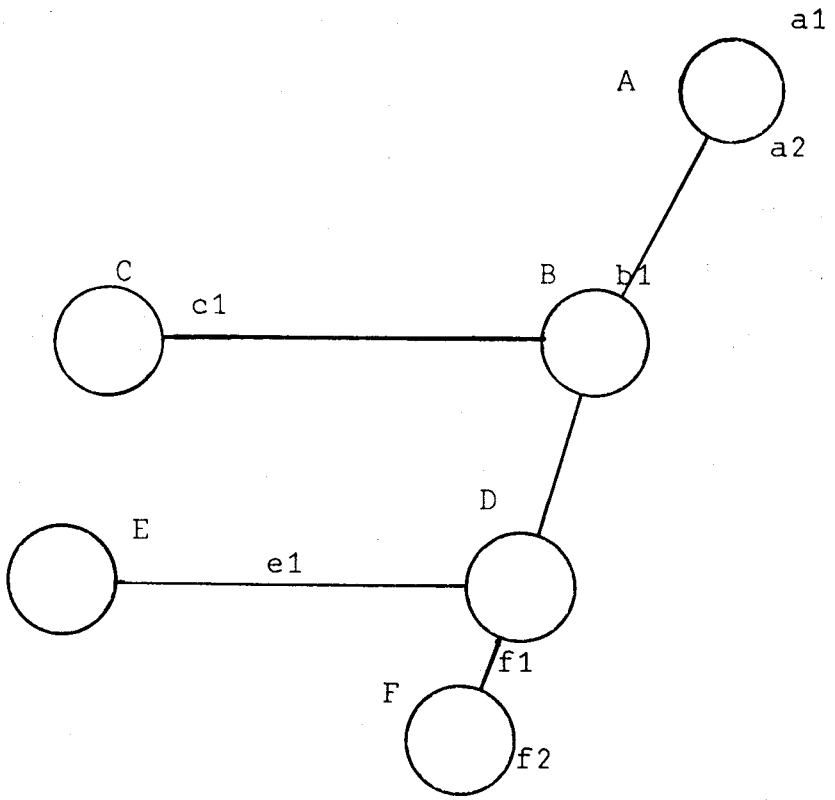



FIGURE II.1.

(4).

La liaison A--B est partagée entre les couples (1) et (2) et la liaison D--F entre les couples (3) et (4).

Comme dans tout partage de ressource on dispose de moyens de contrôle et d'attribution afin d'éviter des situations critiques. Dans ce cas le but est de prévenir une baisse d'efficacité dûe à une charge trop grande et d'éliminer tout traitement abusif d'informations de contrôle dû à l'utilisation d'un nombre trop grand de liaisons.

I.2.1. Liaisons permanentes.

Le processus P0 a été introduit plus haut. Sa principale fonction est de fournir un lien permanent entre le

centre local et le monde extérieur, c'est à dire l'ensemble des centres éloignés. Ainsi le processus P0 d'un centre connait l'existence de tous les P0 du réseau et de ce fait l'ensemble des processus P0 constitue des liens permanents entre les divers centres du réseau. Ceci par opposition à des liens qui peuvent être créés dynamiquement.

I.2.2. Liaisons temporaires.

Les liens temporaires sont créés au fur et à mesure des besoins. Leur création dépend du déroulement de certains travaux en cours. Ils demeurent jusqu'à ce que l'activité qui a commandé leur établissement demande leur suppression. Du fait de leur caractère dynamique ces liens sont soumis à un contrôle qui consiste à limiter leur nombre et à faire un certain nombre de vérifications suivant la charge du réseau et suivant les liaisons physiques disponibles.

Pour l'établissement d'une liaison temporaire les programmes de gestion et de contrôle utilisent les liaisons permanentes. Un processus Pi demande l'établissement d'une liaison avec un processus Pj éloigné par l'intermédiaire du P0 local qui transmet la demande au P0 du centre correspondant. Nous nous appercevons donc à ce point de la dépendance entre les liaisons temporaires et les liaisons permanentes. Ceci soulève immédiatement le problème posé par la destruction d'une liaison permanente par la déconnexion d'un centre ou la destruction d'un processus P0. La première hypothèse affecte de la même manière les liaisons permanentes avec le centre déconnecté et les liaisons temporaires qui le concernent. Les unes comme les autres sont inutilisables et les processus en sont informés par

l'interface de communication à la moindre utilisation. Dans le cas où le processus P0 est détruit cela n'affecte en rien les liaisons temporaires déjà établies mais interdit la création de nouvelles liaisons. Il reste donc la question des liaisons en cours de création; elle sera abordée dans la description du fontionnement interne de la procédure d'établissement d'une liaison temporaire.

I.2.2.1. Etablissement d'une liaison temporaire.

Une liaison est symétrique et de ce fait nécessite une action de la part des deux processus concernés. Ceux-ci doivent faire simultanément une demande de liaison qui se traduit par un appel à la routine RFL.

Les paramètres de l'appel sont:

a) Le numéro du centre éloigné RSITE.

b) Le numéro du processus éloigné RPNO.

c) Le numéro de voie logique locale de la liaison.

Par exemple les processus Pi et Pj devant communiquer doivent effectuer les appels suivants:

Pour Pi dans A: call RFL(B,Pj,vlik,t)

Pour Pj dans B: call RFL(A,Pi,vljl,t)

La routine RFL répond au processus appelant en lui indiquant si la liaison est opérationnelle ou pas.

L'établissement d'une liaison temporaire pose certains problèmes :

- Cet établissement doit être nécessairement symétrique. Aucun des processus qui demandent la liaison ne doit être privilégié par rapport à l'autre. Lors de l'exécution des deux processus il est impossible de prévoir lequel des deux va émettre sa demande en premier. Ceci en application de l'idée que dans un vrai réseau il ne peut y avoir de relation d'ordre entre les partenaires par opposition avec les réseaux de 'remote job' basés sur des relations de maitres à esclaves. C'est pour cela qu'il ne suffit pas d'enregistrer la première demande qui arrive et d'attendre la seconde pour passer au premier demandeur les informations envoyées par le second et transmettre au second celles du premier.

- Le problème le plus délicat est celui de la synchronisation des deux processus. Il englobe celui de la détection des pannes de transmission pendant l'établissement d'une liaison qui peuvent induire en erreur l'un des processus. La synchronisation des processus doit assurer chacun des processus de ne recevoir que les messages qui lui sont destinés.

Le premier point est résolu par une méthode dite de table de rendez-vous symetrique. Pour établir une liaison il faut disposer des deux demandes au même instant et ceci dans les deux centres concernés. La première condition ne peut être réalisée que par l'enregistrement de la première demande arrivée en attendant la seconde. Ceci justifie le nom de table de rendez-vous. La symetrie est dûe au fait que la demande est enregistrée dans les deux centres à la fois. Cette méthode

utilise dans chaque centre deux tables dont les entrées correspondent aux demandes RFL non satisfaites, un des processus n'ayant pas encore émis sa demande.

Les deux tables sont appelées RFLRQTAB et RFLANTAB et représentent respectivement la table des requêtes et la table des réponses.

Une entrée de RFLRQTAB est composée des champs suivants :

RQECF : sert à mettre le processus en attente.

RQSITE : contient le numéro du centre demandé.

RQLPNO : contient le numéro du processus demandeur.

RQRPNO : contient le numéro du processus demandé.

RQREPLY : destiné à recevoir soit le numéro de voie logique du processus correspondant soit zéro au cas ou il aurait été impossible de l'obtenir.

Une entrée de la table RFLANTAB contient les éléments suivants :

ANLPNO : contient le numéro du processus demandeur.

ANLSITE : contient le numéro du centre demandeur.

ANLLNKNO : représente le numéro de voie logique éloignée.

ANRPNO : identifie le processus demandé.

La méthode consiste à enregistrer les demandes RFL dans les deux centres simultanément. L'algorithme d'établissement d'une liaison est alors le suivant:

PAS 1.

Dès qu'un processus émet une demande RFL on explore la table RFLANTAB pour déterminer si le processus correspondant n'a pas été plus rapide. Dans ce cas cette table doit contenir une entrée dont les éléments correspondent aux paramètres d'appel de RFL. Si oui le message suivant (MES1) est envoyé au P0 du centre correspondant:

RFLANTORQ code indiquant à la routine de traitement du message que c'est une réponse à une demande déjà émise.
RFLLSITE identificateur du centre local.
RFLLPNO numéro du processus local.
RFLRPNO numéro du processus éloigné.
RFLLNKNO numéro de voie logique locale.

Les éléments 2,3 et 4 du message correspondent dans le même ordre à ceux de l'entrée dans la table RFLRQTAB du centre demandé qui attend la réponse. De cette manière la recherche se fait plus rapidement.

Une fois le message envoyé, l'entrée de RFLANTAB sélectée précédemment est libérée avant de passer au PAS 2.

Si aucune entrée de RFLANTAB n'intéresse la demande RFL en cours la recherche d'une entrée libre dans RFLRQTAB est lancée. Si aucune entrée n'est libre le processus est mis en attente jusqu'à ce qu'une entrée dans cette table soit libérée par une demande satifaite.

Quand une entrée est trouvée, ses éléments sont alors remplis avec les paramètres d'appel de RFL. Un message analogue à MES1 est envoyé au centre correspondant. Il en

diffère par le code en tête précisant qu'il s'agit d'une requête.

Le processus appelant RFL est alors mis en attente jusqu'à ce que la réponse arrive ou jusqu'à ce que le temps maximum que peut attendre le processus expire. Chacun de ces événements poste différemment RQECF et entraine la réactivation du processus au PAS 2 ou au PAS 22.

Du côté du centre correspondant :

Le message arrive à P0 qui le transmet à RFL. Si c'est une réponse le pas 1' est alors exécuté, sinon c'est le pas 2'.

PAS 1'.

On recherche l'entrée de RFLRQTAB correspondant à la réponse. Si on ne la trouve pas on ne traite pas le message car c'est le cas où les deux messages se sont croisés. Quand l'entrée est trouvée, son élément RQREPLY est rempli avec l'élément RFLLNKNO du message et RQECF est posté.

PAS 2'.

Le message arrivé est une requête. On cherche une entrée libre dans RFLANTAB que l'on remplit avec les éléments du message. Si la table est pleine on retourne immédiatement une voie logique nulle. Le processus ayant fait la demande insistera s'il le désire.

Revenons au cas où un processus en attente est activé.

PAS 22.

Le temps maximum que désire attendre le processus

est écoulé. On lui rend un code indiquant que la liaison qu'il voulait établir avec sa voie logique d'une part et celle fournie par son correspondant d'autre part ne s'est pas bien effectuée par manque de temps. L'exécution de RFL se termine.

PAS 2.

Le numéro de voie logique du correspondant nous est parvenu. La liaison peut alors être effectivement établie. Une des fonctions de l'interface de communication est alors appelée ( LNKOPN) pour indiquer à ce dernier les voies logiques de la liaison, les numéros des processus en communication ainsi que l'identification des deux centres. La procédure de synchronisation commence. Pour cela le message "**OK**" est envoyé par chacun des processus. Il est suivi d'un RCV avec attente limite. Si le temps expire la séquence SEND(**OK**), RCV est recommencée et ceci au maximum cinq fois. La boucle est interrompue quand le RCV en cours est satisfait avant l'expiration du temps et que le message reçu est "**OK**". Le pas suivant PAS 3 est alors exécuté. Si la boucle est exécutée cinq fois sans résultat on considère que la liaison ne peut être établie et le processus appelant en est averti.

PAS 3.

Cette séquence vise à éliminer tous les messages de contrôle "**OK**" envoyés par le processus correspondant. Elle consiste à envoyer un message "*OKOK*" et à exécuter un RCV avec attente. Si le temps expire le pas 22 est exécuté. La réception du message "*OKOK*" arrête l'exécution de RFL

et le processus appelant est averti que la liaison est opérationnelle. Si le message reçu est "**OK**" on relance le RCV avec attente.

L'algorithme qui vient d'être décrit a pour but d'assurer la liaison entre deux processus sur leur demande. La liaison aurait pu être considérée comme établie au PAS 2 après l'appel à la fonction LNKOPN ou à la fin du PAS 2 dès la réception du message "**OK**".

IL faut préciser que l'interface de communication rejette tout message arrivant pour une voie logique non ouverte, c'est à dire pour laquelle le processus auquel elle appartient n'a pas fait appel à la fonction LNKOPN. Aucune procédure ne peut assurer que les deux processus fassent LNKOPN en même temps ou au moins que de part et d'autre le LNKOPN soit exécuté avant que l'un des processus n'émette son premier SEND. C'est pour cela que le PAS 2 comporte après l'ouverture de la voie logique une séquence de SEND et RCV portant sur des messages de contrôle. Cette procédure garantit que les premières données envoyées par un processus à son correspondant seront effectivement reçues et seuls des messages de contrôle peuvent être éventuellement perdus. Les pertes n'ont aucune conséquence car il suffit qu'un seul message "**OK**" parvienne pour passer au PAS 3. L'itération de la phase suivant l'appel à LNKOPN ainsi que le délai associé au "RCV" ont pour but de favoriser ce passage. Le délai est fonction de la longueur du chemin séparant les deux centres, du temps moyen de transmission entre deux centres et du délai moyen de traitement du message dans les centres intermédiaires.

Si Tm est le temps moyen de transmission d'un bloc contenant le message d'un centre à un autre et tm est le temps moyen de traitement d'un message pour le diriger vers le centre voisin sur le chemin de sa destination, le temps minimum que doit attendre un RCV est :

$$T = n*( Tm + tm ) + tm$$

n est le nombre de lignes directes séparant les deux centres. Tm est de l'ordre d'une seconde tandis que tm, de l'ordre de la milliseconde, est négligeable.

Quand le PAS 3 est atteint il est certain que les voies logiques sont ouvertes toutes les deux. Tous les messages de contrôle "**OK**" doivent être éliminés. C'est pour cela que le message du PAS 3 est différent de celui du PAS 2. Quand le message "*OKOK*" est parvenu, on est certain que les voies logiques ont été ouvertes des deux côtés et qu'il n'y a plus de message de contrôle.

Nous remarquerons que les deux fonctions de création de processus à distance et d'établissement de liaisons sont bien distinctes. Nous ne trouvons pas ce découpage dans les deux projets ARPA et CYCLADES. La distinction entre les deux fonctions permet d'appeler la seconde à n'importe quel moment de l'exécution d'un processus.

## II. PROBLEMES DE SYNCHRONISATION.

Les problèmes de synchronisation sont multiples et sont

en général introduits par le paramètre temps des fonctions qui servent à créer des correspondants ou à établir des liaisons.

## II.1. Problèmes de synchronisation dans la fonction RFP.

Prenons un exemple pour présenter la difficulté:

Soit dans le centre A le processus a1 . Celui-ci demande la création d'un processus dans le centre B et peut attendre un temps t. La demande arrive en B, le processus b1 est alors créé et son identificateur envoyé au centre A. Pendant ce temps, supérieur à t, le processus a1 est signifié que le temps a expiré. La réponse envoyée par le centre B est ignorée et le processus b1 reste isolé. Pour éviter ce cas il faut que la création d'un processus par un autre soit immédiatement suivie par une tentative d'établissement de liaison. Le processus isolé peut s'apercevoir que la liaison avec le processus qui l'a créé ne peut s'effectuer. Il peut alors se détruire en exécutant la primitive MOURIR.

| A | B |
|---|---|
| a1: call RFP(B,TYPE,CREATE,t)<br>retour=0 | b1 est créé.<br>call LNKGET<br>retour=vb1 |
| | call RFL(a1,A,vb1)<br>retour≠0 |
| | call MOURIR. |

## II.2. Problèmes de synchronisation dans la fonction RFL.

Dans la procédure d'échange des voies logiques qui servent à définir une liaison entre deux processus la difficulté

vient du choix de l'instant auquel les processus doivent lancer leur commande RFL. Plusieurs cas peuvent alors se présenter:

II.2.1. Liaison initiale.

C'est la liaison établie immédiatement après la demande de création d'un processus correspondant. Cela se traduit par un appel à la fonction RFL suivant la commande RFP du processus demandeur. Le processus demandé exécute quant à lui une commande RFL aussitôt qu'il a le contrôle. Cela constitue en fait un protocole de liaison initiale.

II.2.2. Seconde liaison.

Un processus donné peut obtenir plus d'une voie logique donc il peut établir plus d'une liaison avec le même processus ou avec plusieurs processus. L'établissement d'une seconde liaison entre deux processus peut intervenir à n'importe quel moment de l'exécution des deux processus. Le principe consiste à utiliser la première liaison pour établir la seconde. Montrons sur un exemple la procédure à suivre:

Le processus a1 du centre A et b1 du centre B ont déjà une liaison établie. Sur l'initiative d'un des processus (a1), une seconde liaison est établie. Pour cela a1 envoie à b1 sur la voie logique va1 un message. Ensuite a1 obtient une seconde voie logique va2 par LNKGET et exécute la commande RFL. Quand b1 reçoit le message de a1, il exécute à son tour LNKGET et RFL pour établir la liaison.

Les deux types de liaisons précédentes s'appliquent à

deux processus qui ont chacun les renseignements nécessaires sur l'autre. La seule inconnue est l'instant précis de l'établissement d'une liaison qui peut dépendre de plusieurs facteurs déterminés uniquement au moment de l'exécution. En réitérant cette procédure nous constatons que le nombre de liaisons auxquelles peut participer un processus n'est limité que par le nombre de voies logiques que le système lui affecte. Cependant malgré cette multiplicité de liaisons entre deux processus, il faut envisager une autre procédure si l'on veut permettre des protocoles de plus haut niveau et des coopérations entre plus de deux processus à la fois.

### II.2.3. Liaison télécommandée.

Une liaison télécommandée est une liaison établie entre deux processus donnés à l'initiative d'un troisième. Le problème dans ce cas là est de faire parvenir aux deux processus les paramètres de la liaison. En effet chacun ignore le centre et le numéro du processus avec lequel il va établir la liaison. C'est le troisième processus qui se chargera à l'aide de deux messages envoyés au moyen de liaisons avec chacun des processus de transmettre le numéro du premier ainsi que son centre au second et vice versa.

La figure II.2. regroupe les trois cas et les montre sur un exemple. Dans cette figure ainsi que dans les schémas suivants décrivant les actions effectuées par des processus nous nous servons du langage GSL [G3]. Il faut souligner que ce langage d'écriture de systèmes nous a servi pour l'implémentation qui sera décrite dans le chapitre IV.

```
     A                                  B

call RFP(B,CVRS,CREATE,)          Le processus b1 est
              retour=b1.           créé.
call LNKGET                       call LNKGET
       retour=va1                 retour=vb1
call RFL(b1,B,va1,)               call RFL(a1,A,vb1,)
       retour=0                   retour=0
```

La liaison initiale est établie.

```
     A                                  B

MESS=CODLIAISON
call SEND(va1,MESS)               call RCV(vb1,MESS')
call LNKGET                       call LNKGET
      retour=va2                  retour=vb2
call RFL(b1,B,va2,)               call RFL(a1,A,vb2,)
      retour=0                    retour=0
```

La seconde liaison est établie.

```
   A                    B                    C

call RFP(C,CVRS,CREATE,)                    Le processus c1
   retour=c1                               est créé.
call LNKGET                                    call LNKGET
    retour=va3                               retour=vc1
call RFL(c1,C,va3,)                            call RFL(a1,A,vc1,)
    retour=0                                 retour=0
```

La liaison initiale entre a1 et c1 est établie.

```
MESS1=(CODTL,C,c1)
MESS2=(CODTL,B,b1)
call SEND(va2,MESS1)  call RCV(vb2,MESS1')
call SEND(va3,MESS2)  call LNKGET          call RCV(vc1,MESS2')
                        retour=vb3         call LNKGET
                                            retour=vc2
                    call RFL(c1,C,vb3,)       call RFL(b1,B,vc2,)
                         retour=0             retour=0
                     La liaison commandée par a1 est établie
                     entre b1 et c1.
```

FIGURE II.2.

