Protection logicielle contre les erreurs dans un réseau d'ordinateurs hétérogène - Application à l' IBM 360/67 du réseau CYCLADES

Vincent Quint

THESE

présentée à

Université Scientifique et Médicale de Grenoble

Institut National Polytechnique de Grenoble

pour obtenir le grade de
DOCTEUR INGENIEUR

par

Vincent QUINT

PROTECTION LOGICIELLE CONTRE LES ERREURS DANS UN RESEAU D'ORDINATEURS HETEROGENE APPLICATION AU 360/67 DU RESEAU CYCLADES

Thèse soutenue le 20 décembre 1976 devant la Commission d'Examen

Président : N. GASTINEL

Examinateurs : L. BOLLIET
F. ANCEAU

Rapporteur extérieur : H. ZIMMERMANN

UNIVERSITE SCIENTIFIQUE
ET MEDICALE DE GRENOBLE

Monsieur Gabriel CAU : Président
Monsieur Pierre JULLIEN : Vice-Président

MEMBRES DU CORPS ENSEIGNANT DE L'U.S.M.G.

PROFESSEURS TITULAIRES

MM.	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
	BEAUDOING André	Clinique de Pédiatrie et Puériculture
	BERNARD Alain	Mathématiques Pures
Mme	BERTRANDIAS Françoise	Mathématiques Pures
MM.	BERTRANDIAS Jean-Paul	Mathématiques Pures
	BEZES Henri	Pathologie chirurgicale
	BLAMBERT Maurice	Mathématiques Pures
	BOLLIET Louis	Informatique (IUT B)
	BONNET Georges	Electrotechnique
	BONNET Jean-Louis	Clinique ophtalmologique
	BONNET-EYMARD Joseph	Clinique gastro-entérologique
Mme	BONNIER Marie-Jeanne	Chimie générale
MM.	BOUCHERLE André	Chimie et toxicologie
	BOUCHEZ Robert	Physique nucléaire
	BOUSSARD Jean-Claude	Mathématiques Appliquées
	BOUTET DE MONTVEL Louis	Mathématiques Pures
	BRAVARD Yves	Géographie
	CABANEL Guy	Clinique rhumatologique et hydrologique
	CALAS François	Anatomie
	CARLIER Georges	Biologie végétale
	CARRAZ Gilbert	Biologie animale et pharmacodynamie
	CAU Gabriel	Médecine légale et toxicologie
	CAUQUIS Georges	Chimie organique
	CHABAUTY Claude	Mathématiques Pures
	CHARACHON Robert	Clinique Oto-rhino-laryngologique
	CHATEAU Robert	Clinique de neurologie
	CHIBON Pierre	Biologie animale
	COEUR André	Pharmacie chimique et chimie analytique
	CONTAMIN Robert	Clinique gynécologique
	COUDERC Pierre	Anatomie pathologique
Mme	DEBELMAS Anne-Marie	Matière médicale
MM.	DEBELMAS Jacques	Géologie générale
	DEGRANGE Charles	Zoologie
	DELORMAS Pierre	Pneumophtisiologie

MM.	DEPORTES Charles	Chimie minérale
	DESRE Pierre	Métallurgie
	DESSAUX Georges	Physiologie animale
	DODU Jacques	Mécanique appliquée (IUT A)
	DOLIQUE Jean-Michel	Physique des plasmas
	DREYFUS Bernard	Thermodynamique
	DUCROS Pierre	Cristallographie
	DUGOIS Pierre	Clinique de dermatologie et syphiligraphie
	GAGNAIRE Didier	Chimie physique
	GALLISSOT François	Mathématiques Pures
	GALVANI Octave	Mathématiques Pures
	GASTINEL Noël	Analyse numérique
	GAVEND Michel	Pharmacologie
	GEINDRE Michel	Electroradiologie
	GERBER Robert	Mathématiques Pures
	GERMAIN Jean-Pierre	Mécanique
	GIRAUD Pierre	Géologie
	JANIN Bernard	Géographie
	KAHANE André	Physique générale
	KLEIN Joseph	Mathématiques Pures
	KOSZUL Jean-Louis	Mathématiques Pures
	KRAVTCHENKO Julien	Mécanique
	KUNTZMANN Jean	Mathématiques Appliquées
	LACAZE Albert	Thermodynamique
	LACHARME Jean	Biologie végétale
Mme	LAJZEROWICZ Janine	Physique
MM.	LAJZEROWICZ Joseph	Physique
	LATREILLE René	Chirurgie générale
	LATURAZE Jean	Biochimie pharmaceutique
	LAURENT Pierre-Jean	Mathématiques Appliquées
	LEDRU Jean	Clinique médicale B
	LLIBOUTRY Louis	Géophysique
	LOISEAUX Pierre	Sciences nucléaires
	LONGEQUEUE Jean-Pierre	Physique nucléaire
	LOUP Jean	Géographie
Mlle	LUTZ Elisabeth	Mathématiques Pures
MM.	MALGRANGE Bernard	Mathématiques Pures
	MALINAS Yves	Clinique obstétricale
	MARTIN-NOEL Pierre	Clinique cardiologique
	MAZARE Yves	Clinique médicale A
	MICHEL Robert	Minéralogie et Pétrographie
	MICOUD Max	Clinique maladies infectieuses
	MOURIQUAND Claude	Histologie
	MOUSSA André	Chimie nucléaire
	MULLER Jean-Michel	Thérapeutique (Néphrologie)
	NEEL Louis	Physique du Solide
	OZENDA Paul	Botanique
	PAYAN Jean-Jacques	Mathématiques Pures
	PEBAY-PEYROULA Jean-Claude	Physique
	RASSAT André	Chimie systématique
	RENARD Michel	Thermodynamique
	REVOL Michel	Urologie
	RINALDI Renaud	Physique
	DE ROUGEMONT Jacques	Neuro-chirurgie
	SEIGNEURIN Raymond	Microbiologie et Hygiène
	SENGEL Philippe	Zoologie

MM. SIBILLE Robert	Construction mécanique (IUT A)
SOUTIF Michel	Physique générale
TANCHE Maurice	Physiologie
TRAYNARD Philippe	Chimie générale
VAILLANT François	Zoologie
VALENTIN Jacques	Physique nucléaire
VAUQUOIS Bernard	Calcul électronique
Mme VERAIN Alice	Pharmacie galénique
MM. VERAIN André	Physique
VEYRET Paul	Géographie
VIGNAIS Pierre	Biochimie médicale
YOCCOZ Jean	Physique nucléaire théorique

PROFESSEURS ASSOCIES

MM. CLARK Gilbert	Spectrométrie physique
CRABBE Pierre	CERMO
ENGLMAN Robert	Spectrométrie physique
HOLTZBERG Frédéric	Basses températures
DEMBICKI Eugéniuz	Mécanique
MATSUSHIMA Yozo	Mathématiques Pures

PROFESSEURS SANS CHAIRE

Mlle AGNIUS-DELORD Claudine	Physique pharmaceutique
ALARY Josette	Chimie analytique
MM. AMBROISE-THOMAS Pierre	Parasitologie
BELORIZKY Elie	Physique
BENZAKEN Claude	Mathématiques Appliquées
BIAREZ Jean-Pierre	Mécanique
BILLET Jean	Géographie
BOUCHET Yves	Anatomie
BRUGEL Lucien	Energétique (IUT A)
BUISSON René	Physique (IUT A)
BUTEL Jean	Orthopédie
COHEN ADDAD Pierre	Spectrométrie physique
COLOMB Maurice	Biochimie
CONTE René	Physique (IUT A)
DEPASSEL Roger	Mécanique des fluides
FONTAINE Jean-Marc	Mathématiques Pures
GAUTHIER Yves	Sciences Biologiques
GAUTRON René	Chimie
GIDON Paul	Géologie et Minéralogie
GLENAT René	Chimie organique
GROULADE Joseph	Biochimie médicale
HACQUES Gérard	Calcul numérique
HOLLARD Daniel	Hématologie
HUGONOT Robert	Hygiène et Médecine préventive
IDELMAN Simon	Physiologie animale
JOLY Jean-René	Mathématiques Pures
JULLIEN Pierre	Mathématiques Appliquées
Mme KAHANE Josette	Physique
MM. KRAKOWIAK Sacha	Mathématiques Appliquées
KUHN Gérard	Physique (IUT A)
LE ROY Philippe	Mécanique (IUT A)

MM.	MAYNARD Roger	Physique du solide
Mme	MINIER Colette	Physique (IUT A)
MM.	PELMONT Jean	Biochimie
	PERRIAUX Jean-Jacques	Géologie et Minéralogie
	PFISTER Jean-Claude	Physique du solide
Mlle	PIERY Yvette	Physiologie animale
MM.	RAYNAUD Hervé	M.I.A.G.
	REBECQ Jacques	Biologie (CUS)
	REYMOND Jean-Charles	Chirurgie générale
	RICHARD Lucien	Biologie végétale
Mme	RINAUDO Marguerite	Chimie macromoléculaire
MM.	ROBERT André	Chimie papetière
	SARRAZIN Roger	Anatomie et chirurgie
	SARROT-REYNAULD Jean	Géologie
	SIROT Louis	Chirurgie générale
Mme	SOUTIF Jeanne	Physique générale
MM.	STREGLITZ Paul	Anesthésiologie
	VIALON Pierre	Géologie
	VAN CUTSEM Bernard	Mathématiques Appliquées

MAITRES DE CONFERENCES ET MAITRES DE CONFERENCES AGREGES

MM.	AMBLARD Pierre	Dermatologie
	ARMAND Gilbert	Géographie
	ARMAND Yves	Chimie (IUT A)
	BACHELOT Yvan	Endocrinologie
	BARGE Michel	Neuro-chirurgie
	BARJOLLE Michel	M.I.A.G.
	BEGUIN Claude	Chimie organique
Mme	BERIEL Hélène	Pharmacodynamie
MM.	BOST Michel	Pédiatrie
	BOUCHARLAT Jacques	Psychiatrie adultes
Mme	BOUCHE Liane	Mathématiques (CUS)
MM.	BRODEAU François	Mathématiques (IUT B)
	CHAMBAZ Edmond	Biochimie médicale
	CHAMPETIER Jean	Anatomie et organogénèse
	CHARDON Michel	Géographie
	CHERADAME Hervé	Chimie papetière
	CHIAVERINA Jean	Biologie appliquée (EFP)
	CONTAMIN Charles	Chirurgie thoracique et cardio-vasculaire
	CORDONNIER Daniel	Néphrologie
	COULOMB Max	Radiologie
	CROUZET Guy	Radiologie
	CYROT Michel	Physique du solide
	DELOBEL Claude	M.I.A.G.
	DENIS Bernard	Cardiologie
	DOUCE Roland	Physiologie végétale
	DUSSAUD René	Mathématiques (CUS)
Mme	ETERRADOSSI Jacqueline	Physiologie
MM.	FAURE Jacques	Médecine légale
	FAURE Gilbert	Urologie
	GAUTIER Robert	Chirurgie générale
	GENSAC Pierre	Botanique
	GIDON Maurice	Géologie
	GROS Yves	Physique (IUT A)

MM. GUITTON Jacques	Chimie
HICTER Pierre	Chimie
IVANES Marcel	Electricité
JALBERT Pierre	Histologie
JUNIEN-LAVILLAVROY Claude	O.R.L.
KOLODIE Lucien	Hématologie
LE NOC Pierre	Bactériologie-virologie
LEROY Philippe	IUT A
MACHE Régis	Physiologie végétale
MAGNIN Robert	Hygiène et médecine préventive
MALLION Jean-Michel	Médecine du travail
MARECHAL Jean	Mécanique (IUT A)
MARTIN-BOUYER Michel	Chimie (CUS)
MICHOULIER Jean	Physique (IUT A)
NEGRE Robert	Mécanique (IUT A)
NEMOZ Alain	Thermodynamique
NOUGARET Marcel	Automatique (IUT A)
PARAMELLE Bernard	Pneumologie
PECCOUD François	Analyse (IUT B)
PEFFEN René	Métallurgie (IUT A)
PERRET Jean	Neurologie
PERRIER Guy	Géophysique - Glaciologie
PHELIP Xavier	Rhumatologie
RACHAIL Michel	Médecine interne
RACINET Claude	Gynécologie et obstétrique
RAMBAUD André	Hygiène et hydrologie
RAMBAUD Pierre	Pédiatrie
Mme RENAUDET Jacqueline	Bactériologie
MM. ROBERT Jean-Bernard	Chimie Physique
ROMIER Guy	Mathématiques (IUT B)
SHOM Jean-Claude	Chimie générale
STOEBNER Pierre	Anatomie pathologique
VROUSOS Constantin	Radiologie

MAITRE DE CONFERENCES ASSOCIES

M. COLE Antony	Sciences nucléaires

Fait à SAINT MARTIN D'HERES, AVRIL 1976.

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. Philippe TRAYNARD

Vice-Président : M. Pierre-Jean LAURENT

PROFESSEURS TITULAIRES

MM. BENOIT Jean	Radioélectricité
BESSON Jean	Electrochimie
BLOCH Daniel	Physique du solide
BONNETAIN Lucien	Chimie Minérale
BONNIER Etienne	Electrochimie et Electrométallurgie
BOUDOURIS Georges	Radioélectricité
BRISSONNEAU Pierre	Physique du solide
BUYLE-BODIN Maurice	Electronique
COUMES André	Radioélectricité
DURAND Francis	Métallurgie
FELICI Noël	Electrostatique
FOULARD Claude	Automatique
LESPINARD Georges	Mécanique
MOREAU René	Mécanique
PARIAUD Jean-Charles	Chimie-Physique
PAUTHENET René	Physique du solide
PERRET René	Servomécanismes
POLOUJADOFF Michel	Electrotechnique
SILBER Robert	Mécanique des Fluides

PROFESSEUR ASSOCIE

M. ROUXEL Roland	Automatique

PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuel	Electronique
BOUVARD Maurice	Génie Mécanique
COHEN Joseph	Electrotechnique
LACOUME Jean-Louis	Géophysique
LANCIA Roland	Electronique
ROBERT François	Analyse numérique
VEILLON Gérard	Informatique Fondamentale et Appliquée
ZADWORNY François	Electronique

MAITRES DE CONFERENCES

MM. ANCEAU François	Mathématiques Appliquées
CHARTIER Germain	Electronique
GUYOT Pierre	Chimie Minérale
IVANES Marcel	Electrotechnique
JOUBERT Jean-Claude	Physique du solide
MORET Roger	Electrotechnique Nucléaire
PIERRARD Jean-Marie	Mécanique
SABONNADIERE Jean-Claude	Informatique Fondamentale et Appliquée

MAITRE DE CONFERENCES ASSOCIE

M. LANDAU Ioan	Automatique

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

MM. FRUCHART Robert	Directeur de Recherche
ANSARA Ibrahim	Maître de Recherche
CARRE René	Maître de Recherche
DRIOLE Jean	Maître de Recherche
MATHIEU Jean-Claude	Maître de Recherche
MUNIER Jacques	Maître de Recherche

Je tiens à rendre hommage à la mémoire de Monsieur du MASLE qui m'a confié ce travail et m'a aidé de ses conseils dans cette étude.

Je remercie Monsieur GASTINEL, qui a bien voulu me faire l'honneur de présider le Jury de cette thèse, ainsi que Messieurs les membres du jury, qui ont manifesté leur intérêt pour mon travail.

Je remercie également tous ceux dont les conseils et les remarques m'ont permis de mener à bien cette étude :
Monsieur H.ZIMMERMANN de l'équipe CYCLADES/IRIA, Monsieur REY de l'équipe CP/CMS du CICG, Messieurs J.P.ANSART, N.X.DANG, R.FOURNIER et G.SERGEANT de l'équipe Réseaux de l'ENSIMAG; Messieurs E.ANDRE, P.DECITRE et J.SEGUIN du Centre Scientifique CII-HB.
La réalisation de cette étude doit beaucoup à leur collaboration.

J'exprime enfin ma gratitude à la secrétaire qui a assuré la frappe de ce mémoire, ainsi qu'au service reprographie du CICG qui en a assuré le tirage.

TABLE DES MATIERES

PRESENTATION

Le développement actuel des réseaux d'ordinateurs pose de façon nouvelle le problème de la fiabilité et de la disponibilité des services offerts par un Centre de Calcul connecté. En effet, la multiplicité des matériels et des logiciels mis en jeu dans une application accédée à travers un réseau introduit une probabilité d'incident ou de panne plus forte que dans une application accédée localement.
Par ailleurs la multiplicité des ressources disponibles doit permettre une permanence de service malgré les pannes de certains composants.

Cette étude analyse l'impact du réseau sur la disponibilité des services qu'il offre aux utilisateurs. Elle est axée sur le réseau CYCLADES et notamment sur les services actuellement en exploitation. Les solutions décrites ici sont celles qui ont été adoptées pour les logiciels réseau en exploitation sur l'IBM 360/67 du Centre Interuniversitaire de Calcul de Grenoble.

Le premier chapitre présente le réseau CYCLADES, son architecture et ses protocoles.

Le deuxième chapitre est consacré à l'étude des différentes pannes qui peuvent se produire sur le réseau. Les possibilités de réagir à ces pannes ou de les éviter sont présentées.

Les deux chapitres suivants décrivent les solutions retenues pour l'implémentation : le système de multiprogrammation SYNCOP et les procédures de reprises sur panne des serveurs du 360/67.

Le dernier chapitre évoque d'autres solutions qui pourront être adoptées plus tard.

CHAPITRE I

LE RESEAU CYCLADES

CYCLADES est un réseau d'ordinateurs général et hétérogène [1]. N'étant pas réservé à une application particulière, comme les réseaux des banques ou des compagnies de transport par exemple, CYCLADES est un réseau général. Les ordinateurs et terminaux connectés sont de types et de marques différents : CYCLADES est un réseau hétérogène. Conçu pour expérimenter en vrai grandeur la conception, l'utilisation et l'exploitation d'un réseau, il relie entre eux des Centres de Recherche et des Universités qui participent à la connexion des machines au réseau, à la définition et à la réalisation des applications.

Les premières applications développées sur le réseau sont passées à la phase d'exploitation [2]. Le but de ces applications est de mettre à la disposition des utilisateurs du réseau les services disponibles sur les machines connectées. Ces services sont essentiellement les systèmes de temps partagé et de traitement par lots des gros ordinateurs, ainsi que les applications disponibles sous ces systèmes. Des concentrateurs de terminaux permettent l'accès à ces services.

D'autres applications sont en cours de développement. Le projet pilote base de données réparties a pour but de permettre l'accès en parallèle à des bases de données réparties sur plusieurs sites disposant de matériel et de logiciel différents. Un langage de commande pour le réseau est en cours d'étude. Il fournira un outil d'uniformisation du réseau. Des mini-concentrateurs travaillant directement avec des paquets du réseau de communication sont à l'essai.