Nous avons présenté au cours du chapitre précédent aussi bien que dans celui-ci des outils et des mécanismes qui servent à faire communiquer des processus. Ainsi un processus donné peut déterminer avec qui il va communiquer (RFP). Pour cela il obtient du système réseau des outils nécessaires (LNKGET) et il établit une liaison avec son correspondant (RFL). Après cette étape d'initialisation la nature des communications dépend de la structure des programmes qui constituent les processus. Cependant il est un certain nombre de règles qui doivent être respectées.

## II.3. Mécanismes de communication.

Il est nécessaire pour avoir une communication entre deux processus que ces derniers se connaissent. L'établissement d'une liaison entre eux est nécessaire mais non suffisante. En effet nous avons vu qu'une liaison était bidirectionnelle, cette caractéristique étant induite par le caractère bidirectionnel des voies logiques qui la composent. Donc à chaque extrémité chacun des processus peut émettre ou recevoir, étant entendu qu'il ne peut effectuer qu'une seule opération à la fois. A une opération quelconque, "RCV" ou "SEND", doit correspondre l'opération complémentaire à l'autre extrémité; par conséquent il faut de plus qu'au moment où un processus est récepteur son correspondant soit émetteur. Le récepteur est caractérisé par la suite d'appels:

```
call RCV(v11,MESS,ECB,MSK)
call WAIT(ECB,MSK)
```

tandis que l'émetteur doit lancer l'appel suivant:

```
call SEND(v12,MESS).
```

Le point du programme où le processus récepteur émet l'appel WAIT est appelé point d'attente et le point où l'émetteur lance le SEND point de réveil. Le récepteur restera bloqué au point d'attente jusqu'à ce que l'événement qu'il attend, et qui est commandé du point de réveil, arrive. D'où la règle :

Il faut s'assurer que pour chaque point d'attente d'un processus il existe au moins un point de réveil correspondant.

Un point d'attente et un point de réveil sont dits correspondants quand les voies logiques du SEND et celle du RCV associé au même ECB que le WAIT constituent une liaison.

Cette règle permet d'éviter des situations où un processus est en attente indéfiniment et influe donc sur la structure des programmes qui constituent les processus. Un cas typique de blocage est le suivant, va1 et vb1 définissant une liaison:

```
call RCV(va1,MESS,ECB,MSK) ------ call RCV(vb1,MESS,ECB,MSK)
     call WAIT(ECB,MSK) ------------ call WAIT(ECB,MSK)
```

## II.4. Communication point à point ou 1-1.

Nous appelons communication du type 1-1 la situation où à un point d'attente correspond un seul point de réveil. L'exemple de la figure II.3. montre comment deux processus a1 et b1 communiquent. a1 lit des cartes ( call READ(RDR,MESS) ) et les envoie à b1 qui les imprime. Dans ce cas nous voyons bien qu'au point d'attente b1PA il correspond le point de réveil a1PR.

## II.5. Communication du type 1-n

Sur la base de la communication du type 1-1 peut être construite une communication plus compliquée: un processus se synchronise avec plusieurs autres processus. Une communication du type 1-n est caractérisée par la situation où à un point d'attente il correspond plusieurs points de réveil. Une liaison est définie par un couple de voies logiques. Il n'est pas question ici de faire correspondre à une voie un groupe de voies logiques, mais de faire correspondre à un point d'attente plusieurs "RCV" donc plusieurs "SEND" donc plusieurs points de réveil. Ceci est réalisé par une attente sur une liste d'événements. Le schéma II.4. montre un exemple où un processus est en communication avec quatre processus. C'est par exemple un allocateur de ressources. Suivant les demandes arrivant des processus éloignés, il obtient une clé (par call GETCLE), qu'il envoie à son correspondant, celui-ci ayant désormais accès à la ressource grâce à la clé.

Ce type de communication peut aussi permettre d'effectuer des opérations délicates du type mise à jour de catalogue. La séquence critique à laquelle ne doit accéder qu'un processus à la fois constituerait le processus en communication 1-n avec les autres et qui gérerait des compteurs et des verrous pour donner l'accès du catalogue à tel ou tel processus éloigné.

## II.6. Incidences sur le système-réseau.

Pour permettre que les deux types de communications s'effectuent normalement il faut faire certaines suppositions sur

le système-réseau ou sur des processus-système.

1. Il est clair que les communications du type 1-n ne peuvent se réaliser que si un mécanisme d'attente multiple est disponible (attente sur une liste d'événements). Un processus doit obtenir simultamément plusieurs liaisons donc plusieurs voies logiques.

2. Certaines conditions extérieures peuvent créer des situations de blocage. C'est notamment le cas d'un processus qui est à un point d'attente alors que le centre correspondant est déconnecté (panne de transmission .. ). Pour éviter cela l'interface de communication doit invalider toute liaison qui, non seulement aboutit à un centre déconnecté, ce qui peut être décelé localement, mais aussi à un processus qui a été détruit sans fermer la liaison, ce qui nécessite une action de l'interface correspondant. Le mécanisme suivant proposé pour ARPA par WALDEN [A7] permet d'éviter à un processus de rester en attente alors que son correspondant n'existe plus. C'est la demande correspondant à un "RCV" qui va vers le centre qui doit émettre le "SEND". A l'aide d'une table de rendez-vous, deux demandes "RCV" et "SEND" correspondantes sont identifiées avant d'envoyer les données vers le centre receveur. Ce mécanisme, s'il permet d'éviter certaines difficultés, présente malgré tout l'inconvénient de ralentir l'intéraction entre deux processus surtout si l'émetteur est prêt avant le récepteur.

```
        a1                                  b1
         :                                   :
         :                                   :
dcl 1 MESS,                         dcl 1 MESS,
     2 mesc char(1),                     2 mesc char(1),
     2 msg char(80);                     2 msg char(80);

ET1:call READ(rdr,addr(MESS),returns(cod));
    if cod≠0 then
     mesc=0; /* fin */
    else mesc=1;

a1PR:call SEND(va1,MESS)            ET1:call RCV(vb1,MESS,ECB,MSK);
    if mesc=1 then                  b1pa:call WAIT(ECB,MSK);
     goto et1;                         if mesc=0 then goto et2;
  call LNKCLS(va1)                   call WRITE(prt,msg)
                                       goto et1;
                                    ET2: call LNKCLS(vb1);
```

FIGURE II.3. Communication du type 1-1.

a1

```
dcl v1(5) fixed;
dcl mess(4) char(80);
dcl ECB(5) fixed;
do i=1 to 4;
call LNKGET(returns(v1(i));
ECB(i)=0;
end;
ECB(5)='ffffffff'x;
/* fin de la liste d'ECB */

do i=1 to 4;
call RCV(v1(i),mess(i),ECB(i),MSK)
end;

PA:call WAIT(addr(ECB));
/* point d'attente; le processus
 a1 est débloqué sur le premier
 message arrivé */
do i=1 to 4;
if ECB(i)=MSK then goto trouv;
/* on recherche quel est
 l'événement arrivé */
end;

trouv:call GETCLE(returns(b))
mess(i)=b; /* on envoie la clé */
call SEND(v1(i),mess(i))
```

b1

```
call SEND(vb1,messb)
call RCV(vb1,messb,ECBb,MSK)
call WAIT(ECBb,MSK)
```

---

c1

```
call SEND(vc1,messc)
call RCV(vc1,messc,ECBc,MSK)
call WAIT(ECBc,MSK)
```

---

d1

```
call SEND(vd1,messd)

call RCV(vd1,messd,ECBd,MSK)
call WAIT(ECBd,MSK)
```

---

e1

```
call SEND(ve1,messe)
call RCV(ve1,messe,ECBe,MSK)
call WAIT(ECBe,MSK)
```

FIGURE II.4. Communication du type 1-n.

# C H A P I T R E  3

## UNITES D'ENTREES/SORTIES DANS UN RESEAU D'ORDINATEURS.

Un des buts des réseaux d'ordinateurs est d'offrir aux utilisateurs d'un centre les mêmes possibilités qu'à ceux des autres centres. Cela permet en particulier aux centres dont la puissance de calcul est insuffisante pour leur demande locale de se décharger sur les centres du réseau capables d'absorber d'avantage de travaux. A la limite on peut concevoir qu'un des noeuds du réseau n'offre à ses utilisateurs que des moyens d'accès aux autres centres. C'est essentiellement le but d'un des aspects du réseau ARPA développé pour permettre des connexions directes avec le réseau au moyen de terminaux et appelé "TERMINAL IMP" [A1] .

L'idée d'offrir des services à une communauté très large d'utilisateurs par l'intermédiaire de centres divers peut paraitre très satisfaisante. Mais la diversité du matériel que l'on peut être amené à utiliser fait apparaitre certaines incompatibilités entre les calculateurs et les unités d'entrées/sorties. Cela se fait

particulièrement sentir dans les réseaux hétérogènes. Plusieurs solutions peuvent être proposées pour pallier ces difficultés:

1. Chaque ordinateur doit savoir à quel type de calculateur ou de périphérique éloigné il s'adresse. Il doit donc se conformer aux exigences de son interlocuteur.

2. Chaque destinataire, calculateur ou périphérique doit comprendre le langage des ordinateurs ou périphériques qui lui sont connectés. Pour utiliser les données qu'il reçoit, un système doit donc être capable de les traiter soit directement soit après les avoir traduites dans son propre code.

3. Tous les éléments du réseau -systèmes, programmes particuliers, routines de gestion des périphériques, etc..- appelés à utiliser des données provenant d'un autre centre comprennent un langage commun.

Les deux premières solutions présentent des inconvénients majeurs. Chaque centre en effet doit garder une trop grande masse d'informations. Ainsi un programme qui traite des données en provenance de terminaux de type machine à écrire doit posséder une table de traduction pour chaque modèle susceptible d'être connecté au réseau afin d'effectuer le transcodage avant le traitement. Si l'on a N points de service s'adressant à M destinations de types différents, il est alors indispensable d'avoir M x N procédures de traduction. Ces dernières se servent en général de tables assez encombrantes qu'il vaut mieux ne pas multiplier. Pour éviter toutes ces

difficultés il apparait nécessaire de passer par un intermédiaire qui ignore les caractéristiques des matériels. Cela nous amène à nous adresser à toute unité, périphérique ou point de service au travers d'une interface software. La contrainte imposée est que quelque soit le type de périphérique connecté au réseau l'interface soit capable de transformer les données obtenues à partir du périphérique en données codées d'une manière acceptable par les autres interfaces. On développe ainsi un code commun qui servira pour toute transaction entre une interface SOURCE et une interface PUITS. De cette manière pour N points de service et M destinations nous n'avons plus besoin que de M + N procédures de traduction. Ceci est réalisé par la troisième solution. Pour connecter un périphérique au réseau il est donc nécessaire d'avoir deux interfaces: la première est associée à l'unité , la seconde au programme qui reçoit les données ou qui les émet vers l'unité.

## I. NOTION D'UNITE VIRTUELLE.

La notion d'unité virtuelle est attachée le plus souvent à celle de machine virtuelle. Elle présente beaucoup d'avantages parmi lesquels nous trouvons la possibilité de programmer une unité virtuelle de la même manière que l'unité réelle du même type et de la simuler sur une unité d'un modèle totalement différent. Ainsi les programmes restent indépendants du matériel. C'est d'ailleurs une notion de plus en plus utilisée [ CP1- VM1- B1 ] . Parmi les exigences de l'utilisation de périphériques éloignés dans un réseau d'ordinateurs, certaines présentent les mêmes caractéristiques que les avantages des unités virtuelles. Cependant d'autres problèmes peuvent se poser, notamment en ce qui concerne ceux de

communication. Nous arrivons naturellement à une généralisation ou une adaptation de cette notion au contexte de réseau hétérogène d'ordinateurs pour inclure les problèmes de communication.

## I.1. Définition.

Une unité virtuelle est définie par la donnée de trois interfaces software appelées:

.interface SOURCE,

.interface PUITS,

.interface de contrôle.

Il est des cas ou l'interface de contrôle n'est pas nécessaire.

## I.2. Caractéristiques.

Les unités virtuelles (U.V.) diffèrent par leurs types et leurs attributs.

.Le type d'une unité virtuelle reflète celui du périphérique réel auquel elle sert d'interface. On distingue ainsi les différents types suivants:

-console : CONSL,

-imprimante : PRT,

-lecteur de cartes : RDR,

-perforateur de cartes : PUN,

-bande magnétique : TAPE.

Cette liste n'est pas limitative. Elle dépend des types d'unités supportées par les divers systèmes du réseau.

.Les attributs d'une U.V. servent à définir les centres entre lesquels l'échange d'informations s'effectue ainsi que le sens de

cet échange. Pour cela chaque unité virtuelle possède un attribut SOURCE et un attribut PUITS qui prennent leurs valeurs dans l'ensemble des noms des noeuds du réseau.

.Chaque unité virtuelle possède une liste de commandes ou d'opérations qui constitue en fait le protocole de conversation entre son interface SOURCE et son interface PUITS. La liste des commandes doit être assez complète pour couvrir toute opération qu'un système peut demander à une unité réelle. Cela veut dire que quelque soit l'opération qu'un système peut lancer sur une classe d'unités réelles il existe une commande dans la liste d'une unité virtuelle qui lui corresponde. Inversement toute commande du protocole de dialogue entre l'interface SOURCE et l'interface PUITS d'une unité virtuelle peut être traduite en une ou plusieurs opérations d'une unité réelle. Le choix des commandes est très important car il donne une description des unités en termes de fonctions. Il détermine ainsi l'indépendance de l'unité virtuelle par rapport aux unités réelles, ce qui est un des buts recherchés. Une U.V. servira à piloter des unités réelles d'une même classe. La classe des unités réelles est définie par le type de l'unité virtuelle. Ainsi toutes les unités réelles sont compatibles avec tous les ordinateurs pour peu que l'on fournisse les interfaces qui assurent cette compatibilité.

Les commandes dépendent du type des unités virtuelles. Cependant il existe un ensemble de commandes communes à tous les types. Ce sont les commandes de contrôle.

## I.3. Commandes de contrôle.

BLOQUER(sens). Cette commande est émise par l'une des interfaces pour bloquer le débit de la voie de communication dans un

sens. Seuls les messages de contrôle sont autorisés à emprunter la voie logique. Ceci est valable jusqu'à ce que l'ordre de débloquer le débit arrive.

DEBLOQUER(sens). Dès la réception de ce message de contrôle la voie logique bloquée est libérée.

PURGER (U.V.). Cette commande permet de supprimer les messages en attente. La place qu'ils occupaient peut être ainsi libérée.

SUPPRIMER (U.V.). L'unité virtuelle donnée en paramètre est supprimée par cet ordre. Ceci est fait après avoir exécuté l'ordre PURGE (U.V.).

DEMETAT: L'interface de contrôle demande des renseignements sur l'état de l'interface qui reçoit cet ordre. Ces renseignements portent sur le nombre de messages en attente et sur les voies de communication suspendues.

REPETAT(données): Cet ordre retourne les réponses à l'ordre DEMETAT reçu.

DEFCOD: L'interface de contrôle définit le code qui représente les caractères dans les messages entre les autres interfaces. Pour cela chacune reçoit cette commande de contrôle.

Pour les deux premiers ordres c'est l'interface réceptrice qui prend l'initiative de bloquer ou de débloquer le sens de la voie logique. Ainsi l'interface PUITS envoie à l'interface SOURCE l'ordre BLOQUER (S-P) pour arrêter le débit dans le sens SOURCE vers PUITS.

## I.4. Commandes spécifiques.

Chaque unité virtuelle a une liste de commandes spécifiques.

### I.4.1. Unité virtuelle de type CONSL.

La liste des commandes de l'unité virtuelle du type CONSL est la suivante:

DEMLECT: L'interface PUITS envoie à l'interface SOURCE cet ordre pour lui signifier que le point de service auquel la première est rattachée a émis une demande de lecture.

ECRIRERC(données): Par cette commande l'interface SOURCE reçoit des données à envoyer à l'unité réelle. L'opération d'écriture doit se faire en passant au début de la ligne suivante.

ECRIRNRC(données): C'est un ordre analogue au précédent. La seule différence est que l'écriture doit se terminer sans passer à la ligne suivante.

SUPIMPR: Quand cet ordre arrive le dispositif de suppression d'impression du terminal réel est enclenché au cas ou ce dernier en est muni. Sinon l'ordre est ignoré ou simulé.

RETIMP: L'impression est rétablie dès que cet ordre est reçu ce qui annule l'effet de la commande SUPIMPR.

ATTN: Quand l'interface PUITS reçoit cet ordre, elle transmet au point de service une condition d'attention si celui-ci est en état de la recevoir.

REPLECT(données): Par cette commande l'interface SOURCE transmet à l'interface complémentaire des données constituant une entrée au terminal. C'est une réponse à une commande

DEMLECT reçue.

I.4.2. Unité virtuelle de type PRT.

Les ordres qui permettent de piloter une imprimante sont:

ECRNCR(données): Ecrire les données reçues en paramètre sans sauter à la ligne suivante.

SAUT(n): Sauter n lignes.

PAGE: Sauter à la page suivante.

I.5. Composants d'une unité virtuelle.

Nous avons vu qu'une unité virtuelle était composée de trois parties: une interface SOURCE, une interface PUITS, et une interface de contrôle.

Les deux premières sont complémentaires et sont en relation soit avec un point de service soit avec un périphérique. L'interface de contrôle comme son nom l'indique sert à contrôler les deux autres parties. Prenons comme exemple un travail soumis dans le centre A, s'exécutant en B et produisant des sorties sur une imprimante en C. Pour que les données parviennent à leur destination sans que l'on ait à se préoccuper des caractéristiques de l'imprimante une unité virtuelle de type PRT dont l'attribut SOURCE a comme valeur B et l'attribut PUITS comme valeur C est nécessaire. Celle-ci sera constituée de trois parties: une interface SOURCE en B, une interface PUITS en C et une interface de contrôle en A. L'interface de contrôle peut se confondre avec l'une des deux autres parties. C'est le cas pour une unité virtuelle dont l'un des attributs est égal au centre où est soumis le travail. Cet

exemple est représenté par la figure III.1.

I.5.1. Caractéristiques des interfaces.

Chaque interface SOURCE ou PUITS possède les éléments suivants:

.Une voie logique de communication avec l'interface complémentaire.

.Une voie logique de communication avec l'interface de contrôle. Dans le cas où celle-ci est confondue avec l'une des interfaces SOURCE ou PUITS cette voie logique n'existe pas.

.Le code de représentation des caractères dans les messages.

.Les moyens de communiquer avec le périphérique ou le point de service.

.Le type de l'unité virtuelle.

.Le descripteur de la structure des données manipulées par l'unité.

.Le descripteur de l'état de l'unité.

Les deux voies logiques servent à acheminer les messages. La première transporte des messages qui résultent du dialogue entre les deux interfaces complémentaires. Chaque message comporte deux champs. Le premier contient le code de la commande et le second est constitué par ses paramètres éventuels. La seconde voie logique permet à l'interface de contrôle d'envoyer ou de recevoir des messages de contrôle à partir de l'interface avec laquelle elle est en communication.

Chaque interface garde trace du code qui lui permet de

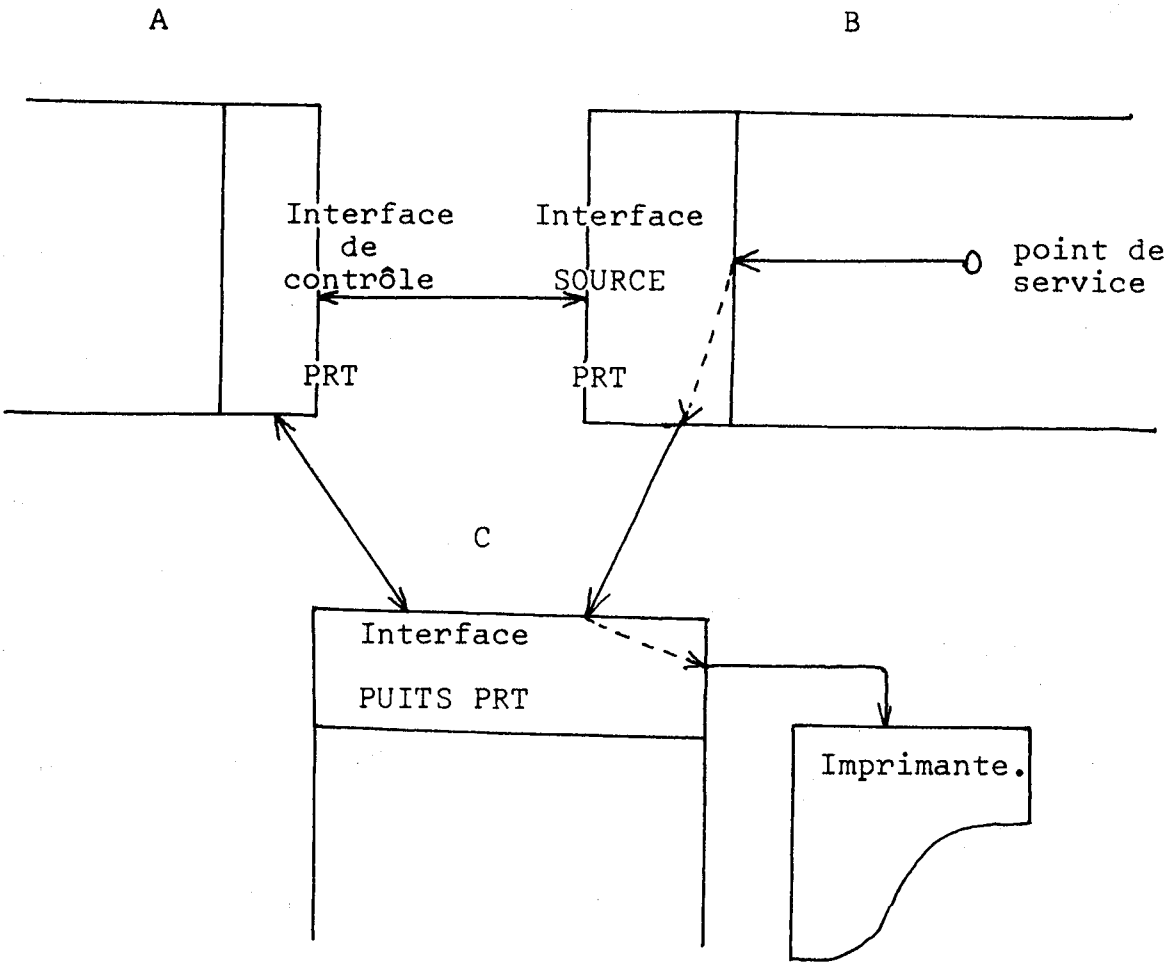



FIGURE III.1.

connaitre dans quelle représentation sont codés les caractères qui sont contenus dans le second champ du message. Les deux interfaces complémentaires de la même unité ont nécessairement le même code à un instant donné. Ce code peut être changé grâce à la commande DEFCOD. Les codes permis sont: ASCII, BCD, EBCDIC.

Les moyens de communication avec le périphérique ou le point de service permettent d'identifier un module appelé Module de Traduction d'Ordres (MTO). Le rôle de celui-ci est essentiellement un rôle d'interprétation des commandes de l'unité virtuelle.

Le descripteur de la structure des données manipulées par l'unité détermine la longueur de l'unité d'enregistrement. Il

peut donner aussi le nombre de lignes par page dans le cas d'imprimante ou d'unité à écran cathodique.

Le descripteur de l'état de l'unité contient des informations sur le nombre de messages reçus, le nombre de messages en attente ainsi que l'état des voies logiques (bloquées ou non).

## II. NOTION DE RESEAU VIRTUEL.

Nous allons définir une notion directement liée à celle d'utilisateur ou de travail et qui permet d'établir une relation entre des unités virtuelles.

### II.1. Définition.

Un réseau virtuel est un ensemble d'unités virtuelles définies pour le même utilisateur ou le même travail.

### II.2. Opérations sur un réseau virtuel.

Un réseau virtuel dépend de l'utilisation que l'on en fait. Les besoins de l'utilisateur varient dans le temps en ayant une répercution sur la structure du réseau virtuel qui lui est associé. Les changements nécessaires se font à l'aide des opérations d'insertion de suppression ou de modification des attributs des unités virtuelles. Ceci peut être fait soit à l'initiative de l'utilisateur quand celui-ci contrôle directement son travail - cas de l'utilisateur connecté à un système interactif - ou automatiquement par le système - allocation dynamique des ressources, désallocation quand l'utilisation d'un périphérique est terminée.

### II.3. Réseau virtuel interactif.

Nous appelons réseau virtuel intéractif, le réseau virtuel d'un utilisateur connecté à un système intéractif. Cette distinction est nécessaire du fait des caractéristiques d'un tel

réseau virtuel et du fait des contraintes imposées.

.Un R.V.I. minimum est constitué d'une unité virtuelle de type CONSL dont l'attribut PUITS a pour valeur le nom d'un centre du réseau supportant un système interactif.

.Les valeurs que peuvent prendre les attributs d'une U.V. doivent obéir à certaines règles. L'ensemble des noeuds du réseau est divisé en deux classes: la classe des noeuds à système conversationnel et la classe des noeuds à système batch respectivement notées NC et NB.

.La première règle que doivent suivre les attributs pour leurs valeurs se résume dans le tableau I.

.La seconde règle donne une relation que doivent vérifier les attributs d'unités virtuelles d'un même RVI. Les égalités suivantes définissent cette relation :

PUITS(CONSL)=SOURCE(PRT)=SOURCE(PUN)=PUITS(RDR).

PUITS(CONSL) par exemple signifie la valeur de l'attribut PUITS de l'unité de type CONSL du réseau virtuel considéré. Les valeurs des attributs de tous les types d'unités sauf celle de type CONSL peuvent varier. La valeur PUITS(CONSL) reste donc fixe. Etant celle à laquelle se refèrent les autres valeurs pour vérifier la seconde règle les changements d'attributs ne peuvent porter que sur les valeurs suivantes: PUITS(PRT), PUITS(PUN), SOURCE(RDR). La configuration minimum d'un RVI étant une U.V. de type CONSL c'est la première unité à créer. C'est aussi la dernière à supprimer. Sa suppression entraine automatiquement celle de toutes les autres unités donc celle du réseau virtuel.

| TYPE | ATTRIBUT | CLASSE DE VALEUR |
|---|---|---|
| CONSL | SOURCE | NB U NC |
| | PUITS | NC |
| PRT | SOURCE | NC |
| | PUITS | NB U NC |
| RDR | SOURCE | NB U NC |
| | PUITS | NC |
| PUN | SOURCE | NC |
| | PUITS | NB U NC |

TABLEAU I.

# III. RELATIONS AVEC LES UNITES REELLES ET LES POINTS DE SERVICE.

Une interface possède essentiellement deux points de connexion. D'une part elle est connectée à l'interface complémentaire et d'autre part elle est reliée soit à une unité réelle soit à un point de service. Nous avons vu qu'une unité virtuelle possédait parmi ses éléments une voie logique pour communiquer avec l'interface complémentaire; c'est donc une liaison, telle qu'elle a été définie au chapitre précédent qui réalise la première connexion. Quant à la manière dont est reliée une interface à une unité réelle ou à un point de service, elle dépend du type de l'unité ou du système auquel on a accès par ce point de service.

## III.1. Opération d'attachement.

Les deux interfaces SOURCE et PUITS d'une unité virtuelle précisent le chemin que doivent suivre les données: du centre SOURCE au centre PUITS. Ce chemin est logique et peut ne pas correspondre au chemin réel que suivront effectivement les informations, ce dernier étant déterminé par les procédures de communications. Ce que le système du centre puits va faire avec les données qui lui parviennent dépend de la fonction ATTACH précisée pour l'interface PUITS.

Avant de préciser le rôle de cette fonction il est nécessaire de dire quelques mots des ressources d'un centre. Dans cet ensemble nous trouvons des unités d'entrée/sorties des processus et des points de service. Le système reconnait les unités d'après leur type et leurs adresses. Les processus sont identifiés par un numéro que le système leur attribue au moment

de leur création. Quant aux points de service, c'est ce qui permet d'avoir accès au système local. C'est aussi un moyen d'interaction entre le système et une interface.

La fonction ATTACH permet d'attacher une ressource à une unité virtuelle. Ainsi les données reçues par l'interface PUITS sont dirigées vers l'unité réelle, le processus de traitement ou le système et l'interface SOURCE est reliée à la ressource qui doit lui transmettre les données.

Prenons un exemple pour illustrer cela: lors d'une connexion à un système conversationnel une unité de type CONSL est créée. L'interface SOURCE est attachée à une unité réelle de façon exclusive qui est le terminal de l'utilisateur. Dans le centre qui supporte le système conversationnel l'interface PUITS est créée puis attachée à une ressource du centre appelée "console-réseau". Celle-ci sera étudiée dans le chapitre IV et a pour rôle de donner l'accès au système conversationnel. La figure III.2. montre l'unité de type CONSL ainsi que ses relations avec le système conversationnel et le terminal.

L'opération d'attachement est une demande d'allocation de ressource. En effet, nous avons vu qu'un réseau virtuel pouvait être étendu à tout instant par l'adjonction d'une unité virtuelle. Donc à tout moment un processus peut demander qu'on attache une ressource à cette nouvelle unité. Il peut demander une ressource particulière auquel cas il doit passer par un allocateur auquel il précise l'adresse de l'unité, son type, son modèle etc (par exemple terminal 2741 d'adresse 04F). Il peut aussi demander les services d'une unité de manière non exclusive. C'est généralement le cas pour les unités telles que l'imprimante

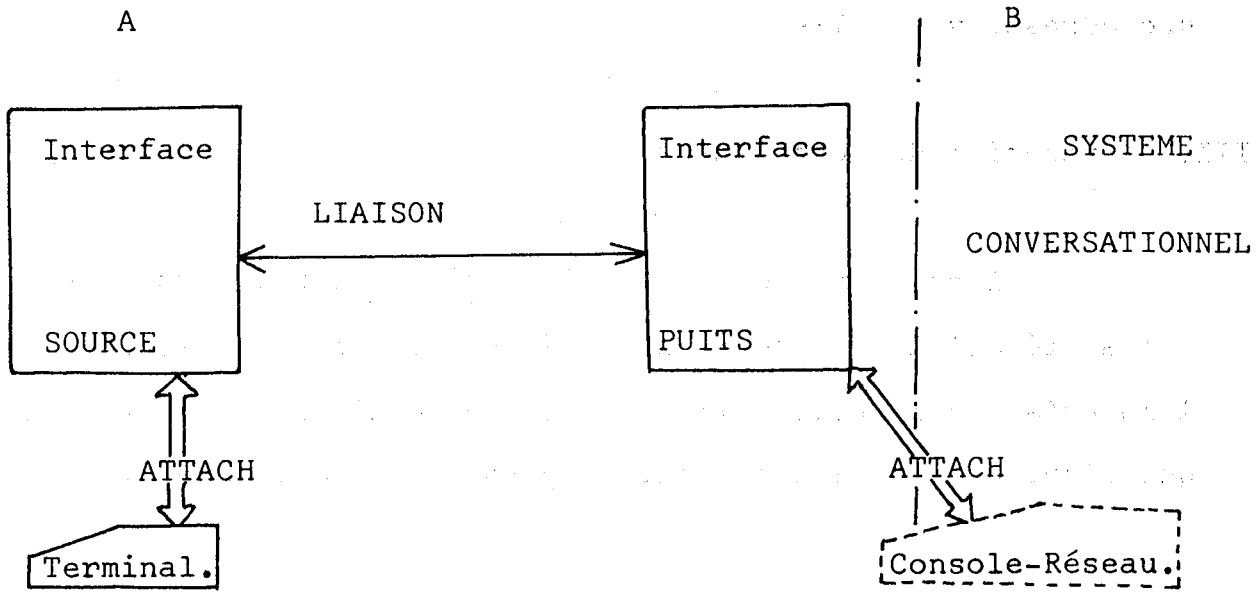



FIGURE III.2. Unité virtuelle du type CONSL et ses relations.

ou le perforateur de cartes. Dans ce cas précis l'interface PUITS est attachée à un programme qui réalisera les sorties en différé (SPOOLING).

Call ATTACH(interface,mode,modèle, adreelle )

Le mode peut être exclusif ou différé.
Le modèle peut prendre une des valeurs permises dans un centre suivant les unités disponibles. Par exemple l'appel peut préciser : IBM2741, TTY, IBM1403, IBM2540R, IBM2540P ...

Ces opérations d'attachement entre entité logique et ressource physique sont classiques et largement utilisées dans les systèmes. Ainsi la fonction ATTACH de MULTICS [ 01 ] permet de lier une unité réelle du système à une unité logique que l'utilisateur référence par son identificateur. La commande ATTACH de CP [CP1] permet d'allouer une unité réelle à un utilisateur de manière exclusive ou en partage et de lui affecter

une adresse virtuelle.

III.2. Opération de détachement.

Lorsqu'un processus a fini d'utiliser une ressource qui lui a été allouée il doit la libérer. Pour cela il la détache de l'interface à laquelle elle est reliée. Ceci doit être fait avant que l'unité virtuelle ne soit supprimée par la commande:

Call DETACH (interface).

Cette opération de détachement peut aussi être utilisée pour changer d'unité réelle en cours d'exécution. Lorsque l'utilisateur veut faire imprimer un fichier avec des caractères particuliers, il doit détacher l'interface PUITS de son unité virtuelle du type PRT de sa ressource. Ensuite il attache cette interface à l'unité réelle qui a la chaine de caractères appropriée avant de faire imprimer son fichier. Pour revenir à son état initial il doit effectuer successivement une opération de détachement et une opération d'attachement.

III.3. Organisation interne.

La fonction ATTACH permet aussi de choisir le Module de Traduction d'Ordres (M.T.O.) qui sera chargé de traduire les commandes de l'unité virtuelle. Il a été précisé plus haut que le protocole de dialogue entre les deux interfaces SOURCE et PUITS d'une unité virtuelle est totalement indépendant des modèles d'unités réelles. Pour produire les données qui parviennent le Module de Traduction d'Ordres est chargé de transformer les

commandes d'unité virtuelle en commandes spécifiques à l'unité qui lui est attachée. De ce fait il y a autant de M.T.O. que de modèles susceptibles d'être connectés à un centre donné. Le résultat d'une traduction d'ordres peut être soit des programmes canaux soit des appels à des fonctions. Dans le premier cas lorsque l'unité réelle est attachée de manière exclusive à l'unité virtuelle, le M.T.O. prépare le programme canal approprié pour faire effectuer les opérations d'entrées/sorties. Ensuite il appelle le superviseur d'entrées/sorties (I.O.S. dans le cas de l'OS/360 par une macro instruction EXCP) auquel il transmet le résultat de sa traduction. Dans le cas où l'unité est attachée à un point de service ou à un programme des appels sont faits à des fonctions réalisant la communication entre l'interface et le système ou la gestion de fichiers de 'spooling'. La figure III.3. montre comment le MTO sert d'intermédiaire entre une interface et le superviseur d'entrées/sorties ou les modules de gestion des fichiers de 'spooling'. C'est aussi le M.T.O qui reprend le contrôle quand l'opération s'est terminée. Les interruptions sont traitées par le superviseur d'entrées/sorties qui avertit le M.T.O. C'est donc à celui-ci de déterminer la façon dont doit s'effectuer l'opération qu'il initialise et en particulier s'il faut attendre ou non la fin de l'opération pour rendre le contrôle au processus qui l'a appelé.

Le dialogue entre le M.T.O. et le superviseur d'entrées/sorties se fait par l'intermédiaire des codes d'opération du programme canal que le premier transmet au second et des conditions de fin d'opération. La manière dont doit se faire ce dialogue dépend du superviseur et des spécifications que celui-ci impose aux utilisateurs. Un protocole analogue a été développé entre un M.T.O. et un point de service:

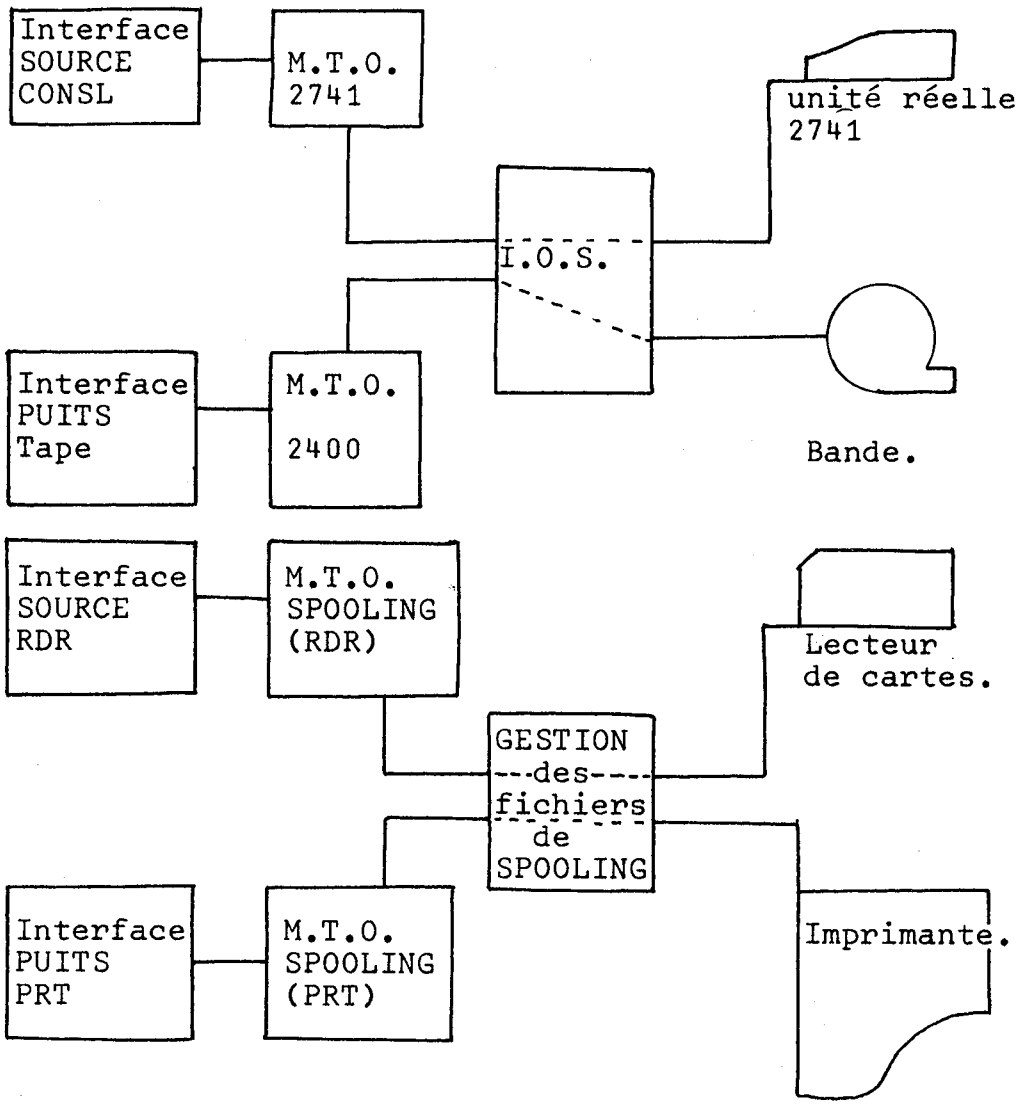



FIGURE III.3.

Le point de service s'adresse à un instant donné à un M.T.O. pour lui signifier des ordres pour une unité:

INIT: Le point de service indique au M.T.O. d'initialiser une opération.

PREP: Le M.T.O. est averti que le système est en état d'accepter une condition de type ATTENTION.

HALT: Le point de service demande au M.T.O. d'annuler l'opération en cours. Les données qui sont éventuellement gardées par lui deviennent caduques.