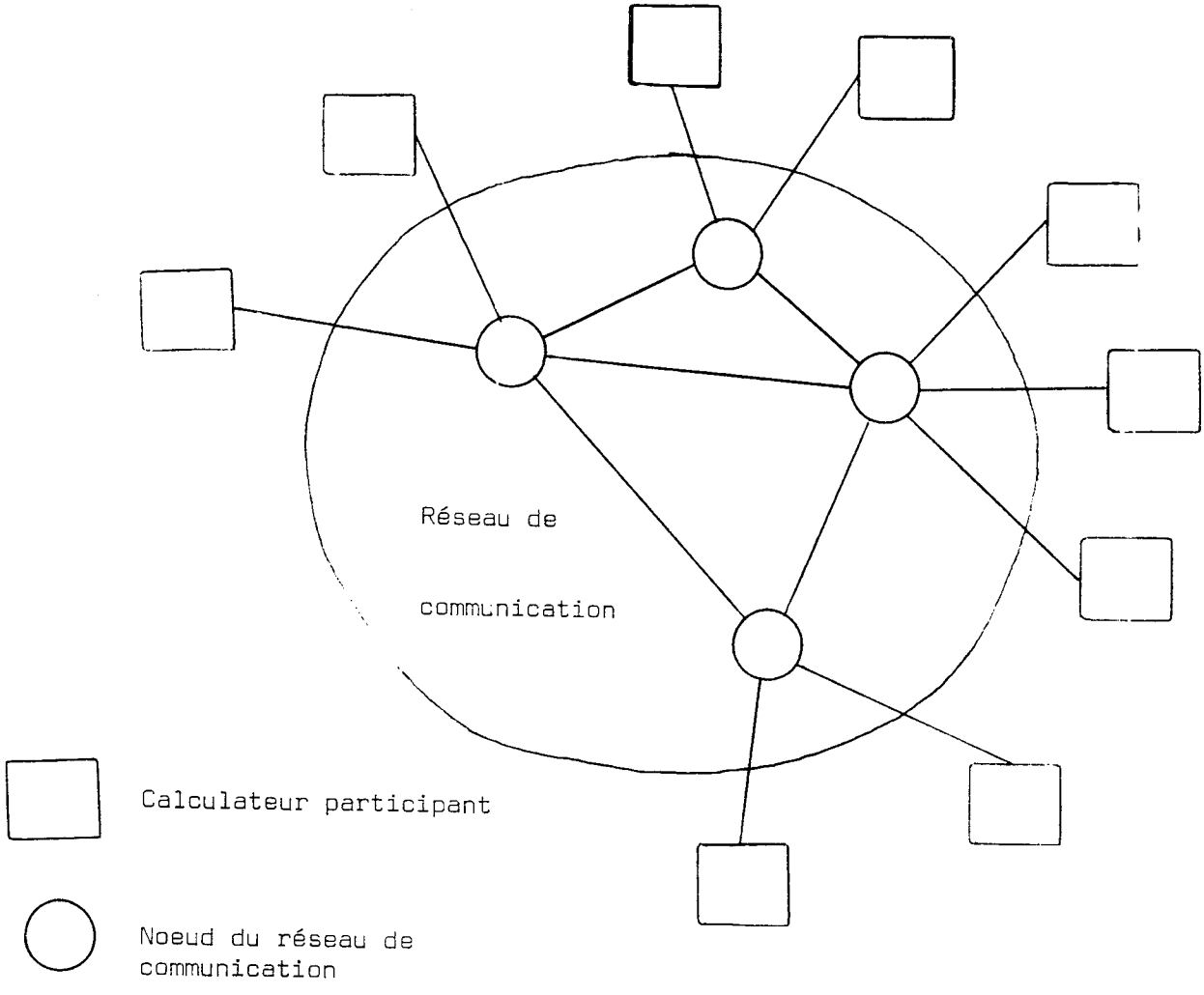
1. LES DEUX NIVEAUX DU RÉSEAU CYCLADES

Comme tous les réseaux généraux hétérogènes à l'étude et comme son prédécesseur, le réseau américain ARPA, CYCLADES est un réseau à deux niveaux.

Le premier niveau est constitué d'ordinateurs CII MITRA 15 reliés entre eux par des lignes de transmission synchrones et formant un réseau maillé, appelé réseau de communication. Les MITRA 15 noeuds du réseau de communication (ou calculateurs de liaison) gèrent également des lignes synchrones par lesquelles sont connectés les ordinateurs de CYCLADES (ou calculateurs participants).
La fonction du réseau de communication (appelé CIGALE) est de transmettre d'un point à un autre des paquets d'information de longueur limitée indépendamment les uns des autres et avec un taux d'erreur limité, mais non nul.

Le deuxième niveau est constitué de tous les calculateurs participants qui communiquent entre eux par l'intermédiaire de CIGALE. L'interface CIGALE-calculateur participant est transparent au contenu des paquets et n'impose aucun protocole particulier si ce n'est la procédure de transmission utilisée pour la gestion de la ligne et le format d'en-tête des paquets.

Les deux niveaux de CYCLADES sont indépendants.



Les deux niveaux du réseau

2. LE RÉSEAU DE COMMUNICATION CIGALE

CIGALE utilise la technique de commutation de paquets. Chaque paquet reçu par un noeud est stocké dans le noeud avant d'être transmis vers un autre noeud ou vers le calculateur participant destinataire. L'itinéraire que suivra un paquet n'est pas préétabli. Des paquets ayant même origine et même destination peuvent voyager par des chemins différents suivant la disponibilité des éléments du réseau : le routage est adaptatif.

Des paquets expédiés dans un certain ordre par un calculateur participant vers un autre calculateur participant peuvent donc éventuellement arriver dans un ordre différent à leur destinataire. De plus, des paquets peuvent exceptionnellement se perdre et ne pas parvenir à leur destinatation, ou être dupliqués par CIGALE et arriver en double exemplaire au récepteur.

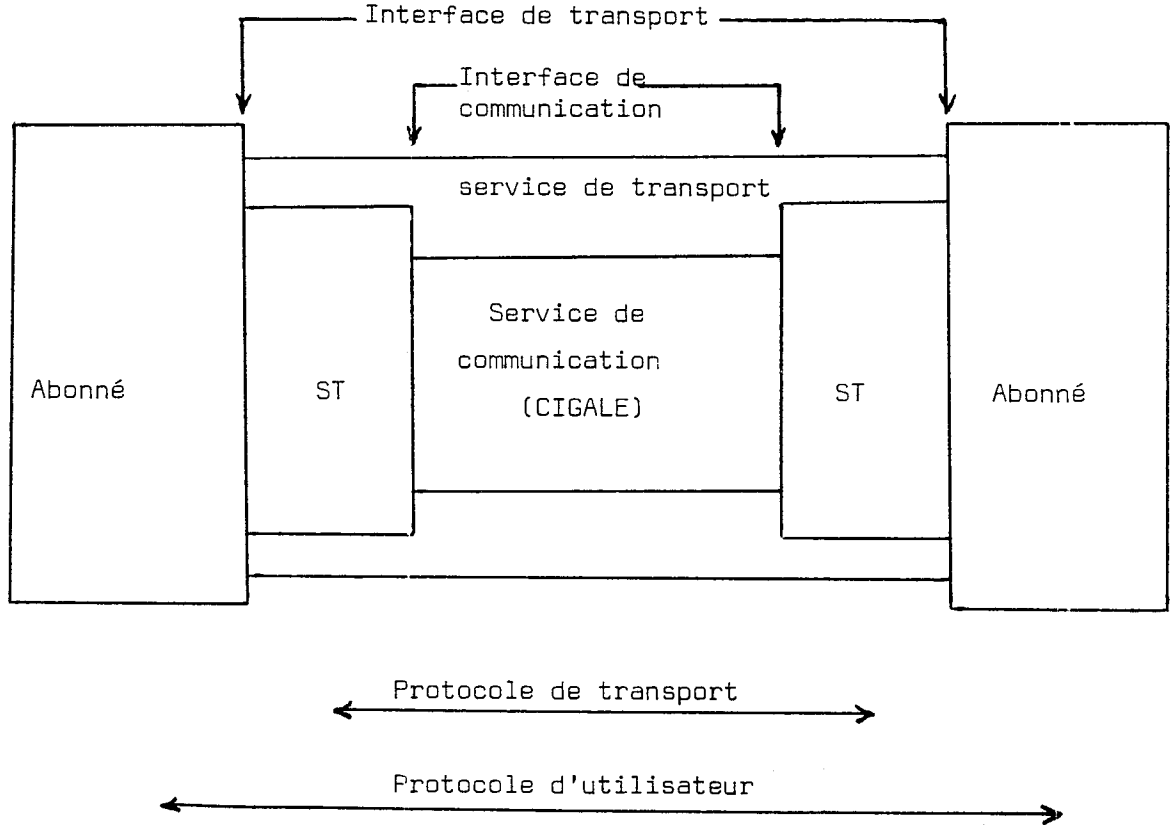
Ces "défauts" sont la conséquence du choix de simplicité qui a été fait pour CIGALE. Les traitements complexes de fragmentation, réassemblage, reprise sur perte, élimination des doubles, séquencement sont reportés au niveau des calculateurs participants, CIGALE n'assurant que la "simple" fonction de transport de paquets indépendants.

3. LES STATIONS DE TRANSPORT

Les utilisateurs (ou abonnés) de deux sites éloignés communiquent entre eux en utilisant le réseau de communication. Ils y accèdent par des stations de transport situées dans chaque site. La station de transport constitue l'interface entre les abonnés et le réseau. Son rôle est triple :

- offrir aux abonnés un accès simple au réseau
- reprendre les erreurs et 'défauts' de CIGALE
- multiplexer plusieurs voies logiques (flots) sur une seule ligne physique.

Tout calculateur participant comporte une station de transport (ST). On peut présenter une application réseau comme deux (ou plusieurs) abonnés communiquant entre eux suivant un protocole qui leur est propre (protocole utilisateur) en utilisant le service de transport.
Le service de transport est réalisé par les ST qui communiquent entre elles suivant le protocole de transport en utilisant le service de communication offert par CIGALE.



L'interface de communication permet à la station de transport d'émettre et de recevoir des paquets CIGALE. L'interface de transport permet aux abonnés de passer à la station de transport des directives de transport et d'obtenir les réponses.

4. LE SERVICE DE TRANSPORT

Le service et le protocole de transport décrits ici correspondent à la deuxième version de ceux utilisés dans CYCLADES [4]. L'expérience acquise avec la première version [5] a permis de définir un protocole et un service de transport plus simples et correspondant mieux aux besoins réels des utilisateurs.

4.1. Adressage

A l'intérieur du réseau, chaque station de transport possède un identificateur unique, son numéro de ST (ST-NB). Chaque abonné peut avoir plusieurs portes (PT) d'accès au réseau par lesquelles il peut émettre et recevoir. A l'intérieur d'une ST chaque porte a un identificateur unique, son identificateur de porte (PT-ID). Une porte est donc identifiée de façon unique dans le réseau par le couple ST-NB/PT-ID. Un couple de deux portes appartenant à deux abonnés qui correspondent entre eux s'appelle un flot. Un flot est identifié (FL-ID) par le couple des identificateurs des portes qui le composent.

4.2. Service de transport de base

Le service de base donne aux abonnés la possibilité d'échanger sur un flot, dans les deux directions, des lettres et des télégrammes. Les lettres (LT) sont des éléments d'information de longueur variable mais limitée. La taille limite d'une lettre est supérieure à la taille d'un paquet, de l'ordre de quelques milliers d'octets. Les télégrammes (TG) sont des éléments de deux octets acheminés indépendamment du trafic des lettres sur le flot. Le service de base fonctionne avec les lettres et les télégrammes comme le service de communication avec les paquets. Des lettres ou des télégrammes peuvent être perdus ou éventuellement arriver dans un ordre différent de l'ordre d'émission, ou encore être dupliqués. Aucun contrôle particulier n'est fait.

4.3. Services additionnels

Les abonnés qui ne peuvent se contenter de ce service de base peuvent demander à leur ST des services additionnels. Ces services additionnels seront appliquées à un flot sur l'accord des deux correspondants après négociation. On appelle session le temps pendant lequel les services additionnels sont disponibles sur un flot.

Deux types de services additionnels peuvent être demandés :

- le contrôle d'erreur
- le contrôle de flux

Le contrôle d'erreur opère dans les deux sens d'un flot et assure que toutes les lettres émises arrivent sans erreur à la ST de leur destinataire, dans l'ordre d'émission et sans duplication. Mais on n'a aucune certitude sur leur prise en compte par l'abonné destinataire. Les lettres sont mises à sa disposition, on ne peut être sûr qu'il les a retirées et traitées.

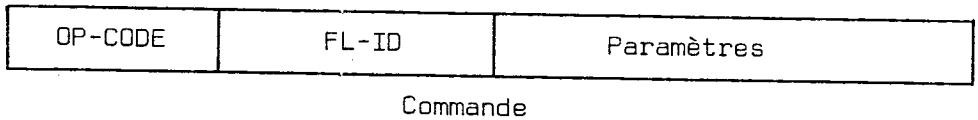
Le contrôle de flux opère dans les deux sens d'un flot qui doit être également soumis au contrôle d'erreur. Il régule le débit d'émission des lettres suivant la capacité du récepteur. Les contrôles de flux et d'erreur s'appliquent uniquement aux lettres. Les télégrammes en sont exclus.

5. LE PROTOCOLE DE TRANSPORT

Le protocole de transport, ou protocole ST-ST, définit les règles d'échange des commandes entre ST et leur signification. Ces commandes sont transportées par CIGALE à raison d'une commande par paquet. Chaque paquet contient dans son en-tête les numéros de la ST émettrice et de la ST destinataire.

Les commandes contenues dans les paquets sont formées :

- d'un code indiquant le type de la commande (OP-CODE)
- de l'identificateur du flot concerné par la commande (FL-ID)
- de paramètres contenant des données et des informations de contrôle.



Commande

5.1. Les télégrammes

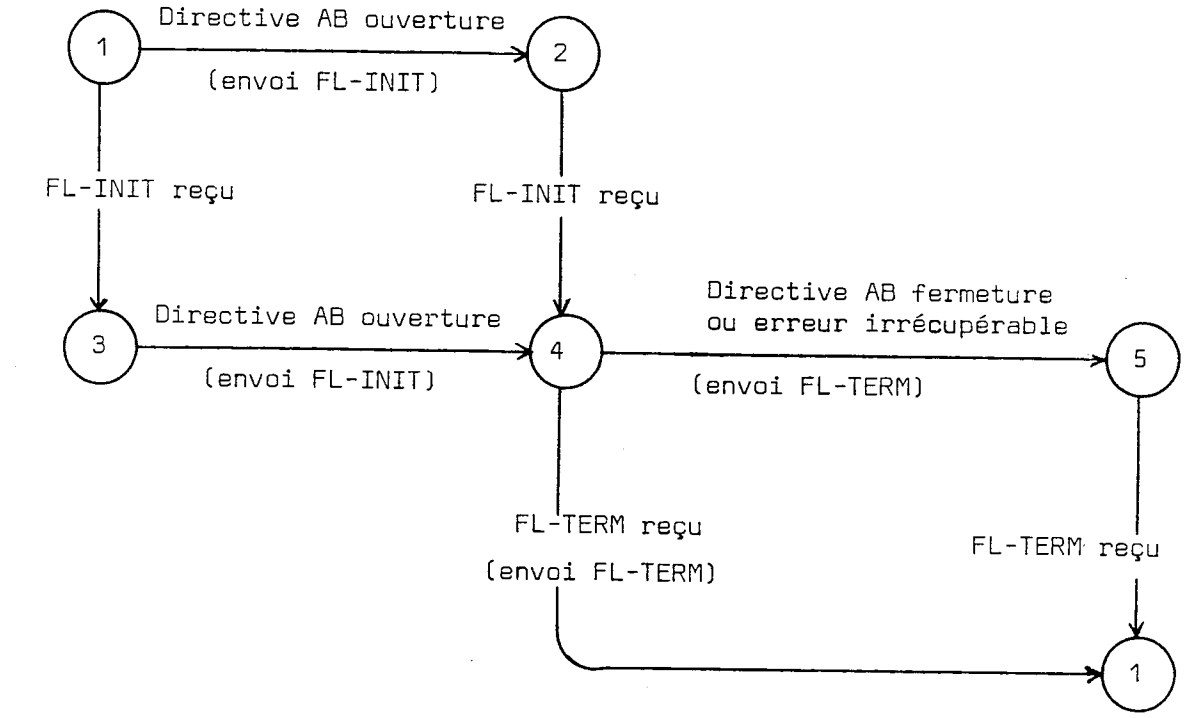
Une seule commande (OP-CODE=FL-TG) concerne les télégrammes. Elle permet d'envoyer un télégramme sur un flot. Le seul paramètre de cette commande est le texte du télégramme (TG-TEXT). Il n'y a pas de commande permettant d'accuser réception d'un télégramme. Un télégramme peut donc se perdre sans que l'émetteur le sache.

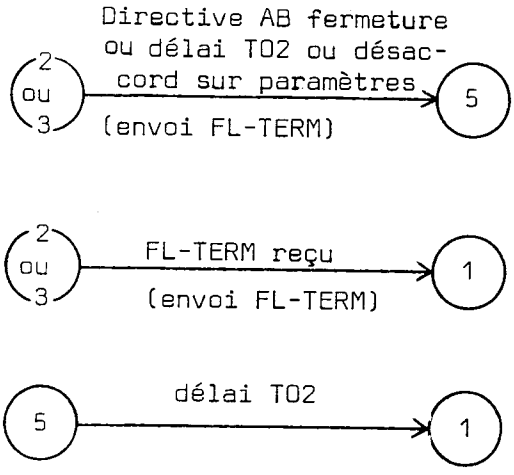
5.2. Négociation de session

Deux commandes permettent de négocier une session.
La commande FL-INIT est utilisée pour ouvrir une session sur un flot. Elle permet de définir les services additionnels demandés sur le flot. Ses paramètres sont la liste des services demandés (contrôle d'erreur ou contrôle d'erreur plus contrôle de flux) et, si le contrôle de flux est demandé, la taille maximum des lettres (MY-LT-LG) qui seront émises sur le flot par la ST envoyant la commande.
Pour fermer une session sur un flot ou pour refuser l'ouverture d'une session, on utilise la commande FL-TERM, dont le seul paramètre est la raison de la fermeture ou du refus.
Pour une ST, une session est effectivement ouverte quand elle a reçu la directive d'ouverture de l'abonné local et la commande FL-INIT de la ST distante, dans un ordre quelconque mais dans un délai limité. Il faut de plus que les services demandés dans la directive et la commande soient les mêmes et que la taille maximum des lettres admises en réception par l'abonné local (précisée dans sa directive) soit compatible avec le paramètre taille maximum contenu dans la commande reçue, si le contrôle de flux est demandé.
La session est fermée par la ST si

- l'abonné local le demande
- la ST reçoit une commande FL-TERM
- si une erreur irrécupérable survient pendant la session
- si un délai trop long s'écoule entre les demandes locales (directive abonné) et distante (commande de l'autre ST).

On peut représenter la négociation de session par l'automate ci-dessous :





Les évènements qui provoquent les transitions d'état de la session sont indiqués sur les flèches, les commandes envoyées par la ST au moment de la transition sont données entre parenthèses.

Etats de la session :

(1) Fermée

(2) En cours d'ouverture sur demande locale

(3) En cours d'ouverture sur demande distante

(4) Ouverte

(5) En cours de fermeture sur demande locale.

Chaque fois que la session passe dans un état intermédiaire ((2),(3),(5)) la ST arme un réveil. A l'arrivée d'un évènement provoquant un changement d'état le réveil est supprimé. Si le réveil sonne (évènement : délai TO2), il y a changement d'état. On évite ainsi qu'une session reste indéfiniment dans un état intermédiaire.

5.3. Fragmentation - Réassemblage

Les lettres ayant une taille maximum supérieure aux paquets doivent le cas échéant être découpées en fragments pour être transportées par CIGALE. La taille des fragments est telle qu'ils puissent tenir dans un paquet.
La ST émettrice découpe donc les lettres en fragments qui sont envoyés chacun comme paramètre (FR-TEXT) d'une commande FL-LT. Ces fragments sont réassemblés et réordonnés par la ST réceptrice qui remet la lettre complète à l'abonné destinataire. Pour assembler les fragments reçus, la ST dispose du paramètre FR-NB contenu dans la commande FL-LT. Ce paramètre donne le numéro du fragment dans la lettre. S'il y a contrôle d'erreur sur le flot, un paramètre supplémentaire (MY-REF) est ajouté dans la commande FL-LT pour assurer que les lettres seront remises à l'abonné destinataire dans l'ordre où elles ont été émises. Ce paramètre donne le numéro de la lettre à laquelle appartient le fragment.
Si un délai TO1 (délai de réassemblage) s'est écoulé depuis la dernière réception d'un fragment d'une lettre, la lettre est considérée comme incomplète et on n'attend plus le ou les fragments manquant.