Le M.T.O. répond au système à l'aide des opérations suivantes:

ANALYSE: Cette commande demande des informations sur l'opération initialisée. L'analyse de ces informations produit des commandes de l'unité virtuelle utilisée.

FINOP(type,données,cond): Le M.T.O. indique au point de service qu'une opération est terminée. Le type indique une lecture ou une écriture. Les données de l'opération ainsi que les conditions dans lesquelles elle s'est effectuée sont précisées.

En conclusion il convient de souligner que l'organisation interne et les relations avec le superviseur d'entrées/sorties offrent bien des avantages: les programmes qui sont écrits dans un centre sont réutilisables quelle que soit l'évolution des unités d'entrées/sorties dans le centre local et surtout quels que soient les modèles des unités dans les centres éloignés. Chaque unité nouvelle pourra être utilisée dès l'instant que le Module de Traduction d'Ordres correspondant est disponible. De plus moyennant certaines conventions il existe une certaine équivalence entre toutes les unités de sortie ainsi qu'entre les unités d'entrées. Ainsi il est possible d'obtenir à partir d'un lecteur de cartes ou d'un fichier des entrées pour une unité virtuelle du type CONSL et produire ses sorties sur une imprimante. Pour cela il suffit d'aménager les M.T.O.

Il faut souligner aussi l'analogie entre cette organisation et celle du système d'entrées/sorties de MULTICS tel qu'il a été décrit par ORGANICK [01]. Le M.T.O. correspond au Device Interface Module de MULTICS et permet d'avoir toutes ses possibilités hormis la distribution d'une sortie entre plusieurs unités. Si la fonction ATTACH de MULTICS permet de lier un nom à une ou plusieurs unités, l'opération d'attachement décrite dans ce chapitre permet de lier une interface à une unité réelle ou à un point de service. Précisons qu'il faut deux opérations d'attachement par unité virtuelle (une pour l'interface SOURCE et une pour l'interface PUITS). A l'aide de ce moyen il est alors possible de relier une unité réelle à un programme éloigné.

# C H A P I T R E  4

# REALISATION PRATIQUE

Dans ce chapitre seront présentées les réalisations qui ont été faites sur les bases décrites dans les chapitres précédents. Les mécanismes du système S.O.C. ont servi de base à l'implémentation. D'autre part le système CP/CMS a été choisi comme exemple de système conversationnel à temps partagé. Les réalisations comprennent:

-Le processus conversationnel,

-L'implémentation du système-réseau sous CP ainsi que celle des entrée/sorties,

-un concentrateur de terminaux.

## I. LE PROCESSUS CONVERSATIONNEL.

Ce qui est désigné sous le nom de processus conversationnel est un ensemble de fonctions qui permettent

d'utiliser un système conversationnel éloigné. Cela comprend la gestion de toutes les unités qui assurent la communication entre le système et son interlocuteur dans le réseau. Ainsi un terminal situé dans un centre quelconque, par l'intermédiaire de ce processus, pourra servir de moyen de communication entre un système conversationnel éloigné et un utilisateur de la même manière que si le terminal était directement relié à l'ordinateur. Les principales fonctions assurées sont:

1. Transformation des caractères et édition élémentaire,
2. Etablissement de connexions entre les centres utilisés,
3. Transparence du programme et du réseau à l'utilisateur.

On distingue deux sortes de processus conversationnels: un processus conversationnel local situé dans le centre de l'utilisateur et un processus éloigné situé dans le centre du système conversationnel et qui sert de liaison entre l'utilisateur et le système. Chacun des processus est un élément du superviseur réseau. Ces programmes sont situés au niveau du système et ne sont donc pas modifiables ni programmables par l'utilisateur.

Il y a un processus conversationnel local par terminal susceptible d'être utilisé dans le réseau. Le processus est standard. Il ne dépend en aucune manière ni du centre dans lequel il se trouve ni du système conversationnel qu'il permet à l'utilisateur d'atteindre. Il offre un ensemble de facilités au moyen de commandes. Celle-ci sont interprétées par le processus

local qui selon les besoins de l'utilisateur, active éventuellement des programmes correspondants dans les centres concernés. Ces programmes sont appelés processus conversationnels éloignés ou serveurs. Il y a un processus conversationnel serveur par centre impliqué dans une utilisation conversationnelle.

Le processus éloigné est différent du processus local. Ceci est dû d'abord à ce que le premier ne gère pas de terminal réel. Il n'a donc pas à s'occuper des procédures d'entrées/sorties. Par contre il doit disposer de mécanismes permettant d'acheminer les données vers le système conversationnel. Ces mécanismes dépendent essentiellement de ce système et c'est à ce titre que le processus éloigné ne peut ignorer le système dont l'utilisateur se sert. Ainsi le processus éloigné peut être considéré comme ayant deux aspects:

1. Un aspect directement lié au système conversationnel. Cette partie dépend de la manière dont le système effectue ses entrées/sorties sur les terminaux et de la méthode utilisée pour simuler ces opérations ou pour les refléter dans le cadre du réseau. Ces mécanismes ont été présentés dans le chapitre précédent. Cet aspect dépendra donc de la manière dont ils seront implémentés dans chaque cas.

2. Un aspect standard qui lui dépend des conventions de connexion. Cette partie sert de correspondant au processus local. Le dialogue se fait entre ce programme et le processus local gérant le terminal réel selon un protocole bien établi qui entre dans le cadre général du protocole du réseau englobant ainsi les procédures de dialogues, le format des messages et les délais séparant les échanges.

## I.1. Le langage de commandes.

L'utilisateur interagit avec le processus qui gère son terminal au moyen d'un langage de commande. Ce langage doit être suffisant pour englober toutes les facilités dont pourrait avoir besoin une activité conversationnelle. Il doit ainsi permettre à l'utilisateur de se connecter au centre de son choix, de choisir les destinations de ses fichiers, de s'adapter aux variations des temps de réponse du système et d'obtenir un certain nombre de renseignements sur les éléments qu'il utilise. Ceci parmi les différentes facilités qu'il offre.

Les commandes de ce langage sont interprétées dans le centre local de l'utilisateur par le processus conversationnel utilisateur. Celui-ci est attaché à tout terminal du réseau et comprend outre les fonctions d'interprétation des commandes, des procédures d'entrées/sorties sur divers types de terminaux.

### I.1.1. Ensemble de caractères.

```
<caractere> ::= <lettre>|<chiffre>|<carspecial>
<lettre>    ::= A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|
                X|Y|Z|a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|
                u|v|w|x|y|z
<chiffre> ::= 0|1|2|3|4|5|6|7|8|9
<carspecial> ::= =|>|;|:|%|'|<|*|(|)|-|+|_|&|§|ç|"|!|°|,|.|à|
                 ?|/
```

I.1.2. Identificateur.

<identificateur> ::= <lettre> | <identificateur> <lettre> |
<identificateur> <chiffre>

Sémantique: Les identificateurs servent à désigner des fichiers, des centres. Un identificateur ne peut excéder huit caractères.

I.1.3. Nombres.

<nombres> ::= <decent> | <hexentier>
<hexentier> ::= <hexachiffre> | <hexentier> <hexachiffre>
<hexachiffre> ::= <chiffre> | <hexa>
<hexa> ::= A|B|C|D|E|F
<decentier> ::= <chiffre> | <decentier> <chiffre>
Sémantique: les nombres servent dans les commandes à désigner un processus ou dans les commandes d'état d'unités.

I.1.4. Commandes.

<commandes> ::= <connect command> |
<disconnect command> |
<escape command> |
<purge command> |
<stack command> |
<request command> |
<status command> |
<printer command> |
<reader command> |

<puncher command>|

<delete command>|

<attach command> |

<userid command> |

<listfile command>|

<free command> |

<tape command>

I.1.4.1. Commande CONNECT.

<connect command> ::= CONNECT <idsite> <pno> |

CONNECT <idsite>

<idsite> ::= IMAG

<pno> ::= <hexachiffre> <hexachiffre>

Sémantique: La commande CONNECT sert à indiquer au processus conversationnel le centre auquel on désire se connnecter. Dans le cas où la connexion existe déjà, on donne deux chiffres hexadécimaux qui identifient la connexion. Si on désire obtenir une nouvelle connexion on donne seulement l'identification du centre. Le système répond alors en donnant les deux chiffres caractérisant la nouvelle connexion.

I.1.4.2. Commande DISCONNECT.

<disconnect command> ::= DISCON | DISCON OFF

Sémantique : Avec la commande DISCONNECT l'utilisateur suspend temporairement ou définitivement la connexion courante.

I.1.4.3. Commande ESCAPE.

<escape command> ::= ESCAPE | ESCAPE <carspecial>

Sémantique : La commande ESCAPE permet à l'utilisateur de définir un caractère appelé caractère d'échappement. Ce caractère est utilisé pour passer du mode ELOIGNE au mode LOCAL. L'omission du paramètre a pour effet de préciser à l'utilisateur le caractère d'échappement courant.

I.1.4.4. Commande PURGE.

<purge command>::= PURGE <device>

<device> ::= READER | CONSOLE | PRINTER | PUNCHER | TAPE

Sémantique: La commande PURGE permet à l'utilisateur de purger ses unités des enregistrements ou des fichiers qui leurs sont rattachés.

I.1.4.5. Commande STACK.

<stack command> ::= STACK

Sémantique: Quand l'utilisateur veut anticiper les lectures sur son terminal il lance la commande STACK qui le fait passer dans le mode RESERVOIR.

I.1.4.6. Commande REQUEST.

<request command> ::= REQUEST

Sémantique: La commande REQUEST soumet le terminal au contrôle du système conversationnel.

I.1.4.7. Commande STATUS.

<status command> ::= STATUS | STATUS <device>

Sémantique: La commande STATUS permet d'interroger le processus conversationnel sur l'état des connexions de l'utilisateur et de ses unités. Celui-ci répond en indiquant les connexions actives et les fichiers rattachés aux différentes unités.

I.1.4.8. Commande PRINTER.

<printer command> ::= PRINTER SINK= <skce> SOURCE= <soce>
<soce> ::= IMAG
<skce> ::= CS | BOULOG | CIRCE | MINES

Sémantique: A l'aide des commandes PRINTER, PUNCHER et READER l'utilisateur a la possibilité de définir ou de redéfinir le centre origine et le centre destination des fichiers à imprimer à perforer ou à lire.

I.1.4.9. Commande PUNCHER.

<puncher command> ::= PUNCHER SINK= <skce> SOURCE= <soce>

I.1.4.10. Commande READER.

<reader command> ::= READER SINK= <soce> SOURCE= <skce>

I.1.4.11. Commande DELETE.

<delete command> ::= DELETE | DELETE <lincharsp>
<lincharsp> ::= <linchar> <charspecial>
<linchar> ::= LINE | CHARACTER

Sémantique: Cette commande permet soit de définir les charactères qui servent à annuler une ligne ou un caractère, soit d'interroger le système sur ces caractères.

I.1.4.12. Commande ATTACH.

<attach command> ::= ATTACH <device> TO <addrfile>
<addrfile> ::= <hexachiffre><hexachiffre><hexachiffre> |
<identificateur><troichiff>
<troichiff> ::= <chiffre><chiffre><chiffre>

Sémantique: La commande ATTACH permet d'attacher l'interface locale de l'unité virtuelle spécifiée en paramètre à une unité réelle, au spooling du système ou à un fichier. Quand le second paramètre est une adresse réelle le système vérifie s'il est possible d'allouer l'unité au processus conversationnel qui a émis la commande.

I.1.4.13. Commande USERID.

<userid command> ::= USERID<identificateur>

Sémantique: Cette commande est destinée à préciser au système le destinataire d'un fichier. A la suite d'une commande ATTACH READER TO nomfich typfich, le système s'attend à trouver en tête du fichier nomfich typfich une carte ayant le format: ID identificateur . Si cette carte est manquante ou si le système éloigné trouve que l'identificateur est invalide l'utilisateur en est averti et doit répondre par une commande USERID.

I.1.4.14. Commande LISTFILE.

<listfile command> ::= LISTFILE <idetoile><troichifet>
<idetoile> ::= <identificateur>|*
<troichifet> ::= troichiff>|*

Sémantique: Cette commande permet à l'utilisateur de savoir quels sont les fichiers que la gestion des fichiers de spooling lui permet d'atteindre.

I.1.4.15. Commande FREE.

<free command> ::= FREE <device>

Sémantique: Lorsque l'utilisateur pense ne plus utiliser les services d'une unité il émet la commande FREE pour la libérer. Ceci a pour effet de restituer la ressource attachée à cette unité virtuelle au système-réseau.

I.1.4.16. Commande TAPE.

<tape command> ::= TAPE SINK=<sosk> SOURCE=<sosk>

<sosk> ::= <soce>|<skce>

Sémantique: La commande TAPE définit pour le processus conversationnel les centres entre lesquels des données pour une bande magnétique sont transmises.

I.2. Connexion à un système conversationnel vue par l'utilisateur.

Le jeu de commandes définies précédemment permet d'utiliser un système conversationnel éloigné. Dès la commande CONNECT le réseau virtuel intéractif de l'utilisateur est créé. Il sera élargi ou réduit selon les besoins au cours de la session. C'est dans ce but que sont conçues les commandes PRINTER,READER,PUNCHER,TAPE,FREE.

Le terminal est un outil privilégié. Par cet intermédiaire l'utilisateur s'addresse aussi bien au système éloigné qu'au processus local. Le mode associé au terminal à un instant donné sert à faire la distinction nécessaire. Celui-ci est dit en mode LOCAL ou ELOIGNE suivant qu'il s'adresse au système local ou à son interlocuteur éloigné. Cela se traduit par des possibilités distinctes dans chacun des deux modes. La FIGURE IV.1. présente les différents modes ainsi que leurs relations.

I.2.1. Le mode LOCAL.

Quand le terminal est en mode LOCAL toutes les

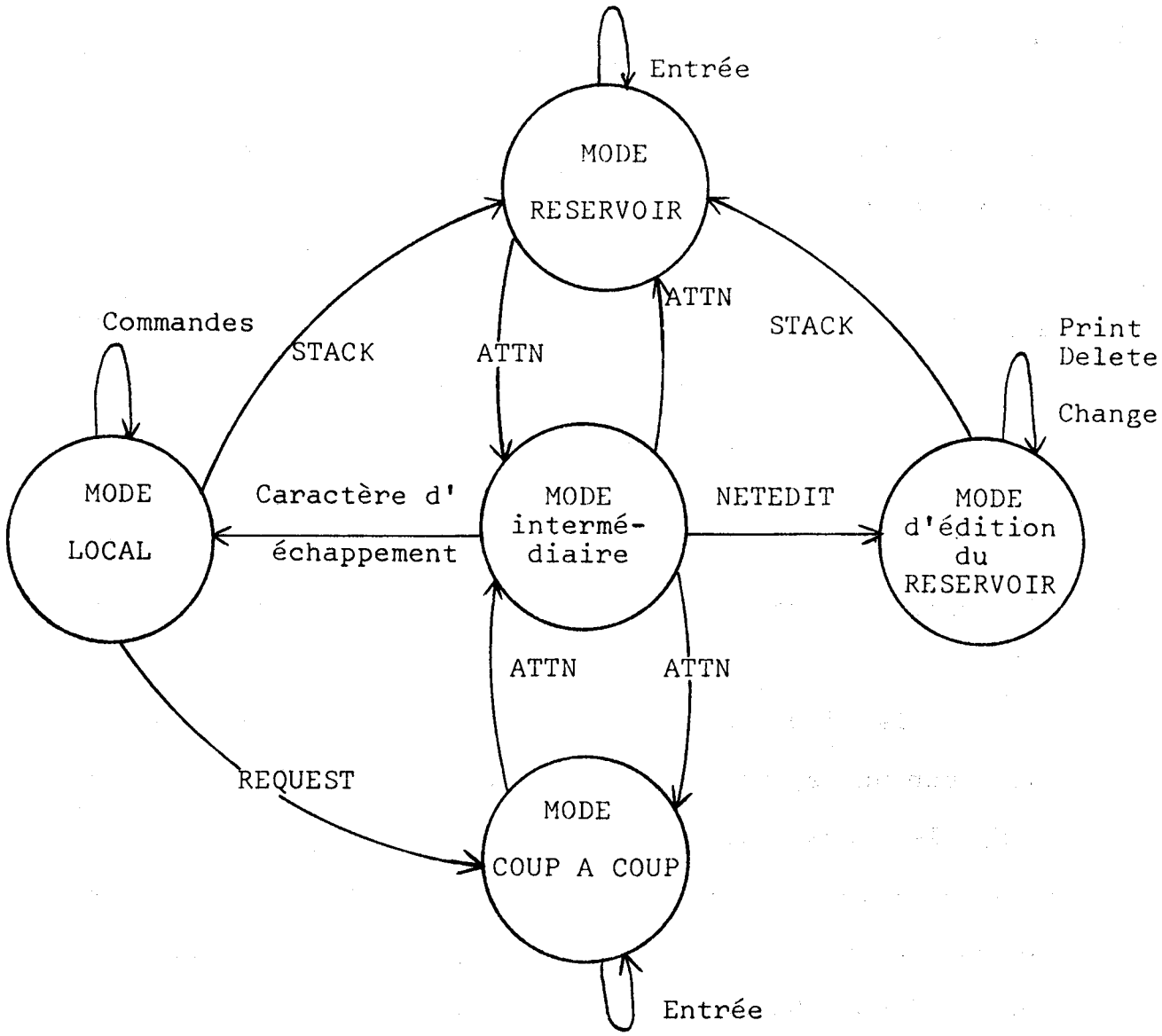



FIGURE IV.1 Les MODES et leurs relations.

commandes définies plus haut sont disponibles. Le programme ayant le contrôle à cet instant est chargé d'interpréter les commandes. Toutes celles-ci sauf STACK et REQUEST conservent le mode du terminal.

I.2.2. Le mode ELOIGNE.

Les deux commandes STACK et REQUEST font passer le

terminal du mode LOCAL au mode ELOIGNE qui se divise lui-même en deux sous-modes: le mode RESERVOIR et le mode COUP A COUP. Dans le mode ELOIGNE toutes les lignes frappées au terminal sont destinées au système éloigné auquel celui-ci est connecté. Ce sont donc soit des commandes de ce système soit des données. Seule l'attention doit être répétée pour être considérée comme destinée au système éloigné.

L'attention émise sous le mode ELOIGNE fait quitter celui-ci pour un mode intermédiaire dans lequel l'utilisateur doit spécifier s'il veut envoyer une ATTENTION au centre éloigné ou quitter le mode courant pour le mode LOCAL. Dans le premier cas il appuie une seconde fois sur la touche ATTENTION ce qui le fait revenir en outre au mode précédent. Dans le second cas il émet le caractère d'échappement. Nous avons préféré ce moyen qui consiste à donner une signification locale à l'ATTENTION car celle-ci est traitée rapidement. Le caractère d'échappement émis en mode ELOIGNE aurait pu constituer un moyen de changer de mode mais au prix d'une interprétation systématique de toutes les entrées sous les modes RESERVOIR et COUP A COUP.

### I.2.2.1. Le mode COUP A COUP.

Dans ce mode les entrées/sorties du terminal sont directement commandées par le centre éloigné. Tant que le système conversationnel n'a pas lancé une lecture le clavier reste bloqué. Quand celui-ci lance une lecture, la demande est enregistrée puis envoyée au centre local qui obtient les données du terminal. Celles-ci sont retournées au centre éloigné. Dans ce mode le réseau est parfaitement transparent à l'utilisateur.

Si Tm est le temps moyen de transmission d'un bloc

élémentaire, si tm est le temps moyen de traitement de ce bloc dans un centre intermédiaire et si n est le nombre de centres que doit traverser un message avant d'atteindre sa destination le temps supplémentaire nécessité par une lecture en mode COUP A COUP est :

$$T = 2 * ( n (Tm + tm) + tm )$$

Pour n=1 Tm=1s tm=1ms, T est sensiblement égal à 2s.

Dans le cas d'une écriture le temps supplémentaire est T/2. Le temps de réponse d'un système conversationnel étant celui qui sépare une requête de sa réponse, le temps de réponse d'un système conversationnel dans le réseau est majoré de 3T/2. Quand le nombre de centres intermédiaires est grand la majoration risque d'être importante. Ce qui peut enlever à un système tout son intérêt. Pour remédier à cet inconvénient le mode RESERVOIR a été introduit.

I.2.2.2. Le mode RESERVOIR.

Le mode RESERVOIR permet de se libérer des délais de transmission en anticipant les entrée/sorties. Plusieurs messages destinés au système éloigné sont ainsi empilés et envoyés en bloc vers leur destination où ils seront traités normalement. Le processus conversationnel serveur sachant quel est le mode courant du terminal puisera dans la pile au fur et à mesure que le système conversationnel lancera des lectures. Ceci est particulièrement utile quand l'utilisateur n'a pas besoin d'attendre les résultats du traitement de sa ligne. C'est le cas où celui-ci crée un fichier, le traitement d'une ligne consiste

uniquement à l'insérer. De même l'utilisateur attendant la fin d'un assemblage peut préparer le travail suivant. La pile créée dans ce mode peut être éditée dans le mode intermédiaire. Un éditeur élémentaire permet de faire imprimer le contenu de la pile de supprimer des lignes ou de les changer.

## I.3. Aspect interne.

L'interaction entre le système et l'utilisateur donne naissance à des échanges d'informations entre plusieurs centres du réseau. Que ce soit des messages ou des fichiers destinés à l'utilisateur ou au système ces flôts doivent être contrôlés pour savoir d'où proviennent les informations à qui elles sont destinées et comment elles ont été produites. Ce contrôle est basé sur les notions de réseaux et d'unités virtuels déjà vues au chapitre précédent. La figure IV.2. montre les moyens mis en oeuvre pour accéder à un système conversationnel. La structure du processus conversationnel dépend des commandes émises par l'utilisateur. Toutefois elle comprend des interfaces d'unités virtuelles constituant le réseau virtuel interactif de l'utilisateur. Le processus conversationnel serveur comprend toutes les interfaces d'unités dont l'un des attributs est le nom du centre serveur. Le processus conversationnel utilisateur comprend en plus les interfaces de contrôle quand les deux attributs d'une unité virtuelle sont différents du centre de l'utilisateur.

Avant toute demande de connexion, le processus conversationnel utilisateur se réduit à l'interpréteur de commandes en relation avec le concentrateur de terminaux d'où il obtient les commandes.

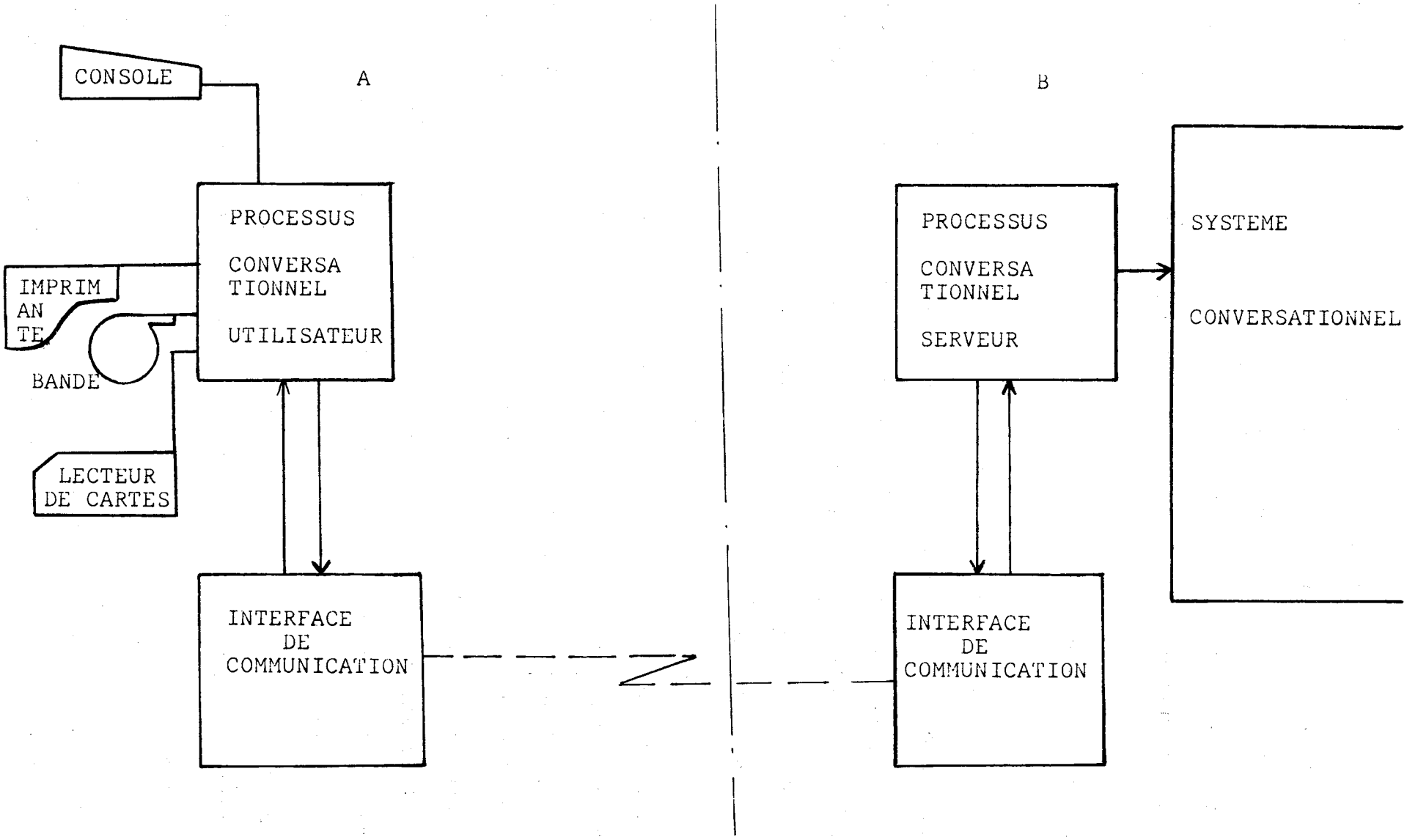



FIGURE IV.2. Connexion à un système conversationnel.

La commande CONNECT obtient un processus correspondant qui est un processus conversationnel serveur. Une unité virtuelle du type CONSL est alors créée ce qui se traduit par la création d'une interface SOURCE dans le centre local et une interface PUITS dans le centre éloigné. Ces interfaces sont ensuite mises en communication. Chacun des processus auxquels elles sont raccrochées fait un appel à la fonction RFL pour établir la liaison.

Les autres commandes qui définissent des unités virtuelles ne demandent pas de processus correspondant tant que les attributs de la nouvelle unité n'ajoutent pas de nouveau centre à ceux déjà référencés par le réseau virtuel. Par contre un processus peut être détruit par une opération de changement d'attribut ou de suppression d'unité. Les liaisons avec ce processus sont alors toutes supprimées. Pour illustrer l'évolution d'un réseau virtuel interactif nous allons voir dans ce qui suit les opérations effectuées par le système-réseau à la suite de commandes de l'utilisateur :

| Commandes utilisateur | Action des processus. |
|---|---|
| CONNECT IMAG | Call RFP(IMAG,CONV,CREATE,) |
| : | retour=RP1 |
| : | Call LNKGET |
| : | retour=vl1 voie logique |
| : | pour unité du type CONSL. |
| : | Call RFL(IMAG,RP1,vl1) |
| : | retour=0 voie opérationnelle. |
| : | : |
| PRINTER SOURCE=IMAG SINK=CS | Call LNKGET |
| : | retour=vl2 voie logique |
| : | pour unité du type PRT. |
| : | Call RFL(IMAG,RP1,vl2) |
| : | retour=0 |
| : | : |
| PUN SOURCE=IMAG SINK=BOULOG | Call RFP(BOULOG,CONV,CREATE) |
| : | retour=RP2. |
| : | Call LNKGET |
| : | retour=vl3 voie logique |
| : | pour unité du type PUN |
| : | Call RFL(BOULOG,RP2,vl3) |
| : | retour=0 |
| : | Call SEND(vl1,'RP2') |
| : | Call SEND(vl3,'RP1') |
| : | envoi des numéros de processus |
| : | pour établir la liaison entre |
| : | RP1 et RP2 |
| : | : |
| DISCONNECT OFF | Call LNKCLS(vl3) liaison |
| : | pour PUN détruite. |
| : | Call LNKCLS(vl2) liaison |
| : | pour PRT détruite. |
| : | Call LNKCLS(vl1) liaison |
| : | pour CONSL détruite. |
| : | Call MOURIR. |

## II. IMPLEMENTATION DU SYSTEME RESEAU SOUS CP.

L'implémentation du système-réseau dans les différentes installations du réseau SOC s'est toujours effectuée dans le souci de modifier le moins possible le système d'exploitation local. Ainsi dans un centre utilisant le système OS/360 le système-réseau a été utilisé en tant que tâche sous le contrôle de ce système. De même cette position a été adoptée vis à vis de l'hyperviseur CP [CP3] . Le système-réseau a été implémenté en tant que tâche d'un O.S/PCP lui même système d'une machine virtuelle.

Pour exposer les principes utilisés nous allons au préalable faire certains rappels sur le système CP/CMS.

### II.1. Le système CP.

Ce système est basé sur la notion de machine virtuelle et sur une technique de partage de temps. Plusieurs utilisateurs peuvent se servir simultanément du calculateur indépendamment les uns des autres. Cette indépendance est assurée par la génération d'une machine virtuelle par utilisateur. Celui-ci à partir de son terminal qui représente la console du calculateur a accès à toutes les fonctions d'un ordinateur.

Le système étant basé sur une technique de partage de temps entre les utilisateurs, une machine virtuelle utilise l'unité centrale pendant une tranche de temps au terme de laquelle le contrôle est donné à une autre machine. Si l'unité centrale est allouée périodiquement, les unités d'entrées/sorties ainsi que la mémoire centrale sont affectées de manière différente.

Le partage de la mémoire centrale est réalisé par un dispositif de mémoire virtuelle. Quant aux unités périphériques leur mode de partage dépend de leurs caractéristiques. Ainsi un disque, unité à accès direct, sera divisé en un certain nombre de "mini-disques". A chaque machine virtuelle est attribuée une ou plusieurs de ces unités. Certaines unités seront affectées tour à tour aux utilisateurs en anticipant ou en différant l'utilisation de ces unités. C'est précisément le cas des unités dites lentes telles que les lecteurs de cartes, les perforateurs de cartes ou les imprimantes. Un paquet de cartes sera donc lu et gardé sur disque par le système qui le restituera enregistrement par enregistrement au lecteur virtuel de la machine au fur et à mesure qu'elle le demande. De même les données produites par une imprimante ou un perforateur virtuel transiteront par un disque avant d'utiliser les unités du système. La technique ainsi décrite s'appelle le SPOOLING (Simultaneous Peripherical Operation On Line).

D'autre part il existe des unités qui ne peuvent être utilisées simultanément par plusieurs machines. C'est le cas des terminaux, des bandes magnétiques et des écrans cathodiques, par exemple, qui seront affectés exclusivement à une machine virtuelle.

Une machine virtuelle se présente alors de la même manière qu'une machine réelle. Elle dispose de mémoire, d'unités d'entrées/sorties ainsi que d'un jeu d'instructions identique à celui d'un 360 standard. La distinction essentielle entre une machine virtuelle et une machine réelle réside dans la différence entre leurs comportements en face d'opérations dépendant du temps.

II.1.1. Représentation d'une machine virtuelle dans le système.

Une machine virtuelle est représentée dans le programme de contrôle CP par une table (UTABLE). Cette table contient toutes les informations sur l'état de la machine. Tous les blocs d'information relatifs à ses ressources sont aussi rattachés a cette table. Elle contient entre autre:

-le PSW quand cette machine est interrompue.

-les interruptions pendantes sur ses unités d'E/S.

-les pointeurs sur les blocs décrivant les canaux et les unités d'E/S de cette machine.

-les pointeurs sur les requêtes d'E/S.

Toute unité virtuelle de cette machine est décrite par un bloc (MVDEBLOK quand cette unité est sur le canal multiple, VDEVBLOK autrement). La console d'une machine virtuelle est attachée à son canal multiple. Elle est donc représentée par un MVDEBLOK.

A chaque terminal rattaché au calculateur est associée une table décrivant son état à tout moment (MRDEBLOK). Quand un nouvel utilisateur joint le système, celui-ci crée les tables qui représenteront la configuration de sa machine. Ainsi il crée entre autre chose une table (MVDEBLOK) représentant sa console virtuelle. Jusqu'à la fin de la session -c'est à dire à l'émission d'une des deux commandes LOGOUT ou DISCONNECT-, les blocs MVDEBLOK de la console virtuelle et MRDEBLOK du terminal réel sont chainés. (voir la FIGURE IV.3.)

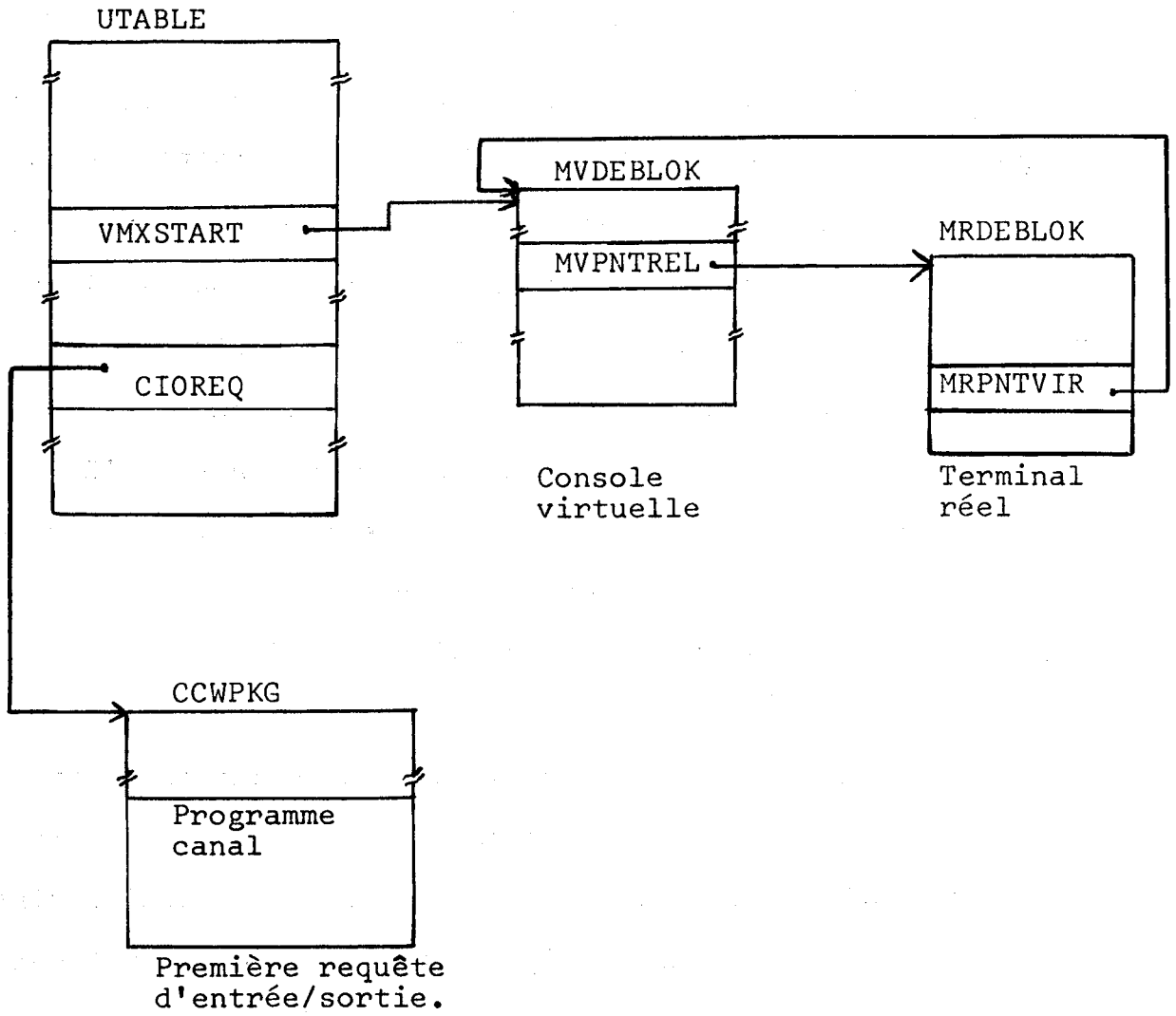



FIGURE IV.3. Chainage des éléments de description d'une console virtuelle et du terminal réel associé; requête d'entrée/sortie pour une console virtuelle.

II.1.2. Console virtuelle.

Un des buts que nous nous sommes fixés dans la réalisation du processus conversationnel est de pouvoir se connecter à un système conversationnel au moyen d'un terminal situé dans n'importe quel centre du réseau. Afin d'utiliser CP il est nécessaire d'étudier la manière dont il effectue les entrées/sorties sur les terminaux à l'aide desquels il interagit

avec les utilisateurs.

Toute opération d'E/S sur la console qu'elle soit commandée par CP ou par le système supporté par la machine virtuelle (opération sur un 1052 virtuel) aboutit au même mécanisme:

-construction d'un CCWPKG dans lequel est construite la chaine des CCWs. Ce bloc représente une requête qui sera chainée à la UTABLE (CIOREQ contenant la première requête). la figure IV.3. décrit ce chainage.

-enregistrement ou initialisation de la requête. Si aucune requête n'est en attente l'opération est aussitôt lancée. Dans le cas contraire, la demande est enregistrée à la suite des requêtes non encore satisfaites.

-interruption. Toute opération se terminant provoque une interruption. L'analyse faite ensuite permet de déterminer les conditions dans lesquelles l'opération s'est terminée. Au cas où la chaine des opérations en attente n'est pas vide, la requête suivante est aussitôt lancée.

II.1.3. L'interface entre CP et le réseau.

Dans le cas de l'utilisation de CP dans un réseau d'ordinateurs, les conditions sont différentes. Nous avons une machine virtuelle sous le contrôle de CP dont la console virtuelle est liée à un terminal qui se trouve dans un autre centre. Ce terminal ne peut être adressé directement par l'ordinateur car il n'est attaché à aucun de ses canaux. Il en est de même pour les unités telles que les perforateurs de cartes, les lecteurs de cartes et les imprimantes des autres centres. Les demandes d'entrées/sorties doivent alors être

transmises au centre éloigné par un mécanisme intermédiaire qui ne doit en aucun cas altérer celui utilisé par CP pour effectuer les entrées/sorties sur les unités locales. Ce mécanisme intermédiaire qui fait partie de l'interface entre CP et le reste du réseau est implanté dans une machine virtuelle que nous appellerons dans la suite MVS (Machine Virtuelle Spécialisée dans les fonctions du réseau).

II.1.3.1. La machine MVS.

Le système CP génère donc une machine virtuelle destinée à servir de support à toutes les fonctions permettant d'utiliser le réseau. Nous avons préféré cette solution plutôt que celle qui consiste à insérer ces diverses fonctions dans les programmes du système pour plusieurs raisons avantageuses.

Le code que le calculateur exécute pour une machine virtuelle se déroule toujours en mode programme, ce qui réalise automatiquement la protection du système par rapport aux fonctions ajoutées pour supporter le réseau. Ceci est particulièrement utile en période de tests. Les modifications apportées au système sont ainsi réduites au minimum. Elles se réduisent en effet à quelques instructions qui permettent d'adapter les programmes qu'exécute la machine MVS au divers mécanismes de CP. Ce sont essentiellement des moyens de communications entre MVS et CP. Tout le reste du temps nous nous efforçons d'utiliser les possibilités du système sans le modifier. Un autre avantage réside dans la possibilité d'utiliser les fonctions du système-réseau développées pour l'O.S. sans modifications majeures. La séparation ainsi obtenue entre les fonctions propres au système-réseau et celles de l'hyperviseur CP

permet d'être indépendant des futurs développements de ce dernier et rend les modules du système-réseau portables, ce qui constitue un avantage dans un environnement de réseau d'ordinateurs, l'implémentation devant être faite dans plusieurs installations différentes. Nous retrouvons toutes ces considérations dans les facteurs qui ont influencé l'étude de base du réseau de systèmes T.S.S. [R2] .