5.4. Contrôle d'erreur

Sur un flot ouvert avec contrôle d'erreur la ST émettrice envoie des lettres en les numérotant séquentiellement (MY-REF) et attend un acquittement dans un délai maximum T03 après l'émission du dernier fragment de la lettre.

La ST réceptrice acquitte les lettres reçues en envoyant une commande avec le paramètre YR-REF. Ce paramètre donne le numéro ou référence (MY-REF pour l'émetteur de la lettre) de la dernière lettre correctement reçue et indique que cette lettre et toutes les précédentes ont été correctement reçues.

Si la ST émettrice ne reçoit pas d'acquittement au bout du temps T03, elle suppose que toutes les lettres émises et non acquittées sont perdues et les réémet. Elle attend à nouveau l'acquittement de ces lettres dans le délai T03.
Si une lettre a été émise N fois sans succès, la ST déclare une erreur irrécupérable sur le flot et ferme la session.
Une activité minimum est maintenue sur les flots fonctionnant en contrôle d'erreur. Les ST envoient sur chaque flot un acquittement de la dernière lettre reçue (ou de la lettre fictive de référence 0 si aucune lettre n'a été reçue) à chaque fois qu'il s'est écoulé un temps T05 depuis l'émission de la dernière commande FL-ACK ou FL-LT.
Les acquittements peuvent être envoyés sous deux formes :

- commande FL-ACK (paramètre YR-REF), s'il n'y a pas de fragment à envoyer sur le flot,
- commande FL-LT avec le paramètre supplémentaire YR-REF s'il y a un fragment à envoyer.

5.5. Contrôle de flux

Sur un flot ouvert en contrôle de flux les ST s'envoient des "crédits" qui indiquent le nombre de lettres que la ST émettrice des lettres peut envoyer sans dépasser les capacités de réception de la ST réceptrice. Ces crédits

sont envoyés en paramètre (CRD-NB) des commandes FL-LT et FL-ACK, ils sont associés à la référence de lettre acquittée (YR-REF) contenue dans la commande. La ST qui reçoit une telle commande a l'autorisation d'émettre les lettres de référence (YR-REF)+1 jusqu'à (YR-REF)+(CRD-NB). Si (CRD-NB)=0 la ST ne peut pas émettre de lettre sur ce flot.

5.6. Liste des commandes

Les commandes échangées entre les ST sont donc les suivantes :

- FL-LT/FL-ID/[YR-REF/[CRD-NB/]]MY-REF/FR-NB/EOL/FR-TEXT

  Le paramètre YR-REF n'est utilisé qu'avec le contrôle d'erreur
  Le paramètre CRD-NB n'est utilisé qu'avec le contrôle de flux
  Le paramètre EOL indique si le fragment est le dernier de la lettre.

- FL-ACK/FL-ID/YR-REF/[CRD-NB]

  Le paramètre CRD-NB n'est utilisé qu'avec le contrôle de flux.

- FL-TG/FL-ID/TG-TEXT

- FL-INIT/FL-ID/SERVICE/[MY-LT-LG/]

  Le paramètre SERVICE indique les services demandés
  Le paramètre MY-LT-LG n'est utilisé qu'avec le contrôle de flux.

- FL-TERM/FL-ID/RAISON

5.7. Remarque

Le protocole décrit ici est un sous-ensemble du protocole ST-ST CYCLADES. Le protocole complet [4] comprend en plus une possibilité d'adressage réduit qui permet après négociation d'utiliser une identification de flot (FL-ID) tenant moins de place dans les commandes.
Une commande supplémentaire (FL-ERROR) permet à une ST qui ne comprend pas une commande reçue de le signaler à la ST émettrice de la commande invalide.

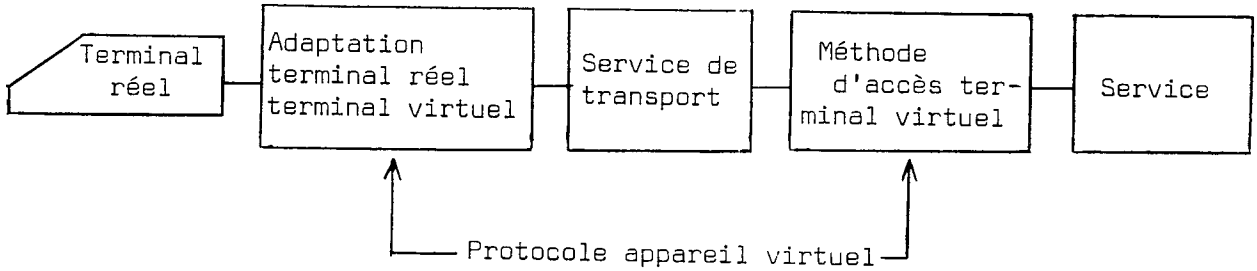
6. LES PROTOCOLES UTILISATEURS

Le protocole de base qui doit impérativement être implémenté dans tous les calculateurs participants de CYCLADES est le protocole de transport qui donne accès au service de transport. Le service de transport étant alors disponible sur tous les calculateurs participants,on peut définir des applications réseau mettant en jeu certains de ces calculateurs participants. Une application est constituée de plusieurs correspondants géographiquement répartis qui coopèrent entre eux en dialoguant à l'aide du service de transport. Pour qu'ils puissent se comprendre, les règles du dialogue doivent être définies. L'ensemble de ces règles forme le protocole utilisateur de l'application. Certains protocoles ont été définis d'une façon standard pour l'ensemble du réseau. D'autres peuvent être définis pour de nouvelles applications. Seuls les protocoles utilisés par les applications en exploitation au CICG seront décrits ici.

6.1. Le protocole appareil virtuel

Défini pour permettre à tout terminal du réseau l'accès aux différents services connectés, le protocole appareil virtuel (PAV) vise à présenter au réseau tous les terminaux comme des appareils standard. Inversement chaque service qui communique avec des terminaux "voit" tous les terminaux du réseau de la même façon. Ainsi pour mettre à la disposition des utilisateurs du réseau un nouveau service, il suffit que ce service sache gérer un seul type de terminal, l'appareil virtuel. Pour connecter au réseau un nouveau modèle de terminal, il suffit de présenter au réseau ce terminal comme l'appareil virtuel et aucune modification des services existants n'est nécessaire.

L'appareil virtuel peut représenter des terminaux conversationnels (machine à écrire, écran alphanumérique...) et des terminaux lourds (lecteur de cartes, imprimante). Il présente les caractéristiques que l'on trouve sur la plupart des terminaux disponibles actuellement. On peut schématiser la connexion d'un terminal réel à un service de la façon suivante :



Nous décrivons la deuxième version [6] du protocole appareil virtuel qui est une extension de la version 1 [7].
Le PAV est formé de quatre sous-ensembles :
- un protocole de connexion
- un protocole de contrôle du dialogue
- un protocole de contrôle d'application
- un protocole de négociation d'options.

6.2. Le protocole de connexion

Avant de pouvoir dialoguer, deux correspondants doivent se connecter, c'est-à-dire établir un flot avec contrôle d'erreur et contrôle de flux. En effet, aucun message du dialogue ne doit être perdu et leur ordre doit être respecté. De plus on ne veut pas saturer un des correspondants avec un débit de lettres trop important pour ses capacités de traitement. Les correspondants sont un serveur et un client. Un serveur est l'interface entre un service et le réseau. Un client est l'interface entre le réseau et un utilisateur (homme ou programme) qui veut accéder à un service.

Le serveur S est à l'écoute du réseau : il possède une porte s, connue des utilisateurs, sur laquelle il attend les demandes de connexion.

Le client C qui veut travailler avec S remet au service de transport sur sa porte c une directive d'ouverture du flot (c,s). Le service de transport avertit S de cette demande. Si S accepte, il fait la demande symétrique

d'ouverture du flot (c;s) et le service de transport prévient C que le flot est ouvert. Le dialogue entre C et S peut alors commencer. On voit que cette méthode de connexion n'est pas symétrique. Le client a l'initiative de la connexion tandis que le serveur attend les demandes des clients.

Une fois la connexion établie l'une de ses extrêmités peut être basculée sur une autre porte. Par exemple, S est un service de renseignements qui indique les autres services disponibles. Par le flot (c,s), C demande à accéder à un service S1. S envoie alors à C, toujours sur (c,s) un message SWITCH signalant à C de fermer le flot (c,s) et d'ouvrir un nouveau flot (c,s1) de la même façon que le flot (c,s). s1 est la porte à laquelle C doit s'adresser pour travailler avec S1.

La commande SWITCH a un seul paramètre, l'adresse de la nouvelle porte (s1 dans l'exemple) :

SWITCH/PT-ID

6.3. Le protocole de contrôle de dialogue

Trois modes de dialogue sont définis pour l'échange des messages de contrôle de l'application :

- Dialogue dans les deux sens simultanément (full duplex). C'est le mode de dialogue utilisé par exemple entre programmes.
- Dialogue dans les deux sens alternativement (ou à l'alternat). A un instant donné seule une extrêmité a le droit d'émettre. Quand elle ne veut plus émettre, elle envoie 'à vous' à l'autre extrêmité qui a alors le droit d'émettre. Ce mode de dialogue est utilisé entre un terminal et un service de temps partagé par exemple. A l'ouverture du flot, c'est le serveur qui a le droit d'émettre.
- Dialogue dans un seul sens (simplex). C'est le cas des lecteurs de cartes et imprimantes.

Les messages de contrôle du dialogue permettent :

a/ De changer le sens de la transmission dans un dialogue à l'alternat. A-VOUS, bien que faisant partie des messages de contrôle de dialogue

est envoyé dans un message de contrôle d'application pour diminuer le nombre de messages échangés.

b/ D'envoyer une attention dans un télégramme. L'attention est ainsi envoyée indépendamment des autres messages.

ATTN/valeur de l'attn (télégramme)

c/ D'envoyer des fonctions

FUNCT/valeur de la fonction (lettre)

d/ D'échanger les états des correspondants.

On peut demander l'état du correspondant par :

VOTRE-ETAT (télégramme)

On peut envoyer son propre état au correspondant soit spontanément soit en réponse à VOTRE-ETAT :

MON-ETAT/valeur de l'état (lettre)

L'état peut prendre les valeurs : prêt, non prêt,hors fonction,inactif.

6.4. Le protocole de contrôle d'application

Le protocole de contrôle d'application pour le terminal virtuel permet d'échanger du texte dont la structure, appellée bouquin est la suivante :

Bouquin = suite de pages

Page = suite de lignes

Ligne = suite de caractères.

Les longueurs des bouquins, pages et lignes sont indéfinies.

On envoie le texte (code EBCDIC) par éléments (EL-TXT) qui sont contenus dans des lettres. Chaque élément est formé de caractères consécutifs. En tête de chaque élément une partie adresse (ADR) indique si le texte de l'élément

- débute une nouvelle page (ADR=NP)
- débute une nouvelle ligne (ADR=NL)
- constitue la suite de la ligne en cours (ADR=CN)

La longueur (LG-EL) de l'élément est également en tête de l'élément.

Plusieurs éléments (avec leur en-tête) peuvent être rangés dans une lettre de texte qui contient elle-même un en-tête indiquant qu'il s'agit de texte (LT-TXT) et contenant l'indicateur A-VOUS.
La structure générale du message est donc :

<message>::=<en-tête de message><suite d'éléments>
<en-tête de message>::=LT-TXT A-VOUS
<suite d'éléments>::=<élément>|<suite d'éléments>
<élément>::=LG-EL ADR EL-TXT|LG-EL ADR

6.5. Protocole de négociation d'options

Ce protocole permet aux deux correspondants de se mettre d'accord sur les caractéristiques du terminal virtuel. En effet le terminal virtuel décrit plus haut est le terminal minimum, implicitement admis par les correspondants. On peut définir des caractéristiques supplémentaires telles que :

- addressage différentiel
- code différent du code EBCDIC
- plusieurs jeux de caractères
- compactage
- etc...

Si elles sont utilisées, ces caractéristiques doivent être négociées grâce à deux types de messages qui sont envoyés par lettre :
D-OPTION permet de demander ou d'accepter une option D-OPTION/CODE/paramètres
CODE est le code de l'option demandée ou acceptée, les paramètres sont relatifs à l'option.
REFUSE (sans paramètre) permet de refuser une option.

CHAPITRE II

LES PROBLEMES DE FIABILITE ET REPRISE

Les premières applications réalisées sur le réseau visaient à mettre les services disponibles localement sur les calculateurs participants à la disposition de la plus large communauté d'utilisateurs. Ces services sont les systèmes d'exploitation des calculateurs participants (temps partagé sous SIRIS-8, système CP/67, traitement par lots sous OS-360, sous SIRIS-8 etc ...). Dans ces applications de type client-serveur, le réseau doit être transparent à l'utilisateur qui doit 'voir' le service auquel il s'adresse depuis son terminal comme le voit un utilisateur local. Le réseau doit introduire le moins de distorsions possibles, faute de quoi, l'expérience l'a prouvé, les utilisateurs fuient le réseau qui reste l'outil des seuls initiés.

Parmi ces distorsions, la plus rédhibitoire est la détérioration de la fiabilité* et de la disponibilité* d'un service à partir d'un terminal du réseau. Si l'utilisateur accepte que son terminal ne s'utilise pas exactement comme un terminal connecté directement, il tolère difficilement une diminution de la disponibilité du service. Les matériels et les logiciels mis en jeu sont nombreux et la moindre panne d'un de ces éléments peut compromettre le travail de l'utilisateur.

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

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

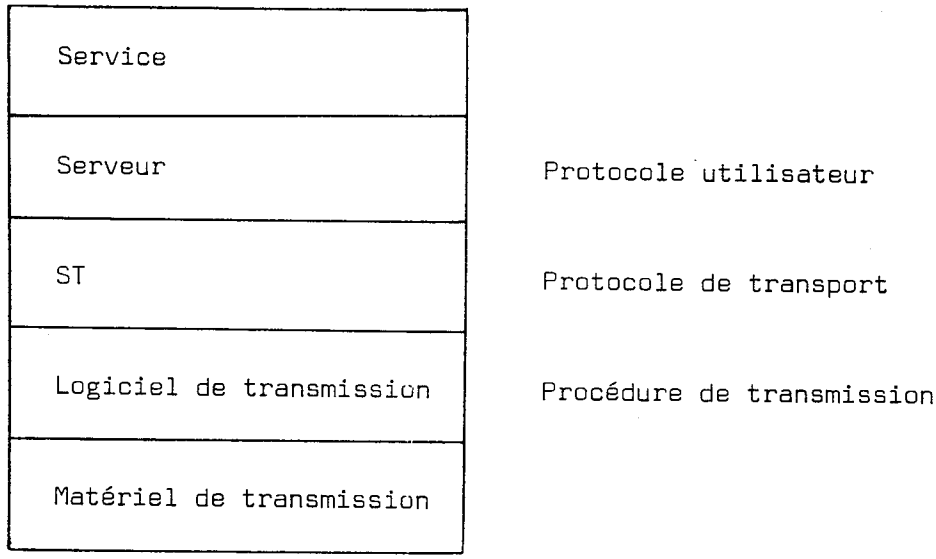
---

* *Définition dans le glossaire*

1. LES PROTOCOLES ET LA FIABILITÉ

1.1. L'empilement des protocoles

Pour mettre un service d'un calculateur participant sur le réseau, il faut réaliser un serveur qui interface le service à offrir avec le service de transport. Le service de transport est représenté par une ST qui utilise le réseau de communication. La ST accède au réseau de communication par un logiciel de transmission qui utilise lui-même le matériel de transmission (ligne, coupleur, modem). On peut représenter l'empilement de ces couches de la façon suivante :



Chacun de ces niveaux utilise un protocole pour dialoguer avec son correspondant.

- Les logiciels de transmission dialoguent suivant une procédure de transmission
- Les ST dialoguent suivant le protocole de transport
- Le serveur et son client dialoguent suivant un protocole utilisateur.

1.2. Stratégie de reprise dans CYCLADES

Ces protocoles sont conçus de sorte qu'ils récupèrent les erreurs du niveau inférieur :

- La procédure de transmission récupère les erreurs dûes au matériel de transmission
- Le protocole de transport récupère les erreurs dûes à l'outil de transmission (réseau de communication)
- Certains protocoles utilisateur tentent de récupérer les erreurs du service de transport.

Cette présentation est très schématique et doit être complétée.

1.2.1. Procédure de transmission

La procédure de transmission récupère effectivement un certain nombre d'erreurs de transmission sur la ligne reliant le calculateur participant au noeud du réseau de communication : messages tronqués, modifiés, perdus, incompris, etc... Mais en cas de problème grave (ligne coupée, modem ou coupleur en panne), elle ne peut rien faire et le signale au niveau supérieur. De plus la procédure peut être mal spécifiée et comporter elle-même des erreurs.

1.2.2. Protocole de transport

Le protocole de transport reprend bien les erreurs dûes au réseau de communication (paquets perdus, dupliqués, désordonnés). Mais il ne peut rien contre une panne du matériel de transmission ou du noeud auquel est connecté le calculateur participant, à moins qu'il y ait deux connexions différentes sur deux noeuds différents.

De plus ce protocole peut lui-même introduire des erreurs ou plus exactement des blocages du service de transport. Des modifications lui ont été apportées pour résoudre certains cas de blocages possibles*.

1.2.3. Protocoles de niveau supérieur

Les protocoles de niveau supérieur (protocoles utilisateur) ne se comportent pas tous de la même façon devant une panne de service de transport.

Le protocole de transfert de fichiers [15] prévoit la cassure du flot utilisé pour le transfert. Au cours du transfert des données, le producteur envoie un message signalant qu'il conserve un point de reprise. Cet échange de messages peut être répété régulièrement et si une cassure se produit, le transfert peut être redémarré à partir du dernier point de reprise accepté.

Le protocole appareil virtuel ne comporte pas de tels messages. Les pannes du service de transport doivent être prévues au niveau des serveurs ou des clients.

Ainsi la répétition périodique des commandes d'allocation de crédits sur un flot en contrôle de flux a été ajoutée. En effet, si le flot ne transporte des lettres que dans un sens (cas d'un lecteur de cartes ou d'une imprimante par exemple), et que la ST émettrice de ces lettres a reçu un crédit nul (CRD-NB=0) dans la dernière commande FL-ACK reçue, elle ne peut plus émettre tant qu'elle ne reçoit pas une nouvelle commande FL-ACK avec CRD-NB≠0. Si cette commande est perdue par le réseau de communication, l'émetteur est bloqué. La commande FL-ACK n'est donc pas envoyée uniquement quand le crédit change, mais périodiquement.

2. L'IMPLÉMENTATION ET LA FIABILITÉ

2.1. Trois axes de recherche d'une meilleure fiabilité

La recherche d'une meilleure fiabilité et d'une plus grande disponibilité d'un système complexe, comme une application réseau, peut se faire suivant trois directions.

1ºLe premier objectif est d'éliminer les pannes dans tous les composants du système.

2ºSi cet objectif ne peut être pleinement atteint, on s'efforce de limiter les conséquences d'une panne dans un des composants. Si l'un des composants se casse, il ne doit pas empêcher le bon fonctionnement des autres.

3ºEnfin, en cas de panne d'un ou plusieurs composants conduisant à l'interruption d'un travail en cours, le système doit pouvoir redémarrer rapidement et reprendre le travail interrompu au point où il en était au moment de la panne.

Dans le cas d'un réseau, le premier objectif ne peut être atteint. Des pannes sont toujours possibles dans le matériel. Les ordinateurs, les modems, les lignes téléphoniques ne sont pas parfaits... Pas plus que les logiciels. Il faut donc également viser les autres objectifs tout en essayant de tendre vers le premier.