Parmi les inconvénients par rapport à une intégration des fonctions du réseau dans le système nous pouvons en souligner deux. D'une part la machine fonctionnant en mode programme, ses opérations d'entrées/sorties sont vérifiées et les programmes canaux compilés par CP avant que celui-ci ne lance véritablement les opérations. Ce travail de vérification parait inutile car les entrées/sorties étant préparées par le système de MVS on peut espérer que les programmes canaux sont correctement construits. Une fois ce système mis au point l'exécution de ces opérations par DIAGNOSE peut être envisagée. CP lancerait alors les opérations telles qu'elles lui sont transmises après vérification des paramètres. Cette méthode - analogue à celle déjà réalisée pour effectuer des entrées/sorties pour le système CMS [LH1] - imposerait une structure fixe de programmes canaux afin de réduire la complexité de leur compilation. D'autre part l'activation de l'interface se fait de manière périodique, ce qui risque d'apporter un certain retard dans les opérations. Ceci peut être en partie résolu en accordant à MVS une priorité plus grande que celles des autres machines virtuelles gérées par le système ce qui est justifié par le fait qu'elle fait plus partie du système que des programmes. De plus le travail de MVS est essentiellement du télétraitement dirigé par interruption.

Le système de MVS est un OS/PCP. Il a été choisi car il

comprend les fonctions nécessaires pour utiliser le superviseur du système-réseau telles que une gestion de mémoire, une gestion du temps et surtout un superviseur d'entrées/sorties. Cela nous a d'une part évité de développer spécialement un système pour MVS capable d'effectuer ces fonctions et d'autre part comme il a été souligné plus haut, cela nous a permis d'utiliser les processus développés ailleurs pour le système O.S. tels que PO, l'interface de communication, le processus conversationnel, ainsi que les modules de multiprogrammation et de gestion des processus.

La configuration de MVS est présentée dans la figure IV.4. Elle comprend entre autre des unités du type 1050, des lecteurs de cartes et des perforateurs de cartes virtuels. Ces unités servent dans les divers mécanismes de connexion avec les autres centres.

Pour rendre CP disponible dans le réseau, il faut donc activer MVS et démarrer un travail qui comprend les divers composants du S.S.R. Au cours de l'exploitation les demandes de connexion arrivent, ce qui se traduit chaque fois par la création d'un processus conversationnel serveur auquel on associe une des unités du type 1050. Cette unité accompagne le processus jusqu'à sa suppression. Cette procédure fixe le nombre de processus conversationnels qui peuvent être gérés simultanément par MVS. Quand il n'y a plus d'unité libre la demande de connexion est refusée.

Les moyens de communication entre MVS et CP sont de deux sortes. Sur l'initiative de MVS des données sont transmises à CP grâce à une instruction DIAGNOSE privilégiée donc immédiatement interceptée par le programme de contrôle [CP2] . En créant une interruption sur une des unités virtuelles de MVS le système lui transmet des informations tout en l'activant. Sur ces

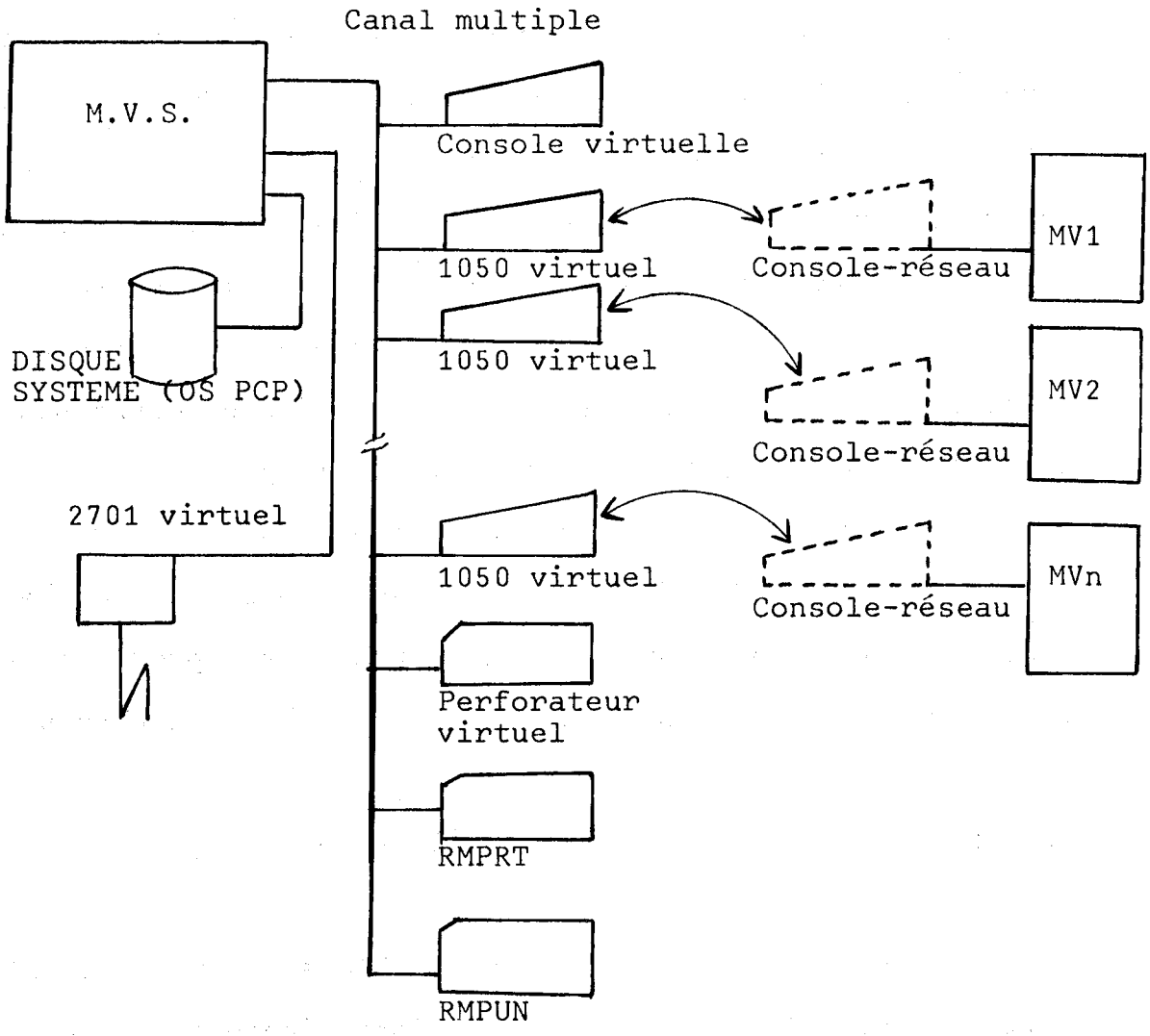



FIGURE IV.4. Configuration de la machine M.V.S. et correspondances entre ses unités et les consoles-réseau.

deux opérations sont basés les protocoles qui ont été décrits dans le chapitre III et qui réglementent les relations entre les unités d'un réseau virtuel et le point de service qui donne accès à CP.

Afin de permettre l'utilisation de CP dans un réseau d'ordinateurs, nous avons ajouté dans ses tables un certain nombre de MRDEBLOK. Ces tables représentent des unités appelées terminaux-réseau ou consoles-réseau. Pour les différencier des

unités existantes (télétypes, IBM 2741 etc...) elles reçoivent un nouveau type appelé 1052X. Les consoles-réseau n'ont aucune existence physique pour le système. Elle représentent en fait les terminaux éloignés. Si un utilisateur éloigné désire se connecter à CP, il devra au préalable se faire allouer un terminal-réseau. Les consoles-réseau de CP sont en correspondance biunivoque avec des unités virtuelles de MVS (FIGURE IV.4.). En associant un processus conversationnel à une unité virtuelle du type 1050 lors de la demande de connexion le terminal-réseau correspondant est automatiquement alloué à l'utilisateur éloigné.

Une console-réseau est considérée par CP comme un terminal parfait. Vis à vis de CP les entrées/sorties se passent sans erreur. Les erreurs qui peuvent se produire sur les terminaux éloignés sont traitées sur place donc ignorées par CP.

II.1.4. Entrées/sorties sur une console-réseau.

Dans ce paragraphe nous allons décrire le mécanisme utilisé pour effectuer les opérations d'entées/sorties sur une console-réseau. Il se base sur des simulations d'opérations qui sont normalement effectuées par le "HARDWARE". On peut considérer qu'une opération d'entrée/sortie se décompose en trois phases [I1] :

- Démarrage de l'opération (exécution de l'instruction SIO au niveau du calculateur),
- Analyse et traitement de l'opération par le canal,
- Fin de l'opération par interruption sur le calculateur.

Le même schéma peut être appliqué à une console réseau:

- Lancement de l'opération par simulation d'une instruction SIO,

- Analyse et traitement de l'opération,
- Fin de l'opération par simulation d'une interruption.

II.1.4.1. Lancement de l'opération.

Le lancement d'une opération sur une console-réseau est simulé par CP qui provoque une interruption sur l'unité virtuelle de MVS correspondant au terminal-réseau (FIGURE IV.4.). Des vérifications sont faites avant pour déterminer si l'unité existe et si elle est libre. Ces vérifications sont aisées car se faisant au niveau du système CP, les blocs décrivant toutes les unités virtuelles de MVS sont accessibles. Si l'unité sélectée est libre la requête qui doit être lancée est racrochée à son bloc (MVDEBLOK).

II.1.4.2. Analyse de l'opération.

L'analyse et le traitement de l'opération sont effectués une fois que le système de MVS a traité l'interruption provoquée par CP. Par une intruction DIAGNOSE la nature de l'opération ainsi que ses paramètres sont déterminés. Cela consiste à placer dans la mémoire de MVS le bloc décrivant la requête d'entrée/sortie. Celui-ci contient dans le cas d'une demande de lecture un code indiquant la nature de l'opération ainsi que l'adresse de la zone que CP a allouée pour recevoir les données lues. Dans le cas d'une écriture les informations transmises à MVS précisent les données à écrire. En ce qui concerne le traitement de l'opération, cela consiste à traduire la requête en commandes du protocole entre l'interface puits et l'interface source de l'unité virtuelle du type CONSL attachée au

processus conversationnel associé au terminal-réseau (cf CHAPITRE III). Les commandes sont ensuite envoyées à l'interface SOURCE du centre correspondant.

II.1.4.3. Fin de l'opération.

La fin de l'opération est signifiée à CP par la simulation d'une interruption. Une interruption sur un calculateur se manifeste de la façon suivante: Le vieux PSW d'entrée/sortie reçoit entre autre chose l'adresse de l'instruction qui allait être exécutée au moment de l'interruption ainsi que l'adresse de l'unité qui la provoque. Le PSW du calculateur quant à lui est chargé avec le double mot appelé nouveau PSW d'entrée/sortie et qui contient l'adresse du module qui traite les interruptions. Le CSW fournit à ce dernier l'état de l'unité ou les conditions dans lesquelles l'opération s'est terminée. Tout cela se déroule automatiquement par "HARDWARE" sans l'intervention d'aucun programme. Dans le cas qui nous interresse il s'agit de simuler toutes ces opérations. La simulation se fait à l'initiative de MVS à l'aide d'une instruction DIAGNOSE qui permet de transmettre à CP dans le cas d'une lecture, l'adresse de la zone fournie par CP lors de l'analyse de l'opération ainsi que les données lues. L'adresse de l'unité concernée et éventuellement une condition d'ATTENTION sont passées en même temps. Il ne reste alors plus qu'à mouvementer les données dans la zone prévue à cet effet quand l'opération est une lecture, positionner le CSW, le vieux PSW d'entrée/sortie avec l'adresse du terminal-réseau et l'adresse de l'instruction suivant l'instruction DIAGNOSE dans la mémoire de MVS. Le nouveau PSW d'entrée/sortie est ensuite substitué à celui

du calculateur.

Il est à remarquer que ce mécanisme est analogue à celui des entrées/sorties sur des terminaux réels. La machine MVS se comporte de la même manière qu'un canal qui emprunterait l'unité arithmétique et logique du calculateur pour effectuer ses calculs, positionner les mémoires et les analyser. Elle va chercher les ordres décrivant la requête d'entrée/sortie dans le descripteur de son unité virtuelle correspondant au terminal-réseau comme le canal va chercher l'adresse du programme canal dans le CAW à l'adresse fixe 72. Le rôle du canal est de décoder les commandes contenues dans le programme canal et de produire des ordres pour l'unité de contrôle. L'ensemble de ces commandes constitue le protocole entre le programme qui demande les opérations et le canal seul habilité à les effectuer. A partir des ordres reçus l'unité de contrôle envoie à son tour des ordres à l'unité. De manière similaire il existe un protocole entre MVS et CP concernant les codes qui sont dans le bloc représentant la requête. Ces ordres sont ensuite traduits en commandes du protocole de dialogue entre les interfaces d'une unité virtuelle (au sens où ce terme a été utilisé dans le chapitre III). MVS joue donc aussi le rôle d'unité de contrôle pour les terminaux-réseau. De la même manière que le canal est lancé sur les opérations de décodage par une instruction SIO, MVS est lancée par CP sur l'analyse de la requête. C'est aussi sur l'initiative de MVS que l'interruption est simulée comme c'est sur l'initiative du canal que l'unité centrale est interrompue.

Nous pouvons dire que par ces simulations nous réalisons des opérations d'entrées/sorties "SOFTWARE". En partant de cette analogie il apparait que ce mécanisme est applicable à tout système utilisant une machine qui dispose du dispositif

d'interruption et d'instructions du type DIAGNOSE. Celle-ci pouvant parfaitement être remplacée par une instruction appel au superviseur (SVC).

II.1.5. Récupération des fichiers.

En nous plaçant du point de vue de CP, nous pouvons distinguer deux sortes de fichiers: les fichiers en entrée et en sortie qui correspondent respectivement aux fichiers destinés aux lecteurs virtuels et à ceux produits par les imprimantes et les perforateurs virtuels. Rappelons brièvement le principe d'utilisation de ces organes d'entrée/sortie.

Les paquets de cartes que l'on destine à un lecteur virtuel sont préalablement lus par un lecteur réel de CP qui reconnait leur destination grâce à la première carte de chacun des paquets. Les images de cartes sont rangées sur disque dans une zone temporaire jusqu'à leur utilisation par une machine virtuelle.

Quant aux sorties produites par une imprimante ou un perforateur virtuel, CP utilise un mécanisme symétrique pour les traiter. Les sorties sont rangées sur disque avant d'être effectivement imprimées ou perforées.

Lorsque CP est utilisé dans l'environnement d'un réseau d'ordinateurs, il s'agit de réaliser les mêmes opérations, c'est à dire transfert de données d'un lecteur réel vers une zone temporaire ou de celle-ci à une imprimante ou un perforateur réels. La seule différence avec l'utilisation locale de CP est que les organes d'entrée/sortie sont situés dans d'autres centres. Nous allons voir les procédures qui permettent à un utilisateur éloigné de CP de récupérer ses sorties et de se

servir d'un fichier de cartes lu réellement dans un autre centre.

II.1.5.1. Fichiers en entrée.

L'acheminement d'un fichier destiné à un lecteur virtuel d'un centre éloigné au centre de CP s'effectue par l'intermédiaire de l'unité virtuelle de type RDR que l'utilisateur a créée grâce à la commande READER. Les données sont compressées avant qu'elles ne soient transmises ce qui permet de gagner sur le temps de transmission. Pour respecter les conventions de CP le fichier est précédé de l'identification de la machine virtuelle qui sera vérifiée avant que l'opération de transfert d'un centre à un autre ne soit commencée. Le processus conversationnel serveur qui gère l'interface PUITS de l'unité du type RDR demande au système de MVS de lui allouer un de ses perforateurs virtuels. Celle-ci exécute ensuite la commande:

XFER ccu TO user,

ccu est l'adresse du perforateur alloué et user est le nom de la machine virtuelle à laquelle est destiné le fichier. La commande XFER de CP permet de transformer les sorties de l'unité d'adresse ccu de la machine qui l'a émise en entrées du lecteur virtuel de la machine spécifiée en paramètre.

Une fois la commande XFER émise, les images de cartes sont perforées. Les cartes peuvent alors être recupérées par leur destinataire. Dès que le paquet est entièrement perforé, la commande: XFER ccu OFF annule le dispositif de transfert pour les sorties suivantes avant que l'unité ne soit libérée.

Il est à noter que ce mécanisme n'introduit aucune modification dans CP. C'est une procédure qui utilise les opérations d'entrées/sorties sur des unités virtuelles. A ce

propos on peut remarquer que ces opérations étant simulées par CP, elles représentent une charge pour le système. Cela pourrait être évité si le système fournissait des moyens d'accès en écriture à la gestion de ses fichiers de spooling. Les images de cartes seraient alors groupées par le processus conversationnel en blocs ayant le format attendu par les modules de cette gestion avant d'être transmises à CP qui effectuerait alors directement l'écriture dans la zone temporaire. Malheureusement seule la lecture de tels blocs est actuellement permise.

II.1.5.2. Fichiers en sortie.

Suivant qu'ils soient produits par des imprimantes ou des perforateurs virtuels, les fichiers en sortie sont transférés du centre de CP à leur destination au moyen d'une unité virtuelle de type PRT ou PUN. Pour faire parvenir les données au processus conversationnel qui gère l'interface SOURCE d'une telle unité nous avons deux possibilités: soit faire parvenir un élément qui correspond à un ordre d'entrée/sortie (un enregistrement), soit de transmettre un élément correspondant à un bloc qui comprend plusieurs enregistrements.

II.1.5.2.1. Traitement au niveau de l'enregistrement.

Cette solution est basée sur le même principe que celui des entrées/sorties sur les consoles-réseau. A une unité du type PRT ou PUN est associée soit une imprimante-réseau soit un perforateur-réseau. Les données sont alors transmises à MVS enregistrement par enregistrement. Les opérations étant exclusivement des écritures, une fois parvenues à MVS celle-ci

est chargée de simuler une interruption et d'envoyer les données à leur destination. Il est évident que MVS doit disposer d'un mécanisme de mémorisation de ces données qui soit capable de garder un grand nombre d'enregistrements. En effet chaque élément transmis à MVS correspond à une opération émise par une machine virtuelle. Ceci exclue d'adopter une méthode qui envoie directement ces éléments au centre éloigné. On serait alors limité par la vitesse de la ligne de transmission avec une répercussion sur la machine ayant émis les opérations. Cette solution est donc acceptable si le système de MVS fournit une gestion de fichiers. Cela permettrait en effet de garder les enregistrements de plusieurs sorties et de les transmettre quand la charge du système le permettra et surtout à la vitesse qu'il permettra.

II.1.5.2.2. Traitement au niveau du bloc.

La seconde solution utilise les facilités que la gestion de fichiers de CP met à la disposition des utilisateurs. Il est possible d'obtenir un bloc (829 octets) d'un fichier créé par CP et ceci par l'intermédiaire d'une unité virtuelle d'un type particulier (RMPUN et RMPRT). La procédure de récupération des fichiers en sortie est alors la suivante:

Dès qu'un utilisateur éloigné déclare une unité virtuelle du type PRT ou PUN, le processus conversationnel crée une interface SOURCE et demande l'allocation d'une unité RPRT ou RPUN. Quand la machine virtuelle de cet utilisateur demande une impression ou une perforation, CP suit la procédure normale jusqu'à ce que les opérations correspondantes soient terminées, c'est à dire jusqu'à la fermeture du fichier de spooling. La

procédure normale met ensuite le descripteur de ce fichier dans une liste qui est consultée à chaque fois qu'une des unités réelles correspondante de CP est disponible. Le fichier est alors imprimé ou perforé suivant le cas. Dans la situation d'un utilisateur éloigné la procédure est modifiée. Le descripteur du fichier, après la fermeture de celui-ci est inséré dans une autre liste qui permet aux unités RMPRT et RMPUN de le récupérer bloc par bloc. Il ne reste plus alors qu'à transformer le contenu de chaque bloc en autant de commandes du protocole de dialogue entre les interfaces SOURCE et PUITS correspondantes qu'il est nécessaire. Cette procédure a l'avantage d'utiliser au maximum les possibilités déjà existantes. La seule modification apportée est d'insérer à la fermeture du fichier un mécanisme qui permet de le rattacher à la liste d'où il est possible de l'obtenir plus tard. La difficulté à surmonter à ce niveau vient du fait que les unités RMPRT et RMPUN étant acquises dynamiquement, il n'est pas possible de savoir à ce moment à quelle adresse il faut rattacher le fichier. Celui-ci reçoit alors une identification temporaire. CP par un mécanisme d'interruption, indique au processus conversationnel qu'il y a un fichier en sortie, son type (PRT ou PUN) ainsi que son identification temporaire. Le processus répond ensuite par une instruction DIAGNOSE qui transmet à CP l'adresse et l'identification de l'unité définie par une commande PRINTER ou PUNCHER. Si une telle unité n'existe pas, CP en est averti et la procédure normale se poursuit; ce qui revient à dire que le centre de CP est la destination par défaut de tous les fichiers.

La seconde solution étant plus facile à mettre en oeuvre a été retenue.

# III. CONCENTRATEUR DE TERMINAUX ET STRUCTURE INTERNE DU PROCESSUS CONVERSATIONNEL.

## III.1. Concentrateur de terminaux.

Le concentrateur de terminaux est la partie du système-réseau chargée de gérer les terminaux. Il est essentiellement constitué de modules qui effectuent les entrées/sorties sur ces périphériques. C'est aussi la liaison entre l'interpréteur de commandes du processus conversationnel et l'utilisateur.

## III. Les terminaux attachés au système-réseau.

Nous avons vu que le système-réseau était une tâche du système O.S. A l'initialisation de cette tâche les terminaux actifs, leurs adresses et leurs types sont précisés. Ceci permet déjà de mémoriser les ressources disponibles pour le système-réseau et de construire les tables nécessaires pour la gestion de ces terminaux. Les adresses des routines de services spécifiques dont le rôle est de lancer les opérations, traiter les erreurs, traiter les fins d'opérations et mettre en attente les requêtes d'entrées/sorties sont associées à chaque terminal suivant son type. Les procédures d'entrées sorties sont utilisées au niveau basique [OS1,OS3] . C'est à dire que les programmes canaux sont construits par des programmes particuliers sans passer par des méthodes d'accès. Les opérations sont soumises au superviseur d'entrées/sorties au moyen de la macro-instruction EXCP de l'O.S. [OS2] . Les routines qui prennent le contrôle sur fin d'entrée/sortie ont été remaniées. Le choix d'une telle

méthode est justifié par le besoin d'un contrôle très poussé des opérations d'entrées/sorties. En effet il est nécessaire de détecter par exemple une ATTENTION et de faire intéragir directement le processus conversationnel avec l'unité qui lui est attachée à un instant donné. C'est le cas par exemple lorsqu'une opération se termine il faut alors placer les indications nécessaires dans les tables du processus.

## III.2. Structure interne du processus conversationnel.

### III.2.1. Composants du processus conversationnel.

Le processus conversationnel a été défini plus haut comme un ensemble de fonctions qui permettent d'utiliser un système intéractif éloigné. L'acheminement des informations qui constituent l'intéraction entre l'utilisateur et le système est assuré sans préjuger du contenu de ces informations et des programmes qui les traitent. Suivant les activités de ces programmes, les informations acheminées sont destinées à diverses unités; ce qui nécessite l'utilisation d'unités virtuelles. La réception et le traitement des données pour les différentes unités n'arrivant pas séquentiellement, il faut prévoir un multitraitement et un mélange des activités associées aux unités. De façon globale nous pouvons distinguer les composants suivants :

.Module d'interprétation des commandes: il est en relation par l'intermédiaire du concentrateur de terminaux avec la console de l'utilisateur.

.Le module de communication avec les processus éloignés chargé

de multiplexer les messages et les démultiplexer. Il communique d'une part avec l'interface de communication (à l'aide des commandes RCV et SEND) et d'autre part avec le module de traduction d'ordres.

.Les modules de traduction d'ordres reçoivent les requêtes préparées par le module précédent, préparent les opérations d'entrées/sorties et sollicitent le superviseur d'entrées sorties.

.Le superviseur d'activités. Le module de communication, les M.T.O. et les tables de description des unités virtuelles constituent les programmes et les données des interfaces d'unités virtuelles. A chacun de ces programmes correspond une activité. Pour assurer le multitaitement de ces activités un superviseur interne est chargé de sélectionner l'activité qui doit prendre le contrôle.

III.2.2. Représentation des interfaces.

Chaque interface correspondant à une unité virtuelle créée par une commande (CONNECT, PRINTER, READER,...) est représentée par un bloc (REPBLOC FIGURE IV.5) qui contient tous les éléments de l'interface. Tous les blocs d'un même réseau virtuel sont chainés entre eux (FIGURE IV.7.) et rattachés à une racine : RVBLOC. Tous les blocs ont la même structure: une en-tête dont la structure est la même pour tous les types d'unités et une extension qui varie selon le type. L'en-tête contient :

.un pointeur pour le chainage (REPNEXT),

.le type de l'unité (TYP),

.l'identificateur du centre source (CS),

.le numéro du processus source (PS),

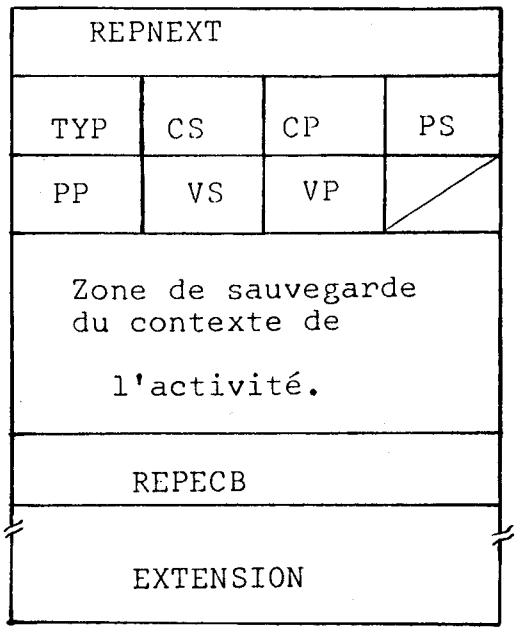



FIGURE IV.5. Structure du REPBLOC.

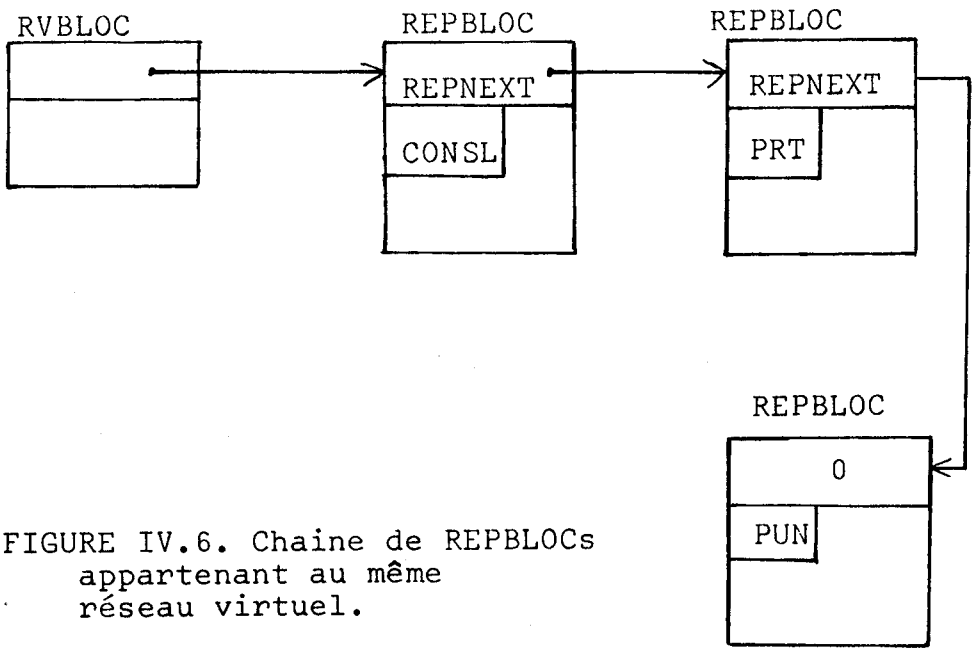



FIGURE IV.6. Chaine de REPBLOCs appartenant au même réseau virtuel.

.l'identificateur du centre puits (CP),

.le numéro du processus puits (PP),

.le numéro de voie logique de communication avec le processus source (VS),

.le numéro de voie logique de communication avec le processus puits (VP),

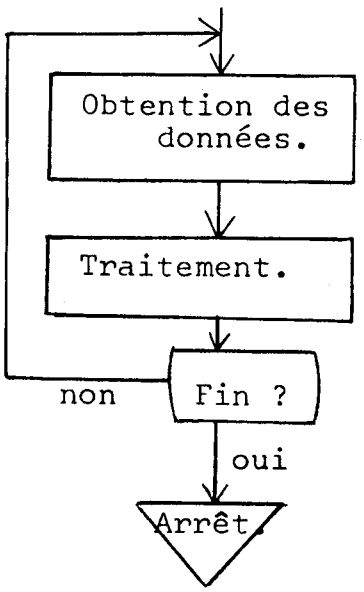
.la zone de sauvegarde des registres et du contexte de l'activité associée,

.l'indicateur d'événement associé à l'interface (REPECB).

La zone de sauvegarde ainsi que l'indicateur d'événement sont utilisés par le superviseur d'activité.

III.2.3. Multitraitement des activités.

Toutes les activités du processus conversationnel ont une caractéristique commune. Elles sont cycliques:




L'obtention des données se fait à partir d'unités ou de l'interface de communication. Ces deux modes nécessitent une mise en attente de l'activité jusqu'à ce que les données arrivent. La

mise en attente d'un processus est réalisée en faisant appel à la fonction WAIT du module de gestion des processus. C'est à ce niveau que le problème du multitraitement se pose. Si la possibilité d'exécuter un appel "Call WAIT" est laissé à une activité tout le processus conversationnel reste bloqué jusqu'à l'arrivée de l'événement attendu. Par exemple le M.T.0 d'un terminal après avoir construit le programme canal approprié appelle le superviseur d'entrées/sorties et se met en attente de la manière suivante:

```
        Call IOS(IOB)
    PA: Call WAIT(ECB,MSK)
```

Le processus reste alors en attente jusqu'à la fin de l'opération. A supposer qu'une activité associée à une unité du type RDR soit activable, elle ne pourra prendre le contrôle que quand le M.T.0 le rendra au superviseur d'activité. Pour éviter ce genre de problème et pour profiter au maximum de l'attente des activités le schéma d'exécution est imposé à toute activité. Seul le superviseur d'activité est habilité à appeler le module de gestion des processus pour mettre le processus en attente. De ce fait toute activité qui à un instant donné doit attendre un événement donne le contrôle au superviseur d'activité par l'appel suivant:

```
        Call SUPACT(REPBLOC,MSK).
```

Ce dernier tient à jour une liste de pointeurs sur les indicateurs d'événement des activités qui ont fait appel à lui. Cette liste est utilisée d'une part comme paramètre à passer au module de gestion des processus et d'autre part dans la recherche de l'activité à laquelle donner le contrôle quand le processus est réactivé après une attente. Le superviseur d'activités reçoit le contrôle soit d'une activité qui est en attente soit du module

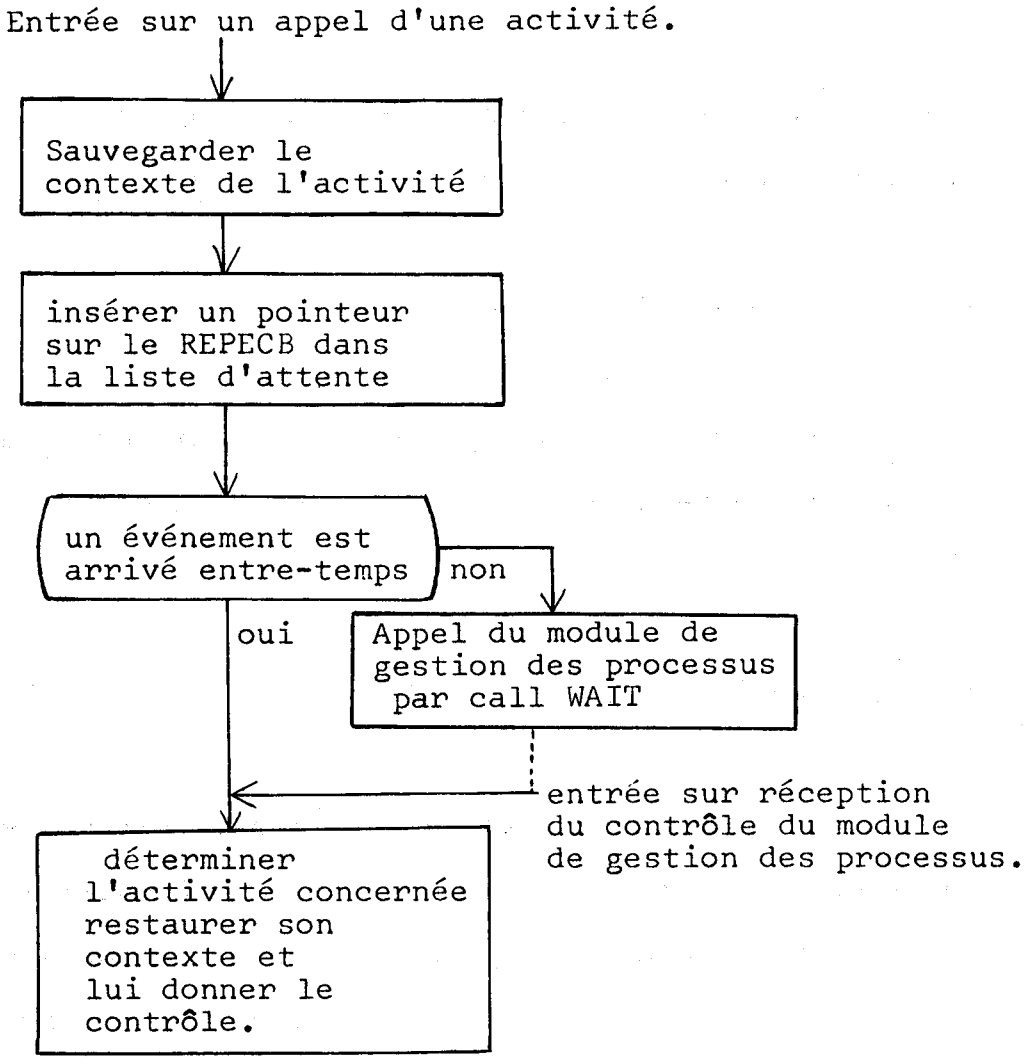



FIGURE IV.7. Schéma du superviseur d'activités.

de gestion des processus du système-réseau. L'organigramme de la figure IV.7 donne une idée de la manière dont agit le superviseur d'activités dans ces deux cas. Lorsque ce dernier reçoit le contrôle du module de gestion des processus il existe au moins une activité prête à reprendre le contrôle. Quand il y en a plus d'une un problème de choix et de priorité se pose. Etant dans un contexte conversationnel, il est souhaitable de donner le contrôle à l'activité rattachée au terminal en priorité. La recherche de l'activité concernée par l'événement arrivé se

faisant en parcourant de manière séquentielle la liste des pointeurs sur les indicateurs d'événement, la priorité d'une activité est détérminée par la position du pointeur associé dans la liste. Ainsi en plaçant en tête le pointeur de l'activité du terminal, celle-ci est assurée de la plus grande priorité.

III.2.4. Exemples d'activités.

Nous allons présenter deux exemples d'activités associées respectivement aux unités virtuelles du type CONSL et RDR.

III.2.4.1. Activité du type CONSL.

L'activité attachée à l'unité virtuelle du type CONSL peut se diviser en deux parties complémentaires: le module de communication avec les processus correspondants et le M.T.O. La première partie est en communication avec l'interface complémentaire. Elle traite donc le bloc de commandes qui arrive pour l'unité virtuelle. Cela consiste à démultiplexer les commandes et à rattacher les requêtes ainsi obtenues au REPBLOC associé. Les commandes arrivent groupées au maximum par cinq unités. Ce nombre est fixé arbitrairement dans le prototype réalisé. Il pourrait faire l'objet d'une détermination par mesure ou simulation de façon a gagner le maximum du temps nécessaire au traitement d'un bloc de transmission par l'interface de communication. Quand le bloc est entièrement traité le contrôle est rendu au superviseur d'activités après l'exécution d'un "RCV" :

```
Call RCV(VP,BUFAD,REPECB,x'80');
```

```
Call SUPACT(REPBLOC,x'80');
```

Le module de traduction d'ordres prend le contrôle pour traiter les requêtes préparées dans la première partie. Ceci se traduit par des opérations sur le terminal attaché à l'unité virtuelle. De la même manière le contrôle est rendu au superviseur d'activités après le lancement de chaque opération; ce qui permet aux autres activités de prendre le contrôle et notamment de traiter un autre bloc de commandes pendant que l'utilisateur rentre une ligne au terminal.

Dans le centre correspondant, le même schéma s'applique. Le M.T.O. est chargé d'obtenir des ordres du protocole de dialogue avec CP par l'intermédiaire de la console-réseau (cf. chapitre III). Ces ordres sont traduits en commandes du protocole entre les interfaces de l'unité virtuelle CONSL qui sont insérées dans un bloc. Ce dernier n'est envoyé au centre correspondant que si la dernière commande est la cinquième ou si c'est une lecture. Ceci permet d'éviter des situations de blocage.

III.2.4.2. Activité du type RDR.

Nous allons décrire brièvement le principe de l'activité attachée à une unité du type RDR. Il faut préciser que l'implémentation se sert des fichiers de spooling de VM370 [VM1] . Ceci est possible quand le système-réseau s'exécute sur une machine virtuelle activée par l'hyperviseur VM370. Dans le cas ou une machine réelle est utilisée la gestion de fichiers de GMS/40 [G2] a été adaptée pour être utilisée sous le système OS/360.

L'activité du type RDR est initialisée dans les deux

centres concernés par la commande READER qui crée les interfaces nécessaires. Mais elle ne prend le contrôle que lorsque l'utilisateur lance la commande ATTACH. Prenons l'exemple où celui-ci a lancé la commande :

ATTACH READER TO RDR 27.

Le MTO attaché à l'unité RDR obtient de la gestion de fichiers de spooling de VM370 les premiers enregistrements du fichier RDR 27. Le premier enregistrement doit contenir l'identification de la machine virtuelle destinataire. l'identification est envoyée au système conversationnel éloigné pour vérifier sa validité. Dans le cas où le système éloigné refuse l'identification, l'utilisateur est invité à respécifier le destinataire (commande USERID). Les enregistrements sont ensuite transmis au module de communication avec le processus correspondant qui les envoie par groupes de dix enregistrements.

Du côté du processus correspondant, dès que l'identification du destinataire est reconnue valide, la commande XFER est exécutée (voir §II.1.5.1.). Les groupes d'enregistrements sont ensuite perforés au fur et à mesure de leur arrivée. Lorsque tout le fichier est reçu et perforé il se retrouve automatiquement attaché au lecteur de la machine virtuelle destinataire.

# C O N C L U S I O N

Comme on peut le constater à travers la présentation qui vient d'être faite, le travail que nous avons effectué comportait des aspects aussi bien pratiques que théoriques.