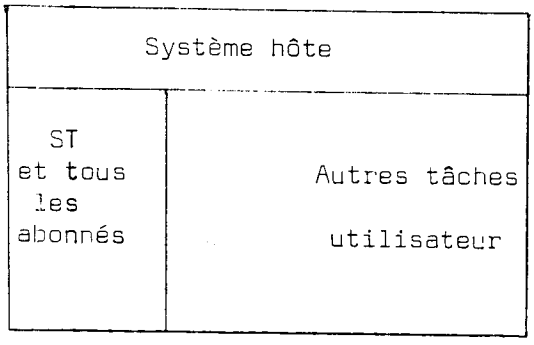
2.2. L'indépendance des fonctions

Un moyen de tendre vers le deuxième objectif est de séparer dans l'implémentation les différentes fonctions utilisées. Dans CYCLADES la fonction de communication est bien séparée du reste du réseau par l'architecture à deux niveaux. De plus le réseau CIGALE peut continuer à travailler en cas de défaillance d'un noeud ou d'une ligne. Le routage adaptatif lui permet de pallier une telle panne.

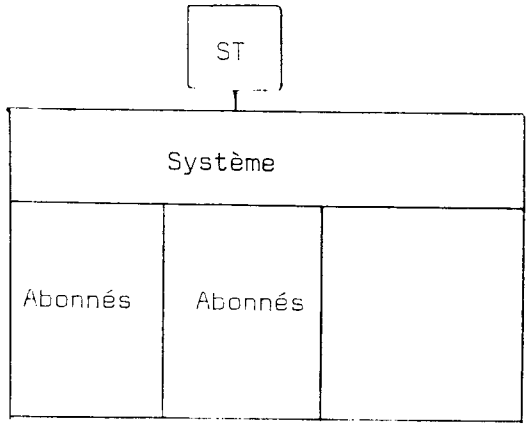
Mais le deuxième niveau n'est pas organisé aussi clairement. Les implémentations réalisées dans les différents calculateurs participants sont très variées [16].

A part les concentrateurs de terminaux, tous les calculateurs participants offrent des services locaux en même temps que les services réseau. Ils ont donc un système d'exploitation sous lequel se trouvent les logiciels réseau (station de transport et abonnés).

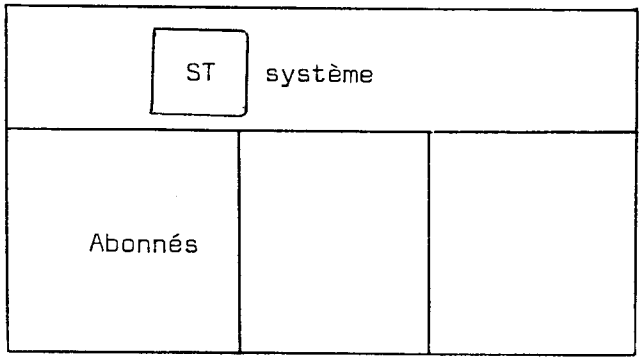
La variété des réalisations actuelles ou prévues peut être montrée par les schémas suivants :



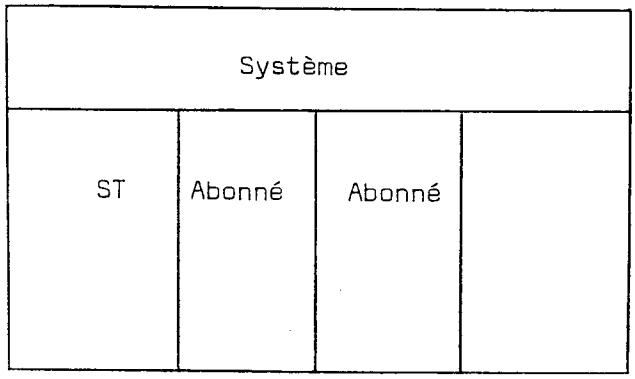
1- La ST et les abonnés se trouvent dans une même tâche utilisateur.



2- Un calculateur frontal héberge la ST. Les abonnés sont des tâches utilisateur (ou une seule). La ST peut être programmée ou micro-programmée.



3- La ST est intégrée au système d'exploitation. Les abonnés sont des tâches utilisateur.



4- La ST et les abonnés sont des tâches utilisateurs. Le système fournit un moyen de communication entre la ST et les tâches abonnés.

Dans le cas où la ST est implémentée dans un calculateur frontal, elle est bien séparée des abonnés. Ainsi une erreur d'un abonné ne peut pas avoir de conséquence pour la ST. De plus l'ordinateur de traitement se trouve déchargé du travail de transport qui ne semble pas être de son ressort. Cependant, cette configuration n'est acceptable que si le moyen de communication entre le frontal et l'ordinateur de traitement offre une bonne fiabilité. On peut envisager cette solution par exemple en présentant le frontal comme un périphérique connecté sur un canal[17]. Une connexion par ligne téléphonique obligerait à refaire au niveau de l'ordinateur de traitement une partie du travail dont on veut le décharger (gestion de la ligne) et introduirait un maillon supplémentaire et peu fiable dans la chaîne.

L'intégration de la ST au système d'exploitation semble une solution logique. On peut ainsi permettre à tout utilisateur d'accéder au réseau par une méthode d'accès standard. Cependant cette solution est coûteuse et délicate. Elle nécessite une très bonne connaissance du système d'exploitation et pose de graves problèmes quand il s'agit d'installer une nouvelle version du système. Comme toute modification importante du système, elle risque d'y introduire des erreurs et donc de faire diminuer la fiabilité. Tous les utilisateurs, y compris ceux qui ne travaillent pas sur le réseau, en subiront les conséquences. Ce type de solution ne peut être adopté qu'au moment où le système est défini, ou, pour un système existant, que si le constructeur prend ces modifications à sa charge ainsi que la maintenance.

La ST peut aussi être implémentée comme une tâche utilisateur, contenant éventuellement certains abonnés, les autres abonnés se trouvant dans d'autres tâches. Dans ce type d'implémentation on réalise bien une indépendance entre les différentes fonctions (ST - Abonné, Abonné - Abonné), mais le problème de la communication entre les tâches se pose. Dans les systèmes ne prévoyant pas ce type de communication, on retrouve les difficultés dûes aux modifications du système, si on veut communiquer par la mémoire.

On peut aussi mettre tout le logiciel réseau (ST et Abonnés) dans une seule tâche utilisateur. Dans ce cas, il est nécessaire d'introduire un niveau supplémentaire de multiprogrammation, les travaux à effectuer en parallèle étant nombreux et variés. Il faut alors tenter de rendre aussi indépendants que possible les différentes fonctions réalisées dans cette tâche. S'il est nécessaire dans ce cas d'utiliser un sous-système de multiprogrammation, il faut noter que dans les cas précédents on utilise également un tel sous-système : la ST est elle-même composée de plusieurs fonctions (gestion de la ligne vers le réseau de communication, réception et analyse des commandes, constitution et expédition des commandes, interface avec les abonnés ou le système).

La solution 1 ne demande pas de matériel particulier ni de services spéciaux du système hôte. C'est sans doute celle qui peut être réalisée le plus aisément, mais elle ne peut présenter une bonne fiabilité que si les fonctions réalisées à l'intérieur de la tâche utilisateur sont clairement séparées (ST- Abonné, Abonné-Abonné) pour que la défaillance de l'une n'entraine pas celle des autres.

Les solutions 2 et 3 sont équivalentes en ce qui concerne la fiabilité à condition que le frontal soit connecté par un moyen fiable (canal) et soit lui-même fiable et que la ST (solution 3) soit intégrée par le fournisseur du système hôte (pour présenter une fiabilité au moins égale à celle du système hôte). Dans ces conditions, si les abonnés sont séparés dans plusieurs tâches, les fonctions sont bien indépendantes. Mais si plusieurs abonnés sont regroupés dans une tâche, on retrouve le problème de leur indépendance.

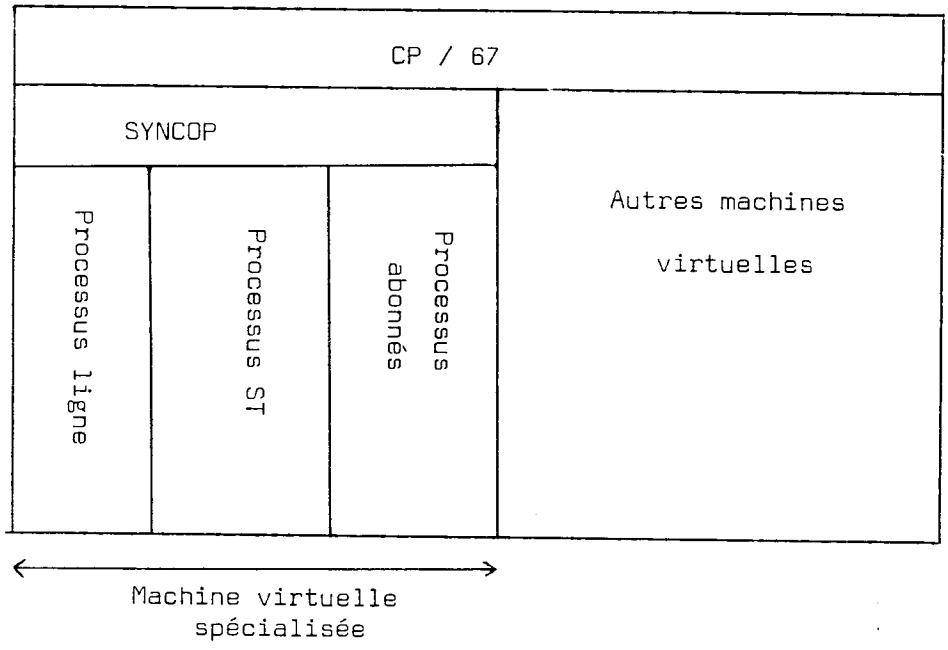
La solution 4 permet de rendre les fonctions indépendantes. A condition que le système hôte fournisse un moyen de communication efficace entre les tâches, c'est la solution qui permet le plus simplement (sans modification du système et sans matériel particulier) d'obtenir une bonne fiabilité, au moins en ce qui concerne la séparation des fonctions.

2.2.1. Les logiciels réseau du 360/67 du C.I.C.G.

Le 360/67 du C.I.C.G. tourne sous deux systèmes d'exploitation alternativement : CP/67 et ASP/OS-MVT. Pour chacun de ces systèmes un logiciel réseau a été réalisé.

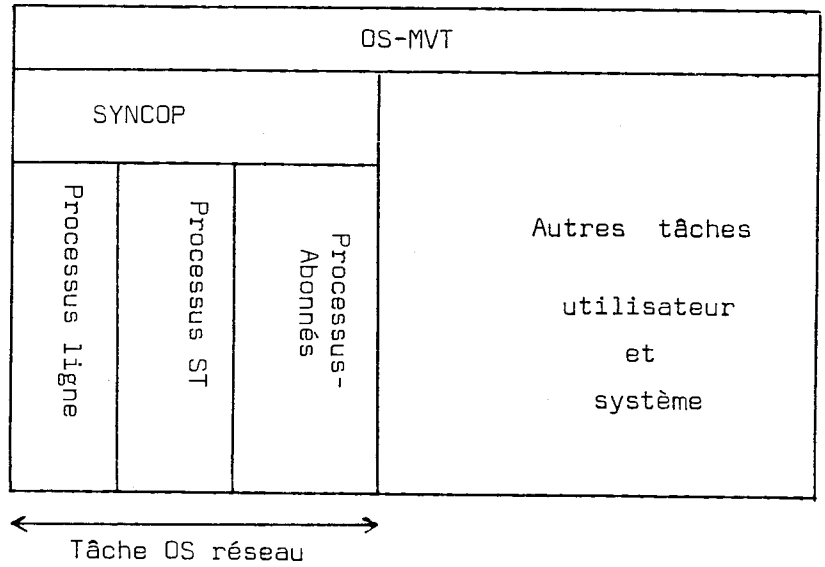
Sous CP/67 tout le logiciel réseau se déroule dans une machine virtuelle spécialisée : gestion de la ligne vers CIGALE, station de transport, abonnés.

Un système de multiprogrammation (SYNCOP) gère les processus qui réalisent ces fonctions.



Le choix de cette implémentation a été dicté par les raisons citées plus haut. On ne disposait pas d'un ordinateur frontal et on ne voulait pas toucher au système CP/67. De plus aucun moyen de communication entre machines virtuelles n'était suffisamment rapide parmi ceux disponibles dans CP/67.

Sous ASP/OS-MVT, l'implémentation est de même type. L'ensemble du logiciel réseau se déroule dans une tâche utilisateur d'OS. Le même système de multiprogrammation (SYNCOP), adapté à OS, gère des processus qui réalisent les mêmes fonctions [11].



2.2.2. Les pannes locales

Ces deux structures tout à fait analogues permettent de classer les causes de pannes locales en quatre catégories.

- erreur du système d'exploitation hôte (ASP/OS - MVT ou CP/67)
- erreur du sous-système de multiprogrammation (SYNCOP)
- erreur d'un processus ST ou du processus de gestion de la ligne
- erreur d'un abonné.

2.2.2.1. Erreur système

Les erreurs du système hôte ou du sous-système ont des conséquences fatales sur l'ensemble. Cependant on peut les séparer :

Le logiciel réseau se présente vis-à-vis du système hôte comme tout autre programme utilisateur. A ce titre, il n'est pas plus favorisé qu'un autre et, comme tous les utilisateurs, il recherche du système hôte la plus grande fiabilité. Nous n'aborderons pas le problème de l'amélioration de la fiabilité du système hôte, qui sort du cadre de cette étude. Quel que soit le type d'implémentation choisi, les erreurs du système hôte ont les mêmes conséquences, même pour la solution du calculateur frontal : la ST est toujours active, mais les services auxquels elle doit donner accès sont indisponibles.
Une erreur du sous-système a les mêmes conséquences, mais ce sous-système est sous le contrôle de l'équipe qui réalise les logiciels réseau. Il doit donc être conçu et réalisé pour présenter une fiabilité maximum. Ce point sera abordé au chapitre suivant.

2.2.2.2. Erreur d'un processus

En cas d'erreur d'un des processus ST ou abonné, il est nécessaire que cette erreur n'en provoque pas d'autres. Ceci impose que le sous-système de multiprogrammation sache analyser l'erreur et faire en sorte que seul le processus fautif soit touché.
Si un processus ST se casse, l'ensemble de la ST est touché : chaque processus est indispensable. Tous les abonnés en subissent les conséquences.
Par contre, si un processus abonné se casse, seul cet abonné voit son travail interrompu.

2.2.3. Les pannes distantes

S'il est important d'isoler les fonctions pour éviter la propagation des pannes, il est aussi important que cet isolement ne soit pas total, de façon qu'une erreur soit connue des fonctions concernées.
Dans une application réseau de type client-serveur, les pannes du serveur (pannes locales) ne sont pas les seules à compromettre l'application. Le

client, sa ST ou le service de communication peuvent se casser. Deux problèmes se posent alors :

- le serveur doit pouvoir détecter ces pannes distantes
- il doit ensuite réagir.

La distinction qui est faite ici entre client et serveur est purement formelle. Vu du client le problème est le même : le serveur, sa ST ou le service de communication peuvent également se casser et le client doit s'en apercevoir et réagir convenablement. Cependant, il n'a pas les mêmes moyens que le serveur : les clients sont en général réalisés sur de petites machines ne disposant pas de mémoire magnétique.

2.2.3.1. Détection des pannes distantes

Les pannes distantes peuvent être signalées par le client lui-même (client au sens large : client + ST), sinon elles doivent être détectées par le serveur seul (serveur + ST). Le serveur, ignorant le comportement du client en cas de panne , doit de toute façon être capable de déceler seul les pannes distantes.Cependant, s'il est prévenu par le client, sa réaction sera plus rapide.

Si le client se casse, sa ST restant active, la ST côté client peut fermer le flot qu'il utilisait. Le serveur est alors prévenu. Mais si le client se casse et que son flot reste actif le serveur doit alors tester lui-même l'état de son correspondant. Pour cela, le serveur peut envoyer périodiquement des messages demandant une réponse. Si après n émissions de ces messages aucune réponse n'est revenue, le client est décrété mort.

Dans une application utilisant le protocole appareil virtuel, le serveur envoie des messages

VOTRE ETAT

et attend la réponse

MON ETAT

Notons que cette solution permet également de détecter les pannes de la ST distante ou du service de communication dans le cas où le serveur n'a pas d'autres messages à émettre. Par exemple :

- sur une connexion simplex récepteur. C'est le cas d'un lecteur de cartes.
- sur une connexion simplex émetteur provisoirement inactive, comme une imprimante quand il n'y a pas de liste à envoyer.
- sur une connexion duplex provisoirement inactive. C'est le cas d'une console ou d'un terminal quand le serveur n'a rien à émettre.

Le maintien d'un trafic minimum sur un flot permet à la ST de s'apercevoir de la panne de la ST distante ou du service de communication.
Pour déceler une panne de la ST distante ou du service de communication, cette méthode est un peu lourde. En effet, cette émission périodique doit être programmée dans tous les abonnés du serveur. De plus, il semble plus logique de confier ce type de contrôle aux stations de transport : le service de transport doit être capable de connaître son état et de le faire savoir aux abonnés.
C'est pourquoi des modifications ont été apportées au protocole de transport. Une fois un flot ouvert en contrôle d'erreur, chacune des deux ST doit envoyer périodiquement des acquittements (commande FL-ACK) de la dernière lettre reçue correctement ou de la lettre fictive de référence 0 si aucune lettre n'a été reçue. Ainsi, si pendant un certain temps (n fois la période d'émission des acquittements) aucun acquittement n'est arrivé sur un flot, la ST décrète que le flot est fermé et prévient l'abonné qui l'exploite. Cette solution a l'avantage d'être programmée une seule fois dans la ST.

L'envoi périodique des acquittements a été ajouté au protocole de transport décrit au chapitre précédent. Le premier protocole de transport ne contenait pas ce mécanisme. On a donc utilisé pour les serveurs les échanges de messages MONT-ETAT, VOTRE-ETAT, ce qui est devenu inutile pour les serveurs utilisant le deuxième protocole de transport.

2.2.3.2. Réaction aux pannes distantes

Une fois la panne distante décelée, le serveur doit réagir. Dans tous les cas, le flot est fermé par la ST. Suivant le type du serveur, le travail

qu'il fait après la fermeture du flot sera différent. Ce travail consiste à sauvegarder l'état dans lequel il se trouve pour pouvoir reprendre convenablement plus tard, c'est-à-dire, quand le client aura pû se reconnecter. La dissymétrie du protocole de connexion interdit au serveur de relancer le client.

En effet, il n'est pas souhaitable que le serveur puisse relancer le client. Le client doit pouvoir se connecter à tout moment au serveur de son choix. Même après une panne , il doit pouvoir changer de serveur. De plus, s'il devait attendre d'être relancé par le serveur avec lequel il travaillait, il risquerait d'être bloqué par une panne durable. Après une panne le serveur attend donc que le client le contacte.
Par contre, le client peut relancer automatiquement le serveur sans risquer de blocage. En cas de fermeture du flot par suite de panne distante, le client peut refaire périodiquement une demande de connexion au serveur avec lequel il travaillait précédemment jusqu'à ce que la connexion soit établie ou que l'utilisateur (humain) arrête ce processus.
L'utilisateur aura été prévenu, au moment de la panne , que le processus de reconnexion automatique est lancé. Il pourra alors l'interrompre à tout moment.

Cette méthode semble particulièrement intéressante pour les connexions qui ne sont pas contrôlées en permanence par un utilisateur (humain), comme la connexion d'une imprimante à un service de traitement par lots.

Tous les serveurs ne se comportent pas de la même façon : le protocole de connexion dit explicitement que le client a l'initiative de la connexion, mais il est muet au sujet de la déconnexion. Certains serveurs ne rompent la connexion qu'en cas de panne, laissant au client l'initiative de la déconnexion. D'autres coupent la connexion si elle reste inactive trop longtemps. Il faut alors que ces serveurs sachent indiquer au client si la déconnexion est normale ou consécutive à une panne. Ceci est possible grâce au paramètre RAISON de la commande FL-TERM.

2.3. Action sur la fréquence des pannes

En ce qui concerne la fréquence des pannes , on n'agira que sur le sous-système de multiprogrammation et les processus ST et abonnés. On ne parviendra pas à la certitude que l'ensemble ne contient pas d'erreur. Cependant, une bonne méthodologie de tests permet de déceler un bon nombre d'erreurs. On ne reviendra pas sur ces tests décrits dans [10] .
Constatons seulement que de bons outils de tests doivent être fournis par le sous-système : trace, vidage mémoire total ou partiel, arrêt sur instruction, consultation et modification des données et des programmes en cours d'exécution, etc ...

2.4. Procédures de reprise

Le rôle des procédures de reprise est de permettre qu'un travail interrompu par une panne puisse être poursuivi correctement quand le système redémarre. Le principe consiste à sauvegarder l'état du système (ou point de reprise) pendant le fonctionnement normal. Cette sauvegarde est déclenchée soit par certains évènements faisant changer l'état du système, soit à intervalles réguliers. Les informations d'état doivent être rangées de façon qu'elles ne soient pas perdues ou altérées par une panne . Après redémarrage du système consécutif à une panne , la dernière sauvegarde sert à remettre le système dans l'état où il était avant la cassure.
De même qu'on a réalisé la détection des pannes distantes au niveau le plus bas, on a essayé d'introduire les procédures de reprise au niveau de la ST. En effet, si la ST seule est capable de réaliser ces procédures de reprise, les abonnés en sont déchargés et le travail de chacun des abonnés est allégé.

2.4.1. Reprises au niveau de la ST

Nous avons limité les procédures de reprise dans la ST aux flots fonctionnant en contrôle d'erreur. Le service de base est par définition un service sans contrôle particulier.

L'abonné qui l'utilise le sait et s'en contente. Il est donc inutile d'essayer de l'améliorer. Par contre, les services additionnels (contrôle d'erreur et contrôle de flux) sont définis comme fiables et les abonnés qui travaillent avec en attendent une fiabilité maximum.

2.4.1.1. Etat de la ST

On définit l'état de la ST comme l'union des états de chacun des flots ouverts en service additionnel.
A un instant donné, l'état d'un tel flot est défini par les informations suivantes :

- identificateur du flot (FL-ID)
- listes des lettres à émettre non encore acquittées
- références de ces lettres (MY-REF)
- références de la dernière lettre reçue (YR-REF)
- liste des lettres reçues non retirées par l'abonné.

De plus, en cas de contrôle de flux :

- la longueur maximum des lettres (MY-LT-LG)
- les crédits en émission et réception (CRD-NB)

On ne s'intéresse qu'aux lettres, pas aux fragments. Le doublement des fragments est prévu par le protocole de transport, on peut donc les émettre et les recevoir deux fois sans risque de confusion. Les télégrammes étant susceptibles de se perdre d'après le protocole de transport, on ne cherche pas à les sauver.

2.4.1.2. Changements d'état

Un flot change d'état chaque fois qu'au moins une de ces informations change, c'est-à-dire quand :

Un abonné local
- ouvre ou ferme un flot
- demande à émettre une lettre ou en retire une.

La ST reçoit
- une commande FL-TERM
- une commande FL-LT ou FL-ACK acquittant une nouvelle lettre ou faisant varier le nombre de crédits en émission

La ST envoie - une commande FL-LT ou FL-ACK acquittant une nouvelle lettre ou faisant varier le nombre de crédits en réception.

Ces changements d'état se produisent donc très souvent. Il semble très couteux de les enregistrer à chaque fois qu'ils se présentent. On peut essayer de faire les sauvegardes d'état moins fréquemment.
Les ouvertures et fermetures de flot locales ou distantes doivent être systématiquement enregistrées, sinon on risquerait de redémarrer sans un flot qui devrait exister ou avec un flot qui devrait avoir disparu, ce qui perturberait l'abonné qui l'exploite. Parmi les évènements cités les ouvertures et fermetures de flot sont les moins fréquents.
Toutes les demandes d'émission et les retraits de lettre de la part de l'abonné doivent également être enregistrées, sinon, après redémarrage, des lettres risqueraient d'être perdues (émission) ou remises en double (réception), ce qui est contraire au principe du contrôle d'erreur.
Le nombre de crédits en réception doit aussi être maintenu constamment. Il risquerait autrement de ne pas correspondre au degré d'anticipation voulu par l'abonné.
La référence de la dernière lettre acquittée (YR-REF) doit toujours être à jour. Si au moment de la cassure YR-REF vaut n et que la dernière valeur sauvegardée est n-p, après le redémarrage on va recevoir la lettre de référence n+1 et on demandera la réémission de la lettre de référence n-p+1. La ST correspondante ne pourra pas la réémettre puisque toutes les lettres jusqu'à la référence n ont été acquittées et sont donc perdues par l'émetteur.
Seuls le nombre de crédits en émission et la référence de la dernière lettre émise et acquittée peuvent ne pas être maintenus constamment. En effet, ces informations sont reçues périodiquement de l'autre ST dans les commandes FL-ACK. Après redémarrage on les récupèrera. Ceci suppose toutefois que les deux ST ne se cassent pas en même temps.
Si on veut qu'une cassure ne soit pas sensible pour l'abonné, on a donc un nombre important d'informations à sauver et une fréquence de mise à jour de ces informations très élevée.

2.4.1.3. Panne

En cas de panne distante (n émissions de la même commande FL-LT sans réponse ou pas de réception de commande FL-ACK depuis longtemps), le protocole de transport prévoit que le flot est fermé localement. Si la ST distante se casse, ayant sauvegardé son état comme on vient de le voir, en redémarrant elle voudra continuer à travailler sur un flot qui est fermé pour l'autre ST, ce qui aboutira à la fermeture complète du flot. On n'aura rien gagné.

Une première solution à ce problème est de modifier le protocole pour éviter la fermeture locale d'un flot sur une panne distante. Le flot, restant dans son état au moment de la panne , pourrait alors fonctionner normalement après la panne. Ceci présente un inconvénient : une panne durable pourrait amener la ST à maintenir des flots inutiles. De plus cette solution ne conviendrait pas aux applications de type client-serveur(cf.2232). Une autre solution consisterait, pour la ST qui n'est pas directement touchée par la panne, à sauvegarder l'état du flot dès que la panne est décelée puis à fermer le flot comme le prévoit le protocole. Dès que l'activité peut reprendre, la ST qui redémarre se retrouve dans l'état précédant la panne et envoie une commande sur ce flot. La ST qui reçoit cette commande, ne trouvant pas ce flot actif, ira chercher son état sauvegardé et pourra le réactiver. Cette solution présente l'inconvénient d'obliger les deux ST à prévoir la possibilité de sauvegarder leur état, ce qui nécessiterait, sur les petites machines servant de concentrateurs de terminaux par exemple, l'installation de mémoire secondaire.

2.4.2. Reprises au niveau abonné

L'introduction de procédures de reprises au niveau de la ST est très couteuse et ne résoud que partiellement le problème. Si on se place au niveau des clients ou des serveurs, la solution proposée permet de pallier les pannes de la ST et/ou du réseau de communication. La seule conséquence pour les abonnés est l'attente de la réparation, leurs travaux n'étant pas compromis.

Mais en cas de panne provoquant la mort d'un abonné ou de son correspondant (erreur du système hôte, du sous-système, d'un processus abonné), cette solution est insuffisante.
La ST ne connait que le mode de transport des informations de l'abonné, elle ignore leur signification et l'utilisation qu'en fait l'abonné. L'abonné seul peut savoir où en est son travail et comment il doit redémarrer en cas de panne .
Il est donc nécessaire de prévoir des procédures de reprise au niveau des abonnés. Ces procédures peuvent être définies soit au niveau des protocoles utilisateur utilisés entre abonnés (c'est le cas du protocole de transfert de fichiers par exemple), soit au niveau de l'implémentation des abonnés.
Si la procédure de reprise est définie par le protocole, elle est réalisée par l'ensemble des abonnés qui coopèrent en utilisant ce protocole. L'implémentation du protocole entraine l'implémentation de la procédure de reprise définie.
Si le protocole ne spécifie pas de procédure de reprise (PAV par exemple), un seul abonné d'une application peut réaliser les reprises sur panne. La stratégie de reprise est alors laissée au choix de l'implémenteur de cet abonné (cf chap.IV).
Les procédures de reprise des abonnés sont différentes pour chaque type d'abonné. Il serait vain d'essayer d'en donner une description générale.
Si les abonnés ont leurs propres procédures de reprise, celles de la ST ont alors peu d'intérêt. En effet les abonnés sauvegardent les informations de reprise d'une façon plus synthétique et à moindre frais. Dans une application de transfert de fichier par exemple, l'émetteur reprend le transfert après une panne en réémettant le dernier enregistrement que les deux correspondants ont accepté comme point de reprise. Dans ce cas la seule information à sauvegarder pour l'émetteur est l'adresse de cet enregistrement. Les sauvegardes que la ST pourrait faire sont inutiles.
Nous n'avons donc pas réalisé la procédure de reprise décrite plus haut.

3. LOCALISATION DES PANNES

Les différents logiciels peuvent s'apercevoir des pannes, comme nous l'avons vu. Mais l'utilisateur, s'il se rend compte de l'impossibilité de travailler en cas de panne, a quelques difficultés à savoir quel maillon de la chaîne le reliant au service est défectueux. Il est important qu'il dispose d'un moyen de connaître l'état de ces maillons pour savoir s'il peut reprendre son travail rapidement ou s'il doit changer de moyen d'accès au service.

Le protocole de transport et le réseau de communication fournissent un moyen de tester l'état des noeuds du réseau et des stations de transport: l'écho.

Chaque station de transport peut émettre une commande ECHO. Un noeud recevant cette commande doit la renvoyer à la station de transport émettrice. Chaque station de transport possède une 'porte ECHO'. Toutes les lettres ou télégrammes reçus sur cette porte sont renvoyés à l'expéditeur. En envoyant des échos sur les noeuds et les stations de transport, on peut donc connaître ceux qui ne répondent pas, et localiser le maillon défaillant.

4. RETOUR SUR LA DÉTECTION DES PANNES

Dans une application réseau, toute panne affectant un des composants mis en jeu peut être détectée. Deux types de mécanismes sont utilisés :

1- Un mécanisme par échange de messages entre correspondants : interrogation pour test et rapport d'anomalie.

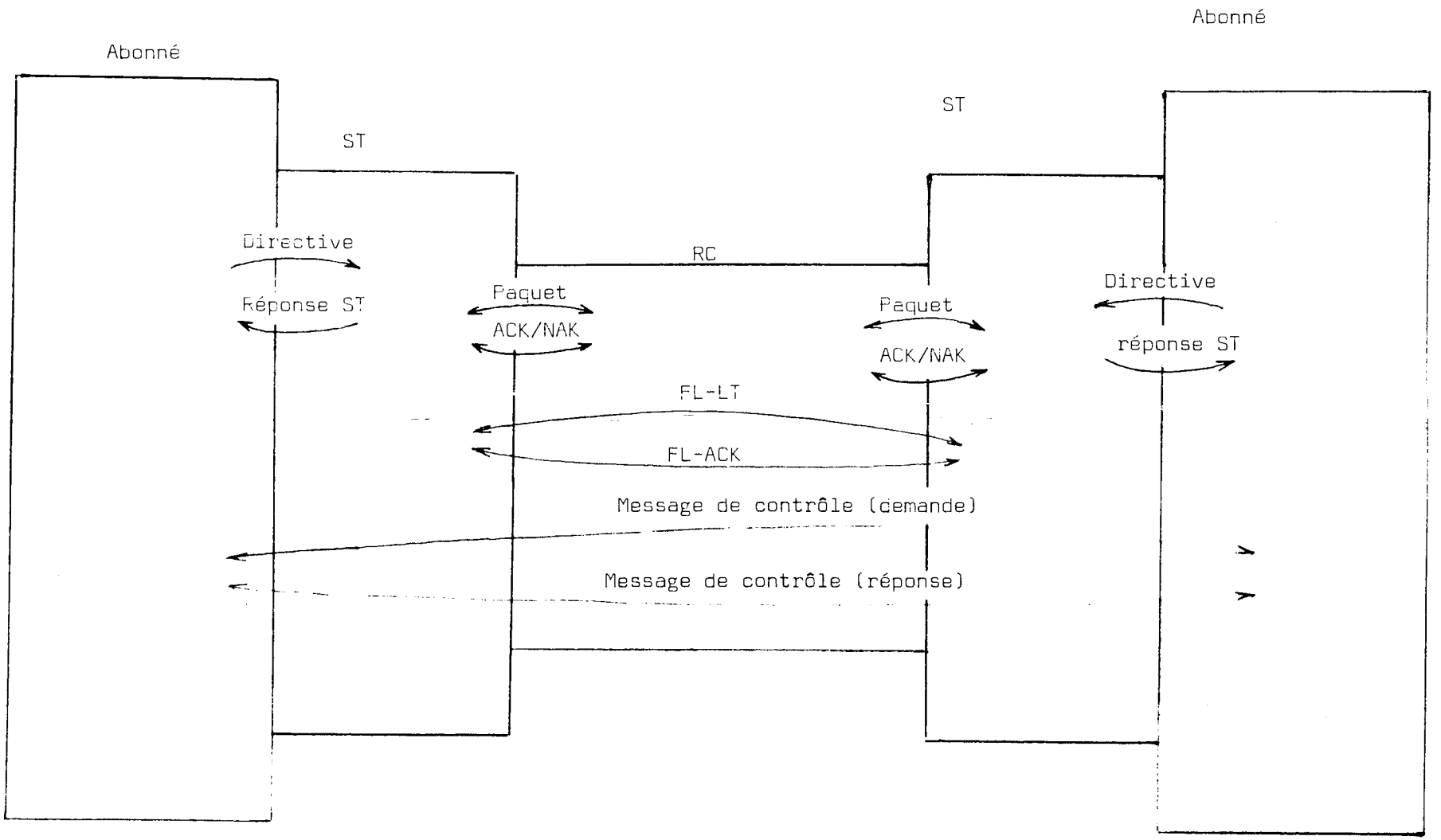
2- Un mécanisme de 'time out'.

On trouve le premier mécanisme entre abonnés, entre deux ST pour un flot, entre ST et réseau de communication (RC), entre ST et abonnés locaux.

- Entre abonnés : les abonnés échangent entre eux des messages de contrôle (lettres ou télégrammes) permettant de demander l'état du correspondant et de répondre à cette demande.
- Entre ST : pour chaque flot, les ST envoient des commandes FL-LT et reçoivent des commandes FL-ACK indiquant si les lettres sont bien reçues ou non.
- Entre ST et RC : la procédure de ligne permet d'envoyer des paquets vers le réseau de communication, le RC (plus exactement le noeud auquel est connecté la ST) répond si ces paquets sont bien reçus par des messages de contrôle de la procédure (ACK, NAK).
- Entre ST et abonnés : les abonnés remettent des directives de transport à la ST qui répond pour indiquer la prise en compte de ces directives.

Ces mécanismes se retrouvent dans toutes les implémentations et sont pour la plupart imposés par les protocoles. L'ECHO constitue une possibilité supplémentaire qui permet aux abonnés de tester les ST et les noeuds du RC.

La figure suivante résume l'ensemble de ces mécanismes. On voit que ces dialogues sont symétriques (sauf entre ST et abonnés) et qu'à chaque niveau on peut tester le correspondant et les niveaux inférieurs.



Détection des pannes par échange de messages

Le mécanisme précédent est insuffisant pour la détection de toute panne. En effet, ces échanges peuvent être compromis par l'indisponibilité d'un des composants utilisés. Dans ce cas, pour éviter l'attente indéfinie d'une réponse, l'émetteur arme un réveil quand il envoie sa demande. Quand la réponse arrive, il supprime ce réveil. Si le réveil sonne, c'est que la réponse n'est pas arrivée dans les délais prévus et la demande est réémise. Après quelques réémissions sans réponse, la communication est considérée comme rompue.

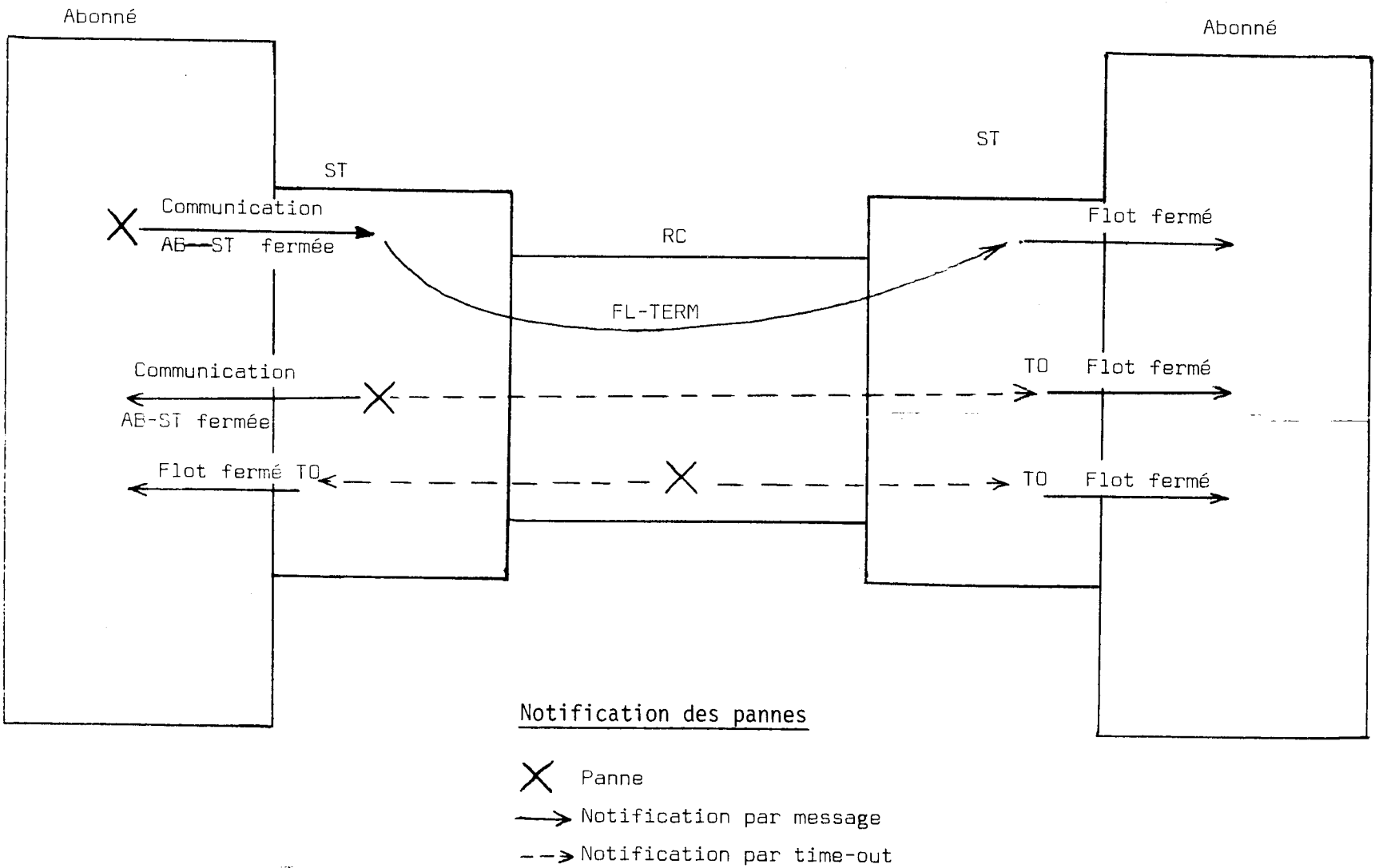
La panne étant détectée, il faut encore que chaque élément concerné soit prévenu. Ceci est obtenu soit par envoi de messages, soit, quand ces messages ne peuvent être envoyés, par time-out (TO).