La réalisation pratique fut profitable à deux titres. Elle nous a donné l'occasion d'approfondir nos connaissances sur un système particulier : CP/CMS, approche indispensable pour introduire des modifications concises et efficaces. Elle nous a aussi permis d'acquérir une expérience dans un domaine qui s'étend de plus en plus; celui des réseaux d'ordinateurs. Cette expérience est d'autant plus riche et réelle que nous étions confrontés à des problèmes concrêts. Si la partie qui consistait à réaliser le processus conversationnel constituait un prototype et de ce fait permettait une certaine liberté, il en était autrement des modifications apportées au système CP/CMS. En effet, en s'insérant dans un système en exploitation il fallait donner un produit fiable et opérationnel d'autant plus qu'un autre projet de réseau d'ordinateurs (CYCLADES) allait être amené à utiliser la méthode que nous avons développée

pour SOC. Toutefois même dans le processus conversationnel, nous nous sommes efforcés d'adopter des solutions aussi générales que possible. Ainsi le mécanisme de communication entre l'hyperviseur CP et le processus conversationnel est assez indépendant des fonctions du système-réseau de SOC pour qu'il puisse être utilisé par d'autres projets. Les notions introduites dans le chapitre III et appliquées dans le chapitre suivant ne sont pas spécifiques à une utilisation conversationnelle et peuvent être généralisées comme nous le verrons plus loin.

La partie théorique, quant à elle, a porté sur des mécanismes élémentaires tels que création de processus correspondant, établissement de liaisons et réalisation d'opérations d'entrées/sorties. Dans cette étude nous nous sommes délibérément placés au niveau du système. Il a donc fallu développer des outils nouveaux qui cependant sont situés trop bas pour être mis à la disposition de l'utilisateur. En effet celui-ci est habitué à être indépendant des unités et de la logique interne des systèmes auxquels il soumet ses travaux. Cela nous amène donc à fournir aux utilisateurs des outils plus élaborés qui automatiseraient la création des processus nécessaires, leurs synchronisations suivant l'application que le système-réseau doit traiter. Ce dernier se charge de l'allocation des ressources et de résoudre les problèmes qu'elle pose [S5] . Les possibilités d'un réseau d'ordinateurs constitueraient alors un nombre de fonctions disponibles aussi bien au niveau d'un langage de commande analogue au langage externe du projet SOC qu'au niveau des programmes particuliers. Nous abordons dès lors le problème du domaine d'utilisation des fonctions d'un réseau d'ordinateurs : faut-il distinguer les utilisateurs d'un système faisant partie d'un réseau suivant le critère d'utilisation

des possibilités éloignées? Si la réponse est oui cela veut dire que l'on peut se contenter d'une organisation du type de celle du réseau expérimental SOC dans laquelle les "travaux-réseau" sont soumis par une entrée spéciale: ce sont des données pour un travail particulier. Ceci a pour conséquence immédiate que tout programme ayant été conçu pour une exécution avec des ressources locales doit être adapté lorsqu'il est soumis dans un environnement de réseau d'ordinateurs. L'effort d'adaptation demandé à l'utilisateur peut être plus ou moins grand suivant l'application et peut décourager les programmeurs si les modifications sont trop importantes.

Dans l'étude et la présentation du réseau SOC faite dans le premier chapitre il ressort que seuls les processus du système-réseau utilisent réellement les procédures dévelopées pour communiquer et pour coopérer avec les programmes éloignés qui sont aussi des processus-système. Ceci constitue une limitation très stricte car quelle que soit la généralité de ces processus-sytème ils ne satisferont jamais tous les besoins des utilisateurs. C'est pour cela qu'il faut envisager une autre conception de la soumission des travaux dans un réseau d'ordinateurs. Les travaux locaux et les travaux qui requièrent les services de centres éloignés doivent être pris en charge par le même système. Ceci implique que toutes les fonctions propres au réseau doivent être placées au niveau du système d'exploitation. Du point de vue externe il n'y aurait donc pas de différence entre tous les travaux. Les programmes utilisant plus d'un centre pourraient le faire de deux manières :

En se servant de fonctions élémentaires ou en se servant de programmes utilitaires ou méthodes d'accès. Cette distinction est analogue à celle que l'on fait en considérant les deux manières d'utiliser les unités d'entrées/sorties :

- programmation basique avec construction des programmes canaux et appel au superviseur d'entrées/sorties.

- programmation à l'aide de méthodes d'accès qui se servent elles mêmes de la programmation basique.

Les fonctions élémentaires dont auraient alors besoin les utilisateurs pourraient comprendre :

- obtention de processus correspondant,
- construction des interfaces nécessaires avec tous les appels aux fonctions du système-réseau (LNKGET, RFL,LNKCLS, etc..),
- primitives de l'interface de communication.

Ces fonctions deviennent accessibles aux programmes particuliers. Cela implique la construction d'un nouvel ensemble de macro-instructions ou l'extension des macro-instructions de la gestion de données (data-management) des systèmes actuels en vue de traiter de nouveaux paramètres relatifs aux localisations et aux caractéristiques des centres du réseau. Des programmes déjà exécutés avec succès sur des données locales pourraient utiliser des données éloignées sans que l'on éprouve pour autant le besoin de les transformer. Les indications concernant les situations géographiques des données se feraient alors à l'aide d'ordres de contrôle (cartes DD ou ordres en langage de contrôle), ce qui se fait déjà pour faire l'association entre les ressources physiques et les ressources logiques du programme dans la plupart des systèmes. Lors de l'exécution l'activation de méthodes d'accès appropriées serait faite pour traiter les données dans ce nouveau contexte. Tout le travail de construction des interfaces nécessaires, des établissement des liaisons, de création de processus correspondant incomberait à ces méthodes d'accès. Le réseau peut être considéré

comme une extension de ressources et l'utilisateur n'a donc plus à se soucier des intermédiaires qu'il faut mettre en place pour faire communiquer deux de ses programmes ou pour obtenir des données éloignées. Tous ces mécanismes doivent rester au niveau du système d'exploitation qui mettrait à la disposition des utilisateurs des fonctions ou des ressources nouvelles telles que :

- Etablissement de liaison avec un programme éloigné et la méthode d'accès associée,
- Soumission dynamique (au moment de l'exécution) d'un programme à un système éloigné, avec toutes les conséquences que cela implique : identification du programme soumis ainsi que son adressage dans l'ensemble du réseau,
- Méthodes d'accès à des fichiers éloignés,
- Primitives de synchronisation entre des programmes,
- Accès sélectifs à des données à l'aide de programmes particuliers ou de méthodes d'accès.

Toutes ces notions ouvrent un champ nouveau de recherches et d'applications qui aborde le domaine de l'exécution distribuée de programmes et du partage de charge entre les machines d'un réseau d'ordinateurs. On peut avoir une idée des applications possibles dans [S6] .

# ANNEXE

# EXEMPLES DE SESSION

```
SOC ON LINE
ENTER ID:zhiri
connect imag
CONNEXION SUCCEEDED 05
reader sou=cs sink=imag
status
05/IMAG     CONNECTED
READER :CS     /IMAG     00 RECORD(S)
CONSOLE:CS     /IMAG     00 BUFFER(S)

discon
05/IMAG    TEMPORARILY DISCONNECTED
connect imag
CONNEXION SUCCEEDED 06
status
06/IMAG     CONNECTED
CONSOLE:CS     /IMAG     00 BUFFER(S)

05/IMAG     NOT CONNECTED
READER :CS     /IMAG     00 RECORD(S)
CONSOLE:CS     /IMAG     00 BUFFER(S)

reader sink=imag source=cs
stat
06/IMAG     CONNECTED
READER :CS     /IMAG     00 RECORD(S)
CONSOLE:CS     /IMAG     00 BUFFER(S)

05/IMAG     NOT CONNECTED
READER :CS     /IMAG     00 RECORD(S)
CONSOLE:CS     /IMAG     00 BUFFER(S)

listf
RDR       0249 00024
RDR       0250 00025
RDR       0251 00025
RDR       0252 00025
attach reader to rdr 249

LOCAL MODE
06 READER ENDED 023 CARDS READ
listf
RDR       0250 00025
RDR       0251 00025
RDR       0252 00025
discon
06/IMAG    TEMPORARILY DISCONNECTED
connect imag 05
IMAG    05 RECONNECTED
attach reader to rdr 250
LOCAL MODE
05 USERID INVALID REPLY OR CANCEL
```

```
userid svm
LOCAL MODE
05 READER ENDED 025 CARDS READ
attach reader to rdr 25
FILE NOT FOUND
attach rdr to ¢attach reader to rdr 251

LOCAL MODE
05 USERID INVALID REPLY OR CANCEL
cancel

LOCAL MODE
05 READER ENDED  NO CARD  READ
listf
RDR      0252 00025
erase rdr 252
listf
FILE NOT FOUND
free buffer
INVALID PARAMETER LIST
free reader
00  BUFFERS PURGED
READER  FREE
status
05/IMAG    CONNECTED
CONSOLE:CS    /IMAG    00 BUFFER(S)

06/IMAG    NOT CONNECTED
READER :CS    /IMAG    00 RECORD(S)
CONSOLE:CS    /IMAG    00 BUFFER(S)

discon
05/IMAG   TEMPORARILY DISCONNECTED
connect imag 06
IMAG   06 RECONNECTED
printer source=imag sink=cs
status
06/IMAG    CONNECTED
PRINTER:IMAG  /CS      00 RECORD(S)
CONSOLE:CS    /IMAG    00 BUFFER(S)
READER :CS    /IMAG    00 RECORD(S)

05/IMAG    NOT CONNECTED
CONSOLE:CS    /IMAG    00 BUFFER(S)
```

```
vm/370 online    1jh359 qsyosu

d svm
DIALED TO SVM      030
SOC ON LINE
ENTER ID:zhiri
connect imag
CONNEXION SUCCEEDED 06
req
REQUEST MODE
CP-67 a votre service
_

1 zhiri
ENTER PASSWORD:
********
 DEV  194 IN USE BY MVS     ; NOT ATTACHED
DEV  198 IN USE BY MVS     ; NOT ATTACHED
   'REAL' TIMER
 READY AT 07:40:58 ON WEDNESDAY 11/28/73
 CP
i cms
 CMS..VERSION 3.0 decembre 72

edit ecrire sysin
NEW FILE.
INPUT:
ecrire>csect
>balr>12,0#>using *,12#>cmstype 'abcdefghijklmnopqrstuvwxyz'
>sr>15,15
>br>14#
EDIT:
top#p 12
ECRIRE   CSECT
         BALR  12,0
         USING *,12
         CMSTYPE 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
         SR    15,15
         BR    14
EOF:
file#global maclib syslib osmacro
R; T=0.13/0.39 07:43:53

R; T=0.01/0.01 07:43:53

asmg ecrire
* 00003A 07FE                                13          BR    14
  STMT  ERROR CODE  MESSAGE
    13   ASMG1011   EOD ON SYSIN.
     1 STATEMENT  FLAGGED IN THIS ASSEMBLY
E(00012); T=0.62/1.28 07:44:17

edit ecrire sysin#b#i >end
EDIT:
         BR    14
file
R; T=0.06/0.13 07:45:05
```

```
asmg ecrire
R; T=0.61/1.25 07:45:28

load ecrire (type
 ECRIRE   AT 12000
R; T=0.02/0.05 07:45:42

genmod ecrire
R; T=0.01/0.03 07:45:55

ecrire
ABCDEFGHIJKLMNOPQRSTUVWXYZ
R; T=0.01/0.01 07:46:03


_
 CP
q u
09 USERS, 00 DIALED

q v
 CORE - 00256K
 009 CONSL 02A>
LOCAL MODE
stat
06/IMAG    CONNECTED
CONSOLE:CS    /IMAG    14 BUFFER(S)

disc
06/IMAG   TEMPORARILY DISCONNECTED
connect imag
CONNEXION SUCCEEDED 07
req
REQUEST MODE
CP-67 a votre service
_

l cpsoc
ENTER PASSWORD:
[redacted]S
 DEV  192 IN USE BY MVS     ; SET TO R/O
DEV  193 IN USE BY MVS     ; NOT ATTACHED
DEV  194 IN USE BY ZHIRI   ; NOT ATTACHED
DEV  195 IN USE BY ZHIRI   ; NOT ATTACHED
DEV  196 IN USE BY ZHIRI   ; NOT ATTACHED
DEV  197 IN USE BY MVS     ; NOT ATTACHED
   'REAL' TIMER
 READY AT 07:47:53 ON WEDNESDAY 11/28/73
 CP
q u
10 USERS, 00 DIALED

q n
  ZHIRI     - 02A, LETI1     - 085, MESURE    DSCON, PLANNING DSCON
  BATCH     DSCON, OPERATOR - 01F, TEG01     - 083, PROFIZI  - 032
  MVS       - 070, CPSOC     - 02B
```

```
stat
?CP: STAT


>
LOCAL MODE
stat
07/IMAG    CONNECTED
CONSOLE:CS    /IMAG    00 BUFFER(S)

06/IMAG    NOT CONNECTED
CONSOLE:CS    /IMAG    14 BUFFER(S)

escape '
disc
07/IMAG   TEMPORARILY DISCONNECTED
connect imag 06
IMAG   06 RECONNECTED
req
 REQUEST MODE
 009 CONSL 02A
 00C SPOOL RDR
 00D SPOOL PUN
 00E SPOOL PRT
 0FF RPQ TIMER
 190 2314- 230 054 CYL R/O CP0000
 191 2314- 334 005 CYL R/W CPDS10
 193 2314- 290 002 CYL R/O CPDSK1
 195 2314- 293 025 CYL R/W VCPSOC
 196 2314- 293 025 CYL R/W VCPSOC
 197 2314- 293 025 CYL R/W VCPSOC
 19F FSPACE 0090 PAGES R/O
 380 2314- 290 003 CYL R/O CPDSK1

logout
CONNECT= 00:10:25  VIRTCPU= 000:01.51  TOTCPU= 000:03.47
LOGOUT AT 07:51:23 ON 11/28/73
CP-67 a votre service
'
LOCAL MODE
disc
06/IMAG   TEMPORARILY DISCONNECTED
connect imag 07
IMAG   07 RECONNECTED
req
 REQUEST MODE
q n
  LET11    - 085, MESURE   DSCON, PLANNING DSCON, BATCH    DSCON
  OPERATOR - 01F, TEG01    - 083, PROFIZI  - 032, MVS      - 070
  CPSOC    - 02B

logout
CONNECT= 00:04:49  VIRTCPU= 000:00.00  TOTCPU= 000:00.09
LOGOUT AT 07:52:41 ON 11/28/73
CP-67 a votre service
```

# REFERENCES BIBLIOGRAPHIQUES.

ARPA :

A1 : The terminal IMP for the ARPA network.
S.M. ORNSTEIN, F.E. HEART, W.R. CROWTHER, H.K. RISING, S.B. RUSSEL, A. MICHEL.
AFIPS Conference Proceedings Spring Joint Computer Conference 1972.

A2 : Computer network development to achieve ressource sharing.
L.G. ROBERTS, B.D. WESSLER.
AFIPS Conference Proceedings Spring Joint Computer Conference 1970.

A3 : Computer communication network design.
H. FRANK, R.E. KAHN, L. KLEINROCK.
AFIPS Conference Proceedings Spring Joint Computer Conference 1972.

A4 : Optimal design of computer networks.
H. FRANK.
Computer Networks.
Courant Computer Science Symposium 3
edité par Randall RUSTIN.

A5 : The Interface Message Processor for the ARPA network.
AFIPS Conference Proceedings Spring Joint Computer Conference 1970.

A6 : Analytic and Simulation methods in computer network design.
L. KLEINROCK.
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

A7 : A system for interprocess communication in a ressource sharing computer network.
D.C. WALDEN.
Communication of the ACM Avril 1972.

A8 : HOST to HOST protocols.
D.C. WALDEN.
INFOTECH Education Limited. Lecture manual.
20-22 Feb. 1973.

CYCLADES:

C1 : Présentation du réseau CYCLADES.
L. POUZIN.
Bibliothèque CYCLADES GAL 006.

C2 : Organisation et directives de MITRANET
J.L. GRANGE.
Bibliothèque CYCLADES MIT029.

CP/CMS:

CP1 : CP-67/CMS user's guide.
IBM GH20-0859-1

CP2 : CP-67/CMS Program Logic Manual.
IBM GY20-0590-1

CP3 : Systèmes CP-67 et CMS
A. AUROUX, C. HANS.
Publication de l'IMAG Juillet 1968.

VM1 : IBM Virtual Machine Facility/370 Control Program.
Program Logic.
IBM SY20-0880-0

DIVERS:

B1 : Mécanismes de base dans les systèmes hyperviseurs
Conception et réalisation d'un système à accès multiple.
J. BELLINO.
Thèse de Sciences Appliquées Sep. 1973
Université Scientifique et Médicale Grenoble.

D1 : Cooperating sequential process in
programming languages
E.W. DIJKSTRA.
NATO Advanced Study Institute.
F. GENUYS editor, Academic Press.

G1 : Prototype de réseau interactif (projet CRIC)
Ph. GUILLIER.
Workshop IRIA/ACM 23-24 Mars 1972.

G2 : Gestion de fichiers du système GMS/40.
J.P. FAURE, P. SERVANT.
INP GRENOBLE, Projet de fin d'étude. Juin 1971.

G3 : Grenoble Scientific Language, Définition du langage.
M. BERTHAUD, D. CLAUZEL, M. JACOLIN.
Etude Nº FF2-0133.
Développement scientifique. IBM France.

H1 : Evaluation of an interactive batch system network.
W.S. HOBGOOD.
IBM System Journal. 1972-1

H2 : Prevention of system deadlocks.
A.N. HABERMAN
CACM 12,7 1969.

H3 : An approach to multiprogramming.
P.B. HANSEN.
A paraître dans ACM computer surveys.

I1 : System 360 Principles of opérations.
IBM A22-6821

LH1 : Généralisation de la notion d'espace virtuel
sous les systèmes CP-67/CMS.
J.P. LEHEIGET
Thèse C.N.A.M. Juillet 1972.

M1 : Allocation de ressources dans un réseau d'ordinateurs.
R. MAHL, S. KAMRAN.
Workshop IRIA:ACM 23-24 Mars 1972.

R1 : Liaison HASP to HASP.
J. REICHLE.
Workshop IRIA/ACM 23-24 Mars 1972.

R2 : An interactive network of time-sharing computers.
R.M. RUTLEDGE. et autres
Proceedings of the 24th. National Conference
ACM, San Francisco, California. August 26-28 1969.

O1 : The MULTICS System: an examination of its structure.
E. ORGANICK.
MIT Press.

OS :

OS1 : OS/360 IOS Program Logic Manual.
IBM GY28-6617-7.

OS2 : IBM System 360/OS.
System control blocks.
IBM GC28-6628-7.

OS3 : IBM system programmer's guide.
IBM GC28-6050-8

OS4 : IBM system 360/OS
Time-Sharing Option, Planning for TSO.
IBM GC28-6698-1.

SOC :

S1 : SOC project : an experimental computer network.
S. GIRARDI.
Proceedings of the International Computer Symposium (ACM) Venice, Italie Avril 1972.

S2 : Etude d'un langage de commandes pour un système d'ordinateurs connectés.
R. DECALUWE.
Rapport de D.E.A. Université scientifique et médicale de GRENOBLE. 1970.

S3 : Etude et réalisation d'un langage de commande pour un réseau d'ordinateurs.
R. DECALUWE. Thèse de troisième cycle
Université scientifique et médicale de GRENOBLE 1973.

S4 : L'interface de communication d'un réseau d'ordinateurs.
S. GIRARDI. Workshop IRIA 23-24 Mars 1972.

S5 : Synchronisation et allocation de ressources dans un réseau d'ordinateurs.
M. SOMIA.
International Computing Symposium 1973. DAVOS.

S6 : Accessing remote resources in a computer network.
S. GIRARDI et A. ZHIRI.
IFIP 1974 (à paraitre).

VU

Grenoble, le 13 Novembre 1973

le Président de la thèse

VU, et permis d'imprimer,

Grenoble, le

le Président de l'Université
Scientifique et Médicale