- Une erreur d'un abonné entraine la fermeture de la communication entre l'abonné et sa ST. La ST ferme alors le(s) flot(s) utilisé(s) par l'abonné (envoi de FL-TERM). Les ST distantes sont donc prévenues et informent les abonnés correspondants de la fermeture du flot.

- Une panne d'une ST provoque la fermeture des communications entre la ST et ses abonnés locaux. La ST en panne n'émettant plus, les ST distantes s'en aperçoivent grâce aux time-out et ferment les flots concernés en prévenant les abonnés.

- Une panne du réseau de communication interrompant les échanges entre ST, provoque la fermeture des flots dans ces ST, qui préviennent les abonnés concernés.

Le schéma suivant résume les mécanismes de notification des pannes. On voit donc que toute panne peut être décelée et que les correspondants concernés par une panne sont prévenus. Notons cependant qu'en cas d'erreur de la ST ou d'un abonné, les protocoles ne précisent pas les moyens de prévenir le correspondant local (abonné en ST respectivement). L'outil de communication abonné$\longleftrightarrow$ST doit être prévu pour déceler l'erreur d'un correspondant et pour prévenir l'autre.


Abonné
Abonné
ST
ST
Communication
AB—ST fermée
RC
Flot fermé
FL-TERM
Communication
AB-ST fermée
TO
Flot fermé
Flot fermé TO
TO
Flot fermé
Notification des pannes
Panne
Notification par message
Notification par time-out

5. Conclusion

Cette étude nous a permis de dégager les points importants sur lesquels doivent se porter les efforts pour obtenir une meilleure fiabilité et un meilleur service.

La connexion du 360/67 au réseau de communication par deux lignes sur deux noeuds différents n'a pû être réalisée faute de matériel (pas de position disponible sur l'unité de transmission du 360/67, pas de deuxième noeud proche).

Nous avons étudié et réalisé un sous-système de multiprogrammation qui possède les caractéristiques citées plus haut. Nous avons défini et implémenté les procédures de reprise des abonnés réalisant les services offerts sur le 360/67. Ces deux points font l'objet des chapitres suivants.

CHAPITRE III

LE SYSTÈME SYNCOP SOUS CP/67

1. LES (SOUS)-SYSTÈMES RÉSEAU

Plusieurs (sous)-systèmes sont (ont été) utilisés pour implémenter des logiciels réseau. Citons, sur IBM/360, ASP pour ARPANET et HASP pour SOC.

Au cours de l'évolution de CYCLADES, on a utilisé plusieurs (sous)-systèmes :

- CRIC [8] développé sur CII 10070, a été ensuite transporté sur IRIS 80, IRIS 45, SIEMENS 4004 et IBM 360.
- TELCOM, écrit initialement sous CP/67 [9] pour réaliser un concentrateur de terminaux lourds a été ensuite modifié pour implémenter la ST 1 [10] puis adapté à OS-MVT [11].

Ces deux (sous)-systèmes ont été utilisés pour l'implémentation de ST 1 et des abonnés sur ces machines.

Profitant de l'expérience acquise avec CRIC et TELCOM, le centre scientifique CII et l'équipe réseaux de l'ENSIMAG ont défini en commun le système SYNCOP (SYstème Normalisé de COmmutation de Processus).

SYNCOP est implémenté sur plusieurs machines :

- IRIS 80 sous SIRIS 8 [14]
- IBM 360 sur machine nue, sous CP/67 (machine virtuelle nue) et OS-MVT [12]
- MITRA 15 sur machine nue
- ORDOPROCESSEUR sur machine nue.

Présentant les mêmes services sur toutes ces machines, SYNCOP permet un transport plus facile des logiciels réseau d'une machine à l'autre. SYNCOP est utilisé pour réaliser la ST2 sur toutes ces machines ainsi que les logiciels utilisant la ST2.

Suivant les implémentations, SYNCOP se présente comme un sous-système

travaillant dans une tâche sous le contrôle du système hôte (SIRIS8 ou OS/MVT) ou comme un système travaillant sur machine nue (CP/67, MITRA 15, ORDOPROCESSEUR).

Sur les grosses machines, l'utilisation d'un sous-système se justifie par la lourdeur des systèmes hôtes :
- le temps de commutation des tâches est trop important
- les services offerts sont trop coûteux.

Ces inconvénients sont dûs en grande partie aux protections du système hôte contre les tâches qu'il contrôle. Le sous-système ne nécessite pas des protections aussi poussées : les processus ne sont pas aussi variés que les tâches du système hôte.

2. CARACTÉRISTIQUES D'UN (SOUS)-SYSTÈME

Le rôle d'un (sous)-système est de gérer un ensemble de processus qui réalisent les services réseau sur un calculateur participant (service de transport, applications).
Les processus coopèrent pour réaliser ces services. Ils doivent donc pouvoir se synchroniser et communiquer entre eux. Ils doivent se partager la mémoire . Un service de réveil doit être offert : les procédures de transmission et protocoles de transport prévoient des actions différées dans le temps. Le (sous)-système doit également gérer les unités d'entrée-sortie ou au moins permettre aux processus de faire des entrées-sorties.
Ces quelques caractéristiques donnent les services nécessaires dans un (sous)-système de multiprogrammation pour des applications réseau. D'autres services peuvent être utiles, notamment pour obtenir une bonne fiabilité : outils de test, traitement des erreurs programme, suppression dynamique de processus.
Bien qu'offrant de nombreux services ce (sous)-système doit être simple : il n'est que la base sur laquelle sont construits les logiciels réseau.

Il ne doit pas être trop volumineux et doit permettre une programmation facile des processus.

3. PRÉSENTATION DE SYNCOP

On retrouve dans SYNCOP toutes les caractéristiques citées plus haut. Les services qu'il offre concernent la création et la destruction des processus, leur synchronisation (par ECB), la gestion de la mémoire, les réveils, les communications par liste, les ressources.

Les primitives sont :

- CREATE : création d'un processus
- KILL : destruction d'un processus par un autre
- EXIT : suicide d'un processus
- STATUS : demande d'état d'un processus
- WAIT : attente d'un évènement unique (attente simple)
- WAITM : attente de n évènements parmi p ($1 \leq n \leq p$)(attente multiple)
- POST : réalisation d'un évènement
- CHECK : test des évènements réalisés
- GTMN : acquisition de mémoire
- FRMN : libération de mémoire
- STIMER : armement d'un réveil
- RTIMER : désarmement d'un réveil
- CRELST : création d'une liste
- FREELST : libération d'une liste
- PUTLST : enfilement d'un élément dans une liste
- GETLST : retrait d'un élément d'une liste
- LOOKLST : recherche d'un élément dans une liste
- ENQ : réservation d'une ressource
- DEQ : libération d'une ressource.

4. L'IMPLÉMENTATION DE SYNCOP SOUS CP/67

La première version de SYNCOP sur IBM 360 a été écrite sous CP/67, pour tourner dans une machine virtuelle. Cette même version tourne également sur machine réelle nue.
Elle se présente comme une refonte et une extension de TELCOM [9].
Certains principes de TELCOM ont été repris, d'autres ont été remis en cause.
La réalisation de ce système a été guidée par un souci de simplicité et de clarté. Le système se présente comme un ensemble de modules assurant chacun une fonction bien distincte : à chaque primitive correspond un module.

Les principales différences par rapport à TELCOM viennent des options :
- simplification de la programmation des processus utilisant SYNCOP
- création et suppression dynamiques des processus
- développement d'outils de test et aide à la mise au point

Ces trois options vont dans le sens d'une meilleure fiabilité. Cette refonte a été mise à profit pour augmenter les performances du système.

Pour simplifier la programmation des processus tournant sous SYNCOP, l'appel des primitives se fait par macro-instructions [13]. A chaque primitive correspond une macro-instruction spécifiant explicitement tous les paramètres d'appel.
On évite ainsi l'inconvénient de TELCOM dans lequel il faut,avant d'appeler une primitive, charger les paramètres dans des registres spécifiés.

Une autre simplification de la programmation est apportée dans le cas des retours négatifs. Il est possible qu'une demande adressée à SYNCOP ne puisse être satisfaite immédiatement, faute de ressources disponibles. C'est le cas pour les primitives CREATE, GTMN, STIMER, CRELST, PULTST, GETLST, LOOKLST, ENQ.

S'il ne reste pas suffisamment de mémoire libre, une demande de mémoire (GTMN) ne peut pas être satisfaite, de même que les primitives qui crèent des blocs de contrôle : CREATE, STIMER, CRELST, ENQ.
Dans le cas de ENQ, la ressource demandée peut être occupée.
Suivant l'état de la liste concernée, les demandes concernant les listes peuvent ne pas être satisfaites immédiatement :
- PUTLST quand la liste est pleine
- GETLST et LOOKLST quand la liste est vide
- LOOKLST quand l'élément recherché n'existe pas dans la liste.

Dans tous les cas, le processus demandeur est prévenu que sa demande n'est pas satisfaite et le système lui donne la possibilité d'attendre (en lui fournissant un ECB) pour réitérer sa demande.*
Pour éviter au retour de chacune de ces primitives, de tester la valeur de retour, appeler WAIT et refaire la demande, les processus ont la possibilité (grâce à l'indicateur PRETNEG du PCB) de demander que le retour ne soit fait que quand la demande est satisfaite. L'attente et la seconde demande sont faites par le sous-programme qui réalise la primitive.
Les processus ont toujours la possibilité de traiter eux-mêmes les retours négatifs dans le cas où ils ont autre chose à faire en cas de non satisfaction de la demande.
Ceci évite la programmation fastidieuse des tests au retour des primitives et surtout limite le risque d'omettre le traitement des cas où la demande n'est pas satisfaite, ce qui est la source de nombreux incidents.

---

* Si le processus ne veut pas utiliser l'ECB, il doit le remettre à zéro.

5. LES PROCESSUS DYNAMIQUES

On a vu que quand un processus provoque une erreur, il faut pouvoir le supprimer, dans la mesure où il n'est pas indispensable au bon fonctionnement de l'ensemble. De plus, comme on ne connaît pas a priori le nombre de correspondants qui travailleront à travers le réseau avec l'application, le nombre de processus utiles est très variable. Il est donc nécessaire de pouvoir les créer et les supprimer suivant les besoins.
La suppression d'un processus peut s'opérer de deux façons :
- le processus se supprime lui-même (EXIT)
- le processus est supprimé par un autre processus ou le système (KILL).

La suppression d'un processus suppose la libération des ressources qu'il mobilise :
- mémoire
- réveils
- listes
- ressources

Si le processus se supprime lui-même, on peut envisager qu'il libère ces ressources avant de se tuer. Ceci simplifie la tâche de la primitive EXIT mais complique la tâche du processus qui doit connaître et libérer une à une toutes ses ressources. Cependant les autres primitives se trouvent simplifiées du fait qu'elles n'ont pas besoin d'entretenir les informations décrivant les ressources occupées par chaque processus.
Si le processus est détruit de l'extérieur il faut alors que la primitive KILL, appelée par le tueur, soit capable de libérer toutes les ressources mobilisées par le tué.
L'intérêt de la primitive KILL est double :
- un processus peut être tué de l'extérieur et notamment par le système en cas de défaillance du processus.
- l'existence de KILL simplifie le travail des processus qui appellent EXIT, puisque EXIT peut libérer les ressources comme le fait KILL. Et on est alors sûr que toutes les ressources sont bien libérées à la mort d'un processus.

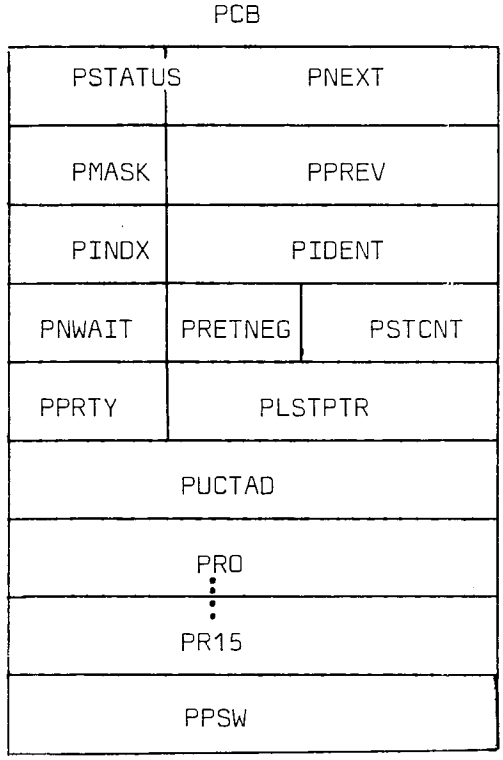
TELCOM ne prévoyait pas la suppression d'un processus (ni KILL, ni EXIT). Ceci s'est révélé gênant dans l'implémentation du serveur sous OS-MVT.

En effet tous les processus qui pouvaient être nécessaires au cours d'une session devaient être créés à l'initialisation du système et étaient présents durant toute la session. Ceci introduisait une charge non négligeable au niveau du distributeur et le contexte de tous ces processus dormants occupait inutilement la mémoire. On a donc introduit dans TELCOM les primitives CREATE et EXIT en laissant au processus qui se supprime le soin de libérer toutes ses ressources. On n'a pas introduit KILL ni la libération automatique des ressources par EXIT qui supposaient une refonte complète du système ainsi que des processus qui l'utilisaient.
L'existence de KILL implique en effet qu'à chaque instant toutes les ressources occupées par chaque processus soient connues du système et donc que chaque primitive qui attribue une nouvelle ressource le signale au système. TELCOM possède donc les primitives CREATE et EXIT et laisse aux processus le soin de libérer leurs ressources avant de se suicider. SYNCOP possède les primitives CREATE, KILL et EXIT et les ressources des processus sont libérées par KILL et EXIT.

5.1. Le PCB

Chaque processus vivant est décrit par un bloc de contrôle, le PCB. Ce bloc décrit l'état du processus et permet de sauvegarder son contexte quand il perd le contrôle. Les informations qu'il contient sont les suivantes :

PSTATUS : état du processus (prêt, en attente, interrompu, actif)
PNEXT : chaînage avant des PCB
PPREV : chaînage arrière
PMASK : compteur de masque
PINDX : index du processus
PIDENT : identificateur du processus
PNWAIT : nombre d'évènements attendus
PRETNEG : indicateur pour retour négatif
PSTCNT : compteur de débordement de pile
PPRTY : priorité du processus
PLSTPTR : adresse de la première liste du processus
PUCTAD : adresse de l'UCT du processus
PR0...PR15 : zone de sauvegarde des registres généraux
PPSW : zone de sauvegarde du mot d'état programme (PSW)



Les PCB sont acquis dans un pool dont la taille est fixée à la génération du système. Ce pool est géré grâce à une table qui contient autant d'entrées que le pool contient de blocs. Ces entrées indiquent si les blocs correspondants sont libres ou non. Un PCB peut donc être identifié par son rang dans le pool (PINDX) ou son adresse. Mais ces informations sont insuffisantes pour identifier sans ambiguité un processus. Il n'y a pas en effet de correspondance biunivoque entre un bloc du pool de PCB et un processus. Un processus vivant peut avoir pour PCB le même bloc du pool qu'un processus mort. Une fois le processus mort le bloc qu'il utilisait comme PCB est libre et un nouveau processus peut l'avoir obtenu.

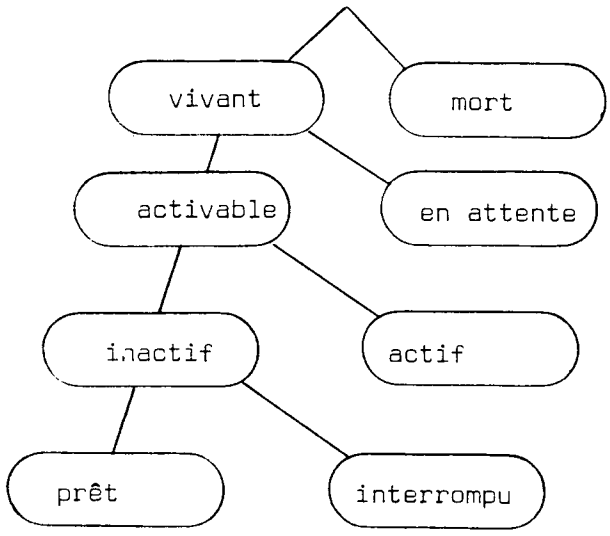
Une autre information est donc associée au processus, son identificateur (PIDENT). Cet identificateur identifie exactement un processus qu'il soit mort ou vivant. C'est un numéro qui est affecté au processus lors de sa création (CREATE). Le premier (dans le temps) processus créé porte le numéro 1, le suivant le numéro 2, etc...

Chaque fois qu'on désigne un processus (pour le tuer : KILL, ou pour demander son état : STATUS) on fournit le couple PINDX-PIDENT. Le premier permet de trouver rapidement l'adresse de son PCB, le second permet de savoir si le PCB trouvé correspond au processus cherché ou si ce processus est mort. Lors de la création d'un processus ce couple PINDX-PIDENT est fourni par CREATE au processus créateur, ce qui lui permettra par la suite de le tuer ou de demander son état.

Les PCB sont acquis dans un pool, et non dans la mémoire libre, pour améliorer les performances du système. En effet chaque fois que le distributeur prend le contrôle il balaie les PCB. Si ceux-ci sont regroupés, il y aura moins de fautes de page que s'ils sont dispersés à travers toute la mémoire [10].

5.2. Les états d'un processus

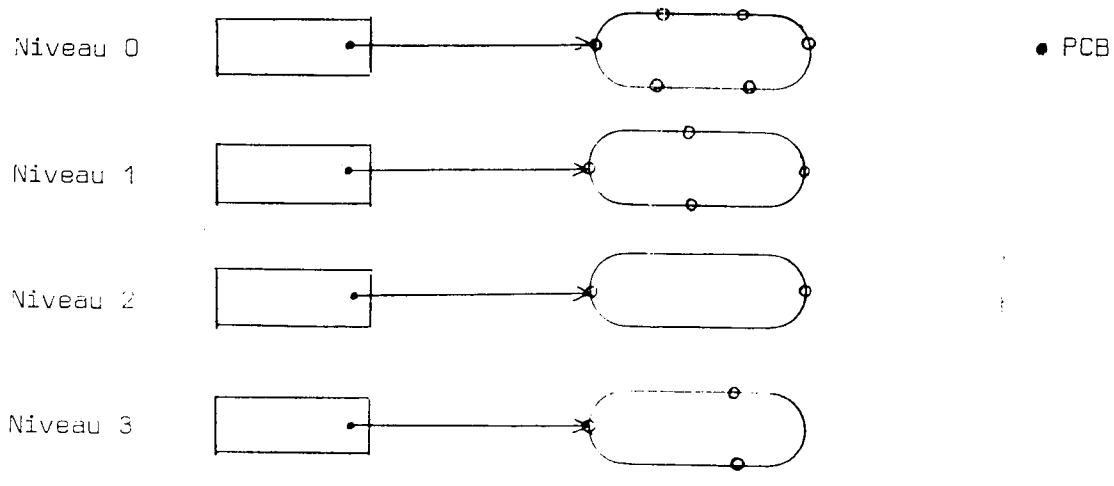
L'arborescence suivante décrit les états d'un processus.



Un processus est vivant dès qu'il a été créé (CREATE) et jusqu'à ce qu'il soit détruit (KILL ou EXIT). Avant sa création et après sa destruction, il est mort. Un processus vivant est en attente quand il demande explicitement de ne plus avoir le contrôle (WAIT ou WAITM) et jusqu'à ce que sa condition de réactivation soit réalisée (POST). Sinon il est activable, c'est-à-dire qu'il demande le contrôle.
Un processus activable est actif quand le distributeur lui a attribué le contrôle. Sinon il est inactif. Un processus inactif est dans l'état interrompu si une interruption est survenue pendant son activité. Sinon, il est prêt.

5.3. Le distributeur et les changements d'état des processus

Le distributeur (qui distribue le temps d'unité centrale entre les différents processus) possède quatre niveaux de priorité. Pour chaque niveau il possède un pointeur qui contient l'adresse du PCB du dernier processus du niveau qui a eu le contrôle, ou zéro si le niveau est vide. Sur chaque niveau les PCB forment un anneau et sont chaînés dans les deux sens (pour simplifier la suppression d'un PCB par KILL ou EXIT).



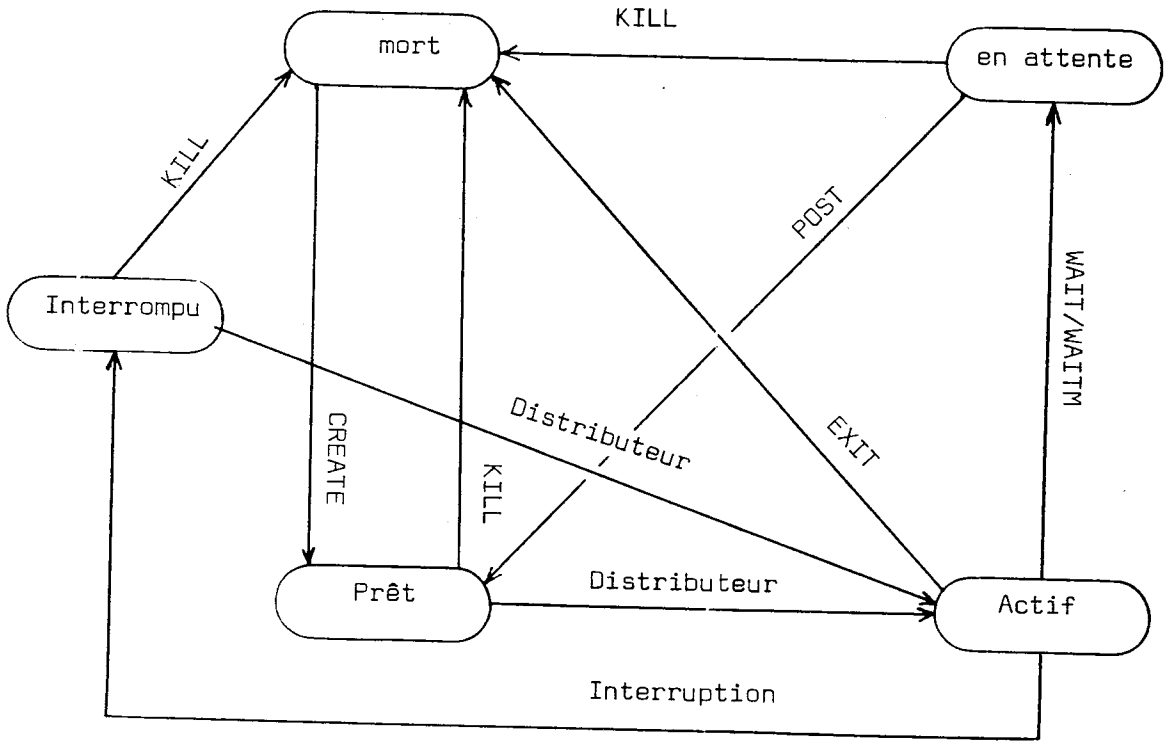
Le distributeur est sans préemption : un processus actif ne perd le contrôle que quand il le demande (WAIT) ou sur interruption, mais dans ce cas il reprend le contrôle dès que l'interruption est traitée.

Le risque d'un distributeur sans préemption est qu'un processus déroule beaucoup de code sans se mettre en attente. Les autres processus sont alors pénalisés. Dans le cas de SYNCOP les processus sont connus et déroulent tous peu de code entre deux attentes. L'absence de préemption n'est donc pas gênante. De plus elle simplifie l'utilisation de ressources non partageables : entre deux attentes le processus est sûr qu'aucun autre processus n'utilisera une ressource qu'il manipule, il n'a donc pas besoin de la protéger. Seules devront être protégées les ressources non partageables dont l'utilisation comporte au moins une attente et celles qui sont également utilisées par les routines de traitement d'interruption. Pour protéger ces dernières on peut masquer l'interruption pendant le temps ou on les mobilise, l'interruption est alors prise en compte et traitée quand la ressource est libre.

Le distributeur utilise l'indicateur PSTATUS des PCB pour activer un processus. Il active le premier processus 'prêt' ou 'interrompu' qu'il rencontre en balayant, dans l'ordre des priorités, les anneaux de PCB. Pour chaque anneau, le balayage commence par le premier PCB qui suit le dernier activé dans l'anneau. Le distributeur dispose d'un autre pointeur qui contient l'adresse du PCB du dernier processus activé. Avant de commencer le balayage, il examine l'état de ce processus. S'il est 'interrompu', c'est ce processus qu'il activera.
Le processus activé est marqué 'actif'. Si une interruption arrive alors qu'un processus est 'actif', la routine de traitement d'interruption le met dans l'état 'interrompu'. Il reprendra ainsi le contrôle immédiatement après le traitement de l'interruption.
Si un processus 'actif' appelle les primitives WAIT ou WAITM, il passe dans l'état 'en attente'. Quand le (ou les) évènement(s) qu'il attend seront réalisé(s) (POST), il repassera dans l'état 'prêt' et pourra alors être activé.
L'activation d'un processus se fait en rechargeant les 16 registres généraux

et le PSW de la machine à partir des valeurs sauvées dans le PCB (PR0,..., PR15, PPSW) au moment où il a perdu le contrôle (WAIT ou interruption).

Les changements d'état d'un processus sont indiqués par le diagramme suivant :



5.4. La création d'un processus

La création d'un processus (CREATE) est une opération simple. Elle consiste à acquérir un bloc PCB dans le pool et une zone de sauvegarde (pile) dans la mémoire libre (cf 14.3), à initialiser ce PCB (état 'prêt') et à l'insérer dans l'anneau de PCB correspondant à sa priorité.

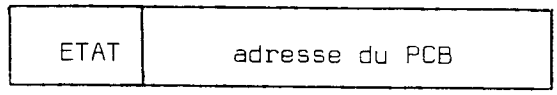
5.5. La suppression d'un processus

KILL et EXIT suppriment un processus en libérant toutes ses ressources, en déchaînant le PCB de son anneau et en rendant le bloc PCB au pool. La libération des ressources est détaillée dans la suite.

6. LA SYNCHRONISATION DES PROCESSUS

6.1. L'ECB

Les processus sont synchronisés par évènement. Les évènements sont mémorisés et chaque évènement ne peut être attendu que par un processus. Un évènement est représenté par un ECB (1 mot).



ECB

L'octet ETAT contient l'état de l'évènement : les états possibles sont : (REALISE ou NON REALISE) et ((ATTENDU et (SIMPLE ou MULTIPLE)) ou NON ATTENDU).
SIMPLE signifie 'évènement d'une attente simple'
MULTIPLE 'évènement d'une attente multiple'
L'adresse est celle du PCB du processus qui attend l'évènement, si l'évènement est ATTENDU. Sinon ce champ est à zéro.

6.2. L'attente

L'attente simple dans SYNCOP (attente d'un évènement unique) se base sur celle de TELCOM. L'attente multiple, qui a été ajoutée à TELCOM [10] pour les besoins de la ST1, a été modifiée pour SYNCOP : le processus TELCOM, après l'attente, ne peut pas connaître de façon simple l'avènement réalisé et ne peut attendre qu'un évènement parmi p. Par contre, un processus de SYNCOP peut attendre n évènements parmi p ($1 \leq n \leq p$).
Il construit une table à p entrées, chaque entrée contenant l'adresse d'un ECB, et fournit à WAITM l'adresse de cette table et le nombre n d'évènements attendus. Dès que n des évènements correspondant aux ECB pointés par la table seront réalisés, le processus sera réactivé. La primitive CHECK, en examinant les p ECB, pourra alors lui indiquer quels sont les n évènements réalisés.

6.3. Les états d'un évènement

L'état d'un évènement est modifié par les primitives WAIT, WAITM,POST et CHECK. Ces changements d'état d'un évènement peuvent provoquer des changements d'état du processus qui l'attend.
Les sous-états 'simple' ou 'multiple' ne peuvent être connus que si l'évènement est 'attendu'. C'est au moment de sa mise en attente que le processus signale si l'attente est simple ou multiple. La primitive WAIT force les sous-états 'attendu' et 'simple'. La primitive POST force le sous-état 'réalisé'.
Chaque fois qu'un évènement est 'simple', 'attendu' et 'réalisé' son ECB est nettoyé pour qu'on puisse à nouveau l'utiliser (il n'est plus 'attendu' ni 'réalisé') et le processus qui l'attendait est mis dans l'état 'prêt'.
La primitive WAITM force les sous-états 'attendu' et 'multiple' des p évènements. Chaque fois qu'un évènement est 'multiple', 'attendu' et 'réalisé', le compteur d'évènements attendus par le processus (PNWAIT dans le PCB) est décrémenté. Ce compteur est chargé à la valeur n quand le processus se

met en attente (WAITM). Quand PNWAIT passe à zéro (tous les évènements attendus sont ralisés), le processus passe dans l'état 'prêt'.
La primitive 'CHECK', appelée après WAITM, nettoie tous les ECB qui contiennent les états 'attendu' et 'réalisé' et signale au processus que les évènements associés sont réalisés.
En fait WAIT et WAITM ne mettent pas systématiquement le processus dans l'état 'en attente'. Si le ou les évènements attendus sont déjà réalisés, le processus ne passe pas dans l'état 'prêt' mais reste dans l'état 'actif' (WAIT passant). Il garde ainsi le contrôle. Si les évènements ne sont pas réalisés, il est bien mis dans l'état 'en attente'. Ceci permet de laisser le contrôle au processus tant qu'il a du travail. Notons qu'avec des processus qui feraient peu d'attente on pourrait compenser un peu l'absence de préemption en donnant systématiquement le contrôle au distributeur en sortant de WAIT ou WAITM.

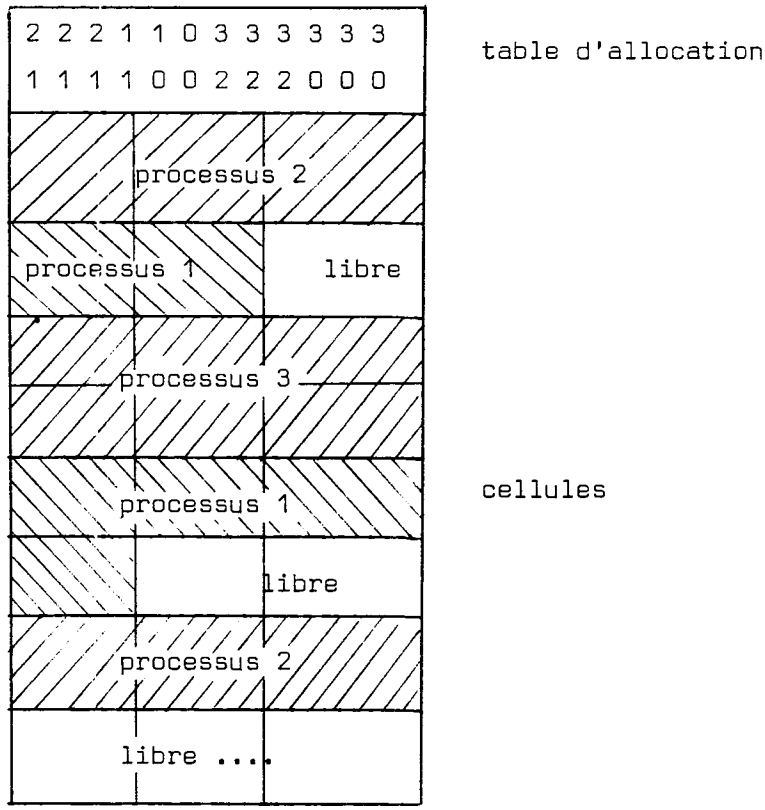
7. L'ALLOCATION DE MÉMOIRE

Une fois tous les modules du système chargés (système et processus) la mémoire centrale libre constitue un fond commun dans lequel le système viendra puiser pour satisfaire les demandes de mémoire des processus. Le nombre des processus étant important et leurs besoins en mémoire très variables au cour de leur vie, on a rejeté la méthode d'allocation d'une région fixe pour chaque processus au moment de sa création. Cette méthode est intéressante pour libérer la mémoire occupée par un processus à sa mort, mais elle fait perdre beaucoup de place dans le cas qui nous intéresse.

7.1. La table d'allocation

Avec ce fonds commun se pose le problème d'identifier le processus propriétaire de chaque élément de mémoire. La solution retenue consiste à diviser

la mémoire libre en blocs de taille fixe (cellules) et à allouer aux processus, à chaque demande, un nombre entier de cellules contigües. Une table d'allocation, construite après le chargement, à l'initialisation du système, contient autant d'octets qu'il y a de cellules. Chaque octet correspond à la cellule de même rang dans la zone des cellules et contient l'index (PINDX) du processus propriétaire de la cellule ou zéro si la cellule est libre.



EXEMPLE D'OCCUPATION MEMOIRE

L'identification du propriétaire par PINDX ne risque pas de créer de confusion puisque à la mort d'un processus toutes ses cellules sont libérées. Seuls les index des processus vivants peuvent figurer dans la table d'allocation.

On retrouve dans cette méthode l'inconvénient de l'allocation par zone : la perte de place. Les processus ne demandent pas toujours des blocs de mémoire dont la taille est un multiple de la taille des cellules. La différence (multiple supérieur - taille demandée) est inutilisée. Cette perte est d'autant plus importante que les cellules sont plus grandes. On perd aussi de la place avec la table d'allocation. La taille de cette table est d'autant plus importante que les cellules sont plus petites. La taille d'une cellule doit être un multiple de 8 (en octets) pour assurer que chaque bloc obtenu par un processus sera aligné sur une frontière de double-mot. L'étude présentée en annexe nous a conduits à choisir des cellules de 16 octets pour minimiser la place perdue.

7.2. Gestion de la mémoire

Avec cette méthode la gestion de la mémoire est une opération simple qui se fait par balayage de la table d'allocation.

GTMN balaie la table jusqu'à trouver suffisamment d'octets consécutifs nuls, met dans ces octets l'index du processus demandeur (PINDX) et calcule l'adresse de la zone mémoire trouvée à partir du déplacement du premier octet du groupe trouvé dans la table et de l'adresse du début de la zone des cellules. FRMN fait le calcul inverse pour trouver le premier octet de la table à partir de l'adresse de la zone à libérer et met cet octet et les suivants (selon la longueur de la zone à libérer) à zéro. Notons que FRMN peut contrôler que le processus libère bien de la mémoire qui lui appartient.

A la mort d'un processus, KILL ou EXIT remet à zéro tous les octets de la table d'allocation qui contiennent l'index du processus à tuer.
Toutes ces opérations se font uniquement sur la table d'allocation, ce qui, sous CP/67, limite la pagination. De plus le balayage de la table peut se faire par des instructions TRT ce qui rend les algorithmes très performants.

La recherche de place libre se fait selon le principe 'first fit'. La fragmentation de la mémoire, inconvénient inhérent à ce principe, n'est pas très importante : on utilise beaucoup de blocs de taille identique. Une commande à la console permet de surveiller ce phénomène: c'est la commande SIZE qui donne le nombre de 'trous'. On observe rarement plus de 10 trous et leur taille ne dépasse pas quelques dizaines d'octets.

Contrairement à de nombreux 'gros' systèmes, les processus sous SYNCOP peuvent attendre de la mémoire libre quand il n'y en a plus. Pour cela une table d'ECB est réservée statiquement à l'initialisation du système. Si GTMN ne peut satisfaire la demande d'un processus faute de mémoire libre, il permet au processus d'attendre sur un des ECB de cette table : chaque processus a son ECB. Chaque fois que de la mémoire est libérée (FRMN ou KILL/EXIT) tous les ECB attendus de cette table sont postés. Si plusieurs processus sont en attente de mémoire et qu'on n'en a pas libéré suffisamment, les plus prioritaires seront les premiers servis, les autres se remettront en attente. Ce mécanisme présente l'inconvénient de provoquer des interblocages dans le cas où les seuls processus qui pourraient libérer de la mémoire sont ceux qui en attendent. Une commande (CORE)* donne la taille de mémoire qui n'a jamais été utilisée depuis le chargement du système. On peut ainsi connaître la taille de la mémoire nécessaire.et éviter ce blocage en augmentant la taille de la mémoire.
Quand un processus est tué (KILL ou EXIT), son ECB est remis à zéro dans la table s'il était attendu. Ainsi, on ne risque pas de le poster après sa mort.

---

* *Voir Annexe IV*

8. LES RÉVEILS

Les principes des réveils dans SYNCOP sont les mêmes que ceux de TELCOM dans la version utilisée pour la ST1. Rappelons ces principes.
Deux types de réveils sont possibles :

- Poster un ECB au bout d'un délai donné (type 1). Ceci permet à un processus de dormir pendant un temps fixé.
- Exécuter une séquence au bout d'un délai donné. (type 2). Cette séquence est exécutée sans perturber le processus à l'expiration du délai.

8.1. Principe

Chaque demande de réveil est décrite par un bloc de contrôle (BRV) qui contient :

- le délai demandé (BRVINTVL)
- le type du réveil (BRVCODE)
- l'adresse de l'ECB à poster ou de la séquence à exécuter à l'expiration du délai (BRVECB)
- l'index et l'identificateur du processus qui lance le réveil (BRVINDX)
- la référence du réveil (BRVREF) qui sera passée à la séquence dans le cas d'un réveil de type 2.

Le système gère une table dont chaque entrée contient l'heure relative d'échéance d'un réveil et l'adresse du BRV associé. Ces entrées sont rangées dans l'ordre des heures d'échéance croissantes. A la première demande de réveil (STIMER) un élément est mis en tête de table et l'horloge de la machine est chargée avec le délai demandé. A chaque nouvelle demande un élément est créé et rangé à sa place dans la table (si c'est en tête, avec chargement de l'horloge).
Quand l'horloge passe à zéro (interruption externe), l'action correspondant

au BRV pointé par l'élément de tête de la table est exécutée, cet élément est supprimé de la table, le BRV est libéré. L'élément qui devient alors le premier de la table est utilisé pour charger l'horloge (à moins que la table soit vide). Le système peut ainsi gérer des demandes de réveil qui se recouvrent.

Si l'action à exécuter est un simple POST (type 1) elle est exécutée par la routine de traitement des interruptions externes, sinon (type 2) elle est confiée à un processus spécialisé, le processus REVEIL. Pour cela, la routine de traitement des interruptions externes enfile le BRV dans une file sur laquelle le processus REVEIL consomme. Il traite tous les BRV qui s'y trouvent, dans l'ordre d'arrivée, en se branchant à l'adresse contenue dans le BRV. Cette solution suppose que les séquences ainsi déroulées sous le contrôle du processus REVEIL présentent les caractéristiques suivantes :

- Séquences courtes pour ne pas pénaliser les autres demandes en attente dans la file.
- Séquences sans attente (WAIT ou WAITM), pour la même raison.
- Retour au processus REVEIL.

Ces contraintes se sont révélées supportables. Il a toutefois été envisagé d'introduire un troisième type de réveil pour le cas où ces contraintes seraient trop fortes. Un processus serait créé pour exécuter la séquence et détruit à la fin de cette séquence. Tout peut alors être permis dans la séquence. On ne peut pas créer un processus à l'aide d'un réveil de type 2 : CREATE peut provoquer une attente si la mémoire fait défaut.

Ce service est d'ailleurs disponible avec les deux types existants : on peut créer un processus qui se mettra initialement en attente puis lancer un réveil de type 1 qui activera le processus.

3.2. La gestion des réveils

STIMER permet de lancer un réveil. Un BRV est créé, pointé par un élément de la table qui est inséré en fonction de son heure d'échéance.

RTIMER fait l'opération inverse : l'élément est supprimé de la table et le BRV est détruit. S'il n'est pas dans la table on le cherche dans la file du processus REVEIL.
A la mort d'un processus (KILL ou EXIT), on parcourt la table à la recherche de tous les BRV qui contiennent l'identification du processus à tuer (BRVIND et on leur applique le même traitement que RTIMER. Les ECB des réveils de ty 1 ne pourront donc pas être postés après la mort du processus.

9. Les listes

Les listes permettent aux processus de se communiquer de l'information. Chaque liste est la propriété d'un processus, celui qui l'a créée. Seul ce processus peut consommer dans la liste, mais tout processus peut y produire. Le nombre des producteurs est fixé à la création de la liste. Les listes fonctionnent suivant le principe 'premier entré premier sorti' (FIFO). Le nombre d'éléments que peut contenir une liste est limité et fixé à la création de la liste. Ceci évite que les producteurs travaillent plus vite que le consommateur. Quand la liste est pleine, les producteurs peuvent attendre que le consommateur ait retiré un élément.
Les éléments d'une liste ne sont pas chaînés entre eux, mais pointés par le bloc descripteur de la liste (LIST). Ce bloc contient en plus de ces pointeurs vers les éléments :

- un ECB par producteur pour attente de liste non pleine
- un ECB permettant au consommateur d'attendre que la liste soit non vide
- des pointeurs sur les pointeurs d'éléments permettant de connaître ceux qui sont libres et ceux qui sont significatifs
- l'index du processus propriétaire (LPROPINX)
- la taille des éléments (LELSIZE)
- l'adresse du pointeur de liste (LPTRLST)
- le pointeur vers le bloc LIST suivant appartenant au même processus (LNEXLST).

Il existe deux types de listes :
- les listes à éléments de taille fixe (≤ 255 octets)
- les listes à éléments de taille variable

Pour une liste à éléments de taille fixe, tous les éléments ont la même taille, indiquée dans LELSIZE. Le contenu des éléments est entièrement libre.

Dans les listes à éléments de taille variable la longueur de l'élément est indiquée dans les deux premiers octets de l'élément, par le producteur. Dans ce cas LELSIZE vaut 0.

Les listes appartenant à un même processus sont chaînées entre elles par le pointeur LNEXLST. La première liste est pointée par le PCB du propriétaire (PLSTPTR). Ceci permet, à la mort du propriétaire de supprimer toutes ses listes.

Le paramètre permettant d'accéder à une liste n'est pas l'adresse de son bloc descripteur, mais l'adresse d'un mot (pointeur de liste) qui pointe sur ce bloc. Ceci évite l'accès à une liste détruite. Ce pointeur est chargé au moment de la création de la liste et remis à zéro au moment de sa destruction. Le premier octet contient toujours la taille des éléments de la liste ou zéro si c'est (était) une liste à éléments de taille variable. Ainsi, si un producteur essaie d'enfiler un élément dans une liste détruite, on s'en apercevra et on pourra libérer l'élément.

Les éléments enfilés dans une liste sont des blocs de mémoire qui ont préalablement été acquis (GTMN)* par le producteur et donc lui appartiennent. Au moment où ils sont enfilés (PUTLST) ils deviennent la propriété du consommateur. C'est pour effectuer cette opération que la taille de l'élément doit être connue ainsi que l'index du processus consommateur. (LPROPINX).

Les ECB concernant les listes ne risquent pas d'être postés après la mort d'un processus. L'ECB du consommateur est posté par un producteur quand il enfile un élément dans la liste et que la liste était vide. Les listes étant supprimées à la mort d'un processus, aucun producteur ne pourra enfiler d'élément et l'ECB ne sera pas posté.

---

* *Un bloc est donc formé d'un nombre entier de cellules.*

Les ECB des producteurs sont postés quand la liste est pleine et que le consommateur retire un élément (GETLST). Ceci a pour effet de réveiller un producteur qui attend de la place dans la liste. Si ce processus est mort, il ne faut pas poster cet ECB. C'est pourquoi, à la mort d'un processus (KILL/EXIT), on examine les ECB d'attente des producteurs dans toutes les listes. Ceux qui sont attendus par le processus à supprimer sont remis à zéro, ainsi ils ne seront plus postés.

10. LES RESSOURCES

Les primitives ENQ et DEQ permettent l'exclusion mutuelle entre les processus qui utilisent des ressources non partageables. Un processus qui veut utiliser une telle ressource appelle ENQ. Quand il libère la ressource, il appelle DEQ.

Chaque ressource utilisée est représentée par un bloc de contrôle (RCT). Ces blocs sont chaînés entre eux. Les ressources libres ne sont pas représentées.

Chaque RCT contient, en plus du pointeur vers le RCT suivant :

- le nom de la ressource
- l'identificateur du processus qui l'utilise
- un pointeur vers la chaîne des ECB.

Cette chaîne comprend un ECB pour chaque processus qui attend la libération de la ressource.

A la mort d'un processus, on libère les ressources qu'il utilise en parcourant la chaîne des RCT. Les processus qui attendaient ces ressources sont réveillés. Si le processus qu'on supprime est en attente de ressources, les ECB qu'il utilise à cet effet sont retirés des chaînes d'ECB et libérés. Ainsi on ne risque pas de réveiller un prodessus mort quand les ressources qu'il attendait seront libérées.

11. LES ENTRÉES-SORTIES

La gestion des entrées-scrties de SYNCOP est la même que celle de TELCOM. La La seule différence réside dans la possibilité de supprimer un processus qui utilise une unité d'entrée-sortie.
Rappelons brièvement le principe de la gestion des entrées-sorties. Chaque unité d'entrée-sortie de la machine virtuelle est gérée par un processus. Seul ce processus peut lancer des opérations d'entrée-sortie sur cette unité. Il dispose pour cela d'un ensemble de sous-programmes, appelé 'device routine', qui préparent l'opération en constituant un bloc de contrôle (BIO) qui la décrit. Il existe une 'device routine' par type d'unité. Le BIO est passé à la routine EXCP (Execute Channel Program) qui lance l'opération par une instruction SIO. Les processus peuvent lancer les opérations d'entrée-sortie avec ou sans attente. Si elles sont lancées sans attente et que l'unité est occupée, EXCP range les BIO correspondants dans une file d'attente propre à l'unité. Si l'opération est lancée avec attente, le processus dispose d'un ECB dans le BIO sur lequel il attendra la fin de l'opération.

Les interruptions d'entrée-sortie sont traitées par le module IOINT, qui est chargé de terminer l'opération. Chaque unité d'entrée-sortie est représentée par une table (UCT) qui décrit l'état de l'unité. Pour une opération avec attente, IOINT poste l'ECB correspondant à l'opération pour en signaler la fin au processus qui l'a lancée. Pour une opération sans attente, IOINT appelle EXCP pour lancer la prochaine opération en attente et libère le BIO de l'opération terminée.

Quand on supprime un processus qui gère une unité d'entrée-sortie, il faut supprimer l'opération en cours, s'il y en a une,et supprimer la file des BIO en attente, si les opérations se font sans attente.
Pour cela chaque processus qui gère une unité, contient dans son PCB l'adresse de l'UCT correspondante (ou zéro s'il n'y a pas d'unité). KILL ou EXIT marque dans l'UCT du processus à supprimer que l'unité est en cours de libération, si une opération est en cours. Sinon, l'unité est marquée libre.
Puis on indique dans l'UCT qu'il n'y a plus d'opération en cours ni en attente de lancement. Les BIO seront libérés avec l'ensemble de la mémoire appartenant au processus.

IOINT ignore les interruptions qui sont provoquées par une unité en cours de libération. Après une telle interruption, il marque l'UCT libre. On est sûr ainsi de ne pas poster d'ECB de fin d'entrée-sortie attendu par un processus mort ni de lancer des opérations pour le compte d'un processus mort.
L'opération en cours n'est pas suspendue, elle est seulement ignorée. Ceci évite de lancer un HIO, opération simulée par CP et donc coûteuse.

12. TRAITEMENT DES INTERRUPTIONS PROGRAMME

Toute erreur dans un processus ou dans le système déclenche une interruption programme. Ces interruptions sont traitées par le module PGMCK qui sort un dump de la mémoire et des registres de la machine virtuelle ainsi que les SNAP (cf 13.4), envoie un message à l'opérateur de CP et recharge tout le système qui se réinitialise.
Cette solution est adoptée dans le cas où tous les processus sont des 'processus système', c'est-à-dire des processus indispensables. Mais s'il existe des 'processus utilisateurs' que l'on peut supprimer sans nuire au reste du système, si l'un de ces processus provoque une interruption programme, PGMCK le supprimera.
Dans ce cas, au lieu de recharger tout le système, PGMCK doit vérifier si le processus fautif est un processus utilisateur tuable (ce qui peut être indiqué dans son PCB lors de sa création) et appeler KILL pour le supprimer, avant de rendre le contrôle au distributeur.
Ceci permet d'ajouter aux processus existants des processus peu fiables : des processus en cours de tests ou des processus inconnus. Ainsi une erreur dans un de ces processus ne nuira pas au reste du système.

13. CONSÉQUENCES DE LA SUPPRESSION DES PROCESSUS SUR LES ÉVÈNEMENTS

Quand un processus change d'état ou que son compteur d'attente est modifié on accède au PCB par l'adresse contenue dans l'ECB. On a vu que cette adresse n'identifie pas de façon unique le processus. On risque donc de modifier l'état ou le compteur d'attente d'un processus qui n'est pas celui qu'on veut : il ne faut pas "poster" un processus mort.
On peut distinguer deux catégories d'évènements :
- les évènements système
- les évènements processus.

Les évènements système sont ceux qui restent sous le contrôle du système. Ce sont ceux qui concernent les opérations d'entrée-sortie, les listes, la mémoire libre, les réveils, les ressources. Ils sont tous postés par le système et on a vu qu'ils ne risquent pas d'être postés si le processus qui les attendait est mort.

Les évènements processus sont ceux qui sont attendus et postés par les processus eux-mêmes. C'est pour ceux-là qu'a été introduite la primitive STATUS : les processus peuvent, avant de poster un évènement, vérifier que le processus qu'ils veulent réveiller est bien vivant. Ce contrôle n'a pas été fait systématiquement dans POST pour éviter des contrôles inutiles pour les évènements système qui sont les plus nombreux. On aurait aussi pu mettre dans l'ECB l'identificateur du processus en attente au lieu de l'adresse de son PCB. Mais cela aurait compliqué la recherche du PCB et augmenté l'overhead du système. De plus les ECB n'étant pas déclarés (au sens des langages)cette solution serait insuffisante:l'ECB peut avoir été rendu à la mémoire libre au moment où le processus est mort. La déclaration ou la demande d'ECB au système a été jugée trop couteûse pour les besoins.

14. LES OUTILS DE TESTS

Pour aider à la mise au point des programmes, SYNCOP fournit des outils de test. Ces outils se limitent à des traces. SYNCOP se déroulant dans une machine virtuelle de CP/67 on dispose donc des commandes de CP pour l'aide à la mise au point.

14.1. Trace du distributeur

Le distributeur enregistre dans une zone de mémoire qui lui est propre la succession des processus qu'il active ainsi que les attentes qu'il fait quand aucun processus n'est prêt à être activé. On peut ainsi connaître, au moment où une erreur se produit, les derniers processus actifs. On peut aussi vérifier que les processus se synchronisent correctement.

14.2. Trace des listes

A chaque liste est associée une zone de trace. Chaque fois qu'un élément est ajouté ou retiré (PUTLST, GETLST, LOOKLST), on note dans cette zone l'identification du processus qui agit sur la liste et le nombre d'éléments contenus dans la liste à la fin de l'opération. Ceci donne des indications précieuses sur le fonctionnement des processus qui travaillent sur chaque liste et sur leur synchronisation. On peut aussi mieux appprécier la taille nécessaire pour chaque liste.

14.3. SVC 1

On peut introduire dans les programmes à mettre au point des instructions

SVC 1. Le passage sur cette instruction provoque le rangement dans une zone de trace de l'adresse de l'instruction et l'incrémentation d'un compteur associé à l'instruction. On peut ainsi connaître l'ordre dans lequel on est passé sur ces instructions et le nombre de fois qu'on est passé sur chacune.

14.4. Les 'snaps'

Si l'instruction SVC 1 est insuffisante, on a la possibilité d'obtenir une trace plus précise des programmes à tester. Un sous-programme (SNAP) permet d'écrire dans un F-espace [21] une zone de mémoire quelconque et/ou les 16 registres généraux. A chaque appel de SNAP ces informations sont rangées dans le F-espace de la machine virtuelle. Le F-espace peut être sorti sur imprimante par une commande à la console. En cas d'interruption programme, PGMCK sort également le F-espace sur l'imprimante.

15. PERFORMANCES

SYNCOP a été conçu après TELCOM. En profitant de l'expérience acquise on a cherché à améliorer les performances de SYNCOP par rapport à TELCOM. L'annexe V donne les mesures des performances.

15.1. La gestion de la mémoire

La gestion de la mémoire de TELCOM était inspirée de celle d'OS/360 [9]. Chaque zone libre contient un descripteur appelé FQE qui contient la longueur de la zone libre. Chaque FQE se trouve à la fin de la zone qu'il

décrit. Les FQE sont chaînés entre eux dans l'ordre des adresses croissantes. Pour allouer une zone de mémoire, il faut parcourir la chaîne des FQE jusqu'à ce qu'on trouve une zone libre assez grande. Pour libérer une zone de mémoire, il faut parcourir la chaîne des FQE pour trouver l'endroit où insérer le nouveau FQE ou pour trouver le ou les FQE à modifier (si la zone libérée est contigüe à une ou deux zones libres). Sous CP, cette opération de balayage de la chaîne des FQE répartis dans toute la mémoire provoque des fautes de pages et est donc coûteuse en temps.

La gestion de mémoire adoptée pour SYNCOP a l'avantage de n'utiliser que la table d'allocation. Le risque de faute de page est donc moindre. On a ajouté une option à la primitive GTMN, qui permet d'obtenir une zone de mémoire située dans une seule page. Ceci est particulièrement utile quand la zone de mémoire doit contenir des informations à écrire ou lues sur une unité d'entrée-sortie. Si la zone est dans une seule page la traduction des programmes canal effectuée par CP est plus rapide.

15.2. Les processus dynamiques

Dans TELCOM tous les processus sont créés à l'initialisation du système. Tous ne sont pas nécessaires et certains sont donc en attente d'un évènement qui ne se produira que beaucoup plus tard (peut-être jamais). Chaque fois que le distributeur a le contrôle, il perd du temps à examiner l'état de processus qui sont toujours en attente.
SYNCOP permet de créer les processus au moment où ils ont du travail et de les détruire quand ce travail est fini. Le distributeur ne perd pas de temps avec des processus éternellement inactivables.

15.3. Conventions de liaison

Les conventions de liaison de TELCOM sont inspirées de celles d'OS. La section appelante doit acquérir une zone de sauvegarde (GTMN) qu'elle fournit à la section appelée. A l'entrée de la section appelée les registres sont rangés dans la zone de sauvegarde de l'appelant et une nouvelle zone de sauvegarde est acquise pour pouvoir appeler d'autres sections. Chaque appel de sous-programmes coûte donc l'acquisition d'une zone de sauvegarde (GTMN) et sa libération au retour (FRMN).

SYNCOP utilise des piles pour ces opérations. Chaque processus possède une pile, acquise lors de sa création. A l'entrée d'un sous-programme, l'adresse de retour et les registres à sauvegarder sont empilés dans cette pile. A la sortie ils sont dépilés. On économise ainsi un appel à GTMN et un à FRMN pour chaque appel de sous-prog amme.

15.4. L'absence de préemption

TELCOM est un système avec préemption. A tout moment un processus peut être interrompu et les ressources qu'il manipule peuvent être demandées par le processus interrompant. La gestion des ressources est dont délicate et les appels à ENQ et DEQ sont nombreux.

SYNCOP étant sans préemption, la gestion des ressources est plus simple. Seules doivent être protégées les ressources qui sont utilisées lors du traitement des interruptions ou quand un processus se met en attente.

16. Conclusion

Les choix qui ont été faits pour l'implémentation de SYNCOP sont des compromis. Nous avons cherché à réaliser un système simple et performant tout en essayant d'y inclure des contrôles protégeant les processus. Les contrôles trop coûteux ont été abandonnés, comme la protection mémoire : il ne s'agissait pas de réaliser un système général acceptant tout type de processus.

CHAPITRE IV

LES PROCEDURES DE REPRISE DES SERVEURS IBM/360

Nous avons vu que le service de transport est capable de déceler les pannes distantes et de prévenir les abonnés locaux concernés. Mais il laisse aux abonnés le soin de réagir à ces pannes, sa seule action est de fermer les flots touchés par les pannes.

D'autre part, les pannes locales peuvent également compromettre le travail des abonnés, sans les prévenir.

Après avoir décrit les fonctions des abonnés réalisés sur le 360/67 du CICG, nous analyserons les conséquences des pannes sur ces abonnés et en déduirons le comportement qu'ils doivent avoir en cas de panne. Nous décrirons enfin l'implémentation de ces abonnés.

1. Les serveurs du 360

Le 360/67 du CICG est connecté au réseau Cyclades sous les deux systèmes d'exploitation ASP/OS-MVT et CP/67. Des serveurs ont été réalisés pour chacun de ces deux systèmes.

1.1. Le serveur CP/67

Le serveur CP/67 permet à tout utilisateur du réseau d'accéder au service CP/67. Trois types d'entrées permettent cet accès.

- Entrée terminal
- Entrée lecteur de cartes
- Entrée imprimante

L'entrée terminal permet de connecter un terminal distant au serveur et, à partir de ce terminal, de piloter une machine virtuelle de CP/67.
L'entrée lecteur de cartes permet de connecter un lecteur de cartes distant au serveur et, à partir de ce lecteur, d'envoyer des cartes aux machines virtuelles de CP/67.
L'entrée imprimante permet de connecter au serveur une imprimante distante et de recevoir sur cette imprimante les listes provenant des machines virtuelles.

1.2. Le serveur batch ASP/OS-MVT

Le serveur batch ASP/OS-MVT permet aux utilisateurs du réseau d'accéder au service batch. Trois entrées sont disponibles.

- Entrée console
- Entrée lecteur de cartes
- Entrée imprimante

L'entrée console permet la connexion d'un terminal au serveur batch. A partir de ce terminal l'utilisateur peut suivre le déroulement de ses travaux en envoyant des commandes que le serveur transmet à ASP par l'intermédiaire d'une pseudo-console.
L'entrée lecteur de cartes permet la connexion d'un lecteur de cartes distant au serveur. L'utilisateur peut envoyer des travaux de son lecteur de cartes à ASP.
L'entrée imprimante permet la connexion d'une imprimante distante au serveur. Le serveur envoie par cette connexion les listes venant de ASP destinées à cette imprimante.

2. LES PANNES

Sous chacun des deux systèmes d'exploitation des pannes peuvent se produire pendant le transfert de cartes ou de listes ainsi que pendant l'utilisation d'une console ou d'un terminal. Quelle que soit leur origine, ces pannes doivent perturber le moins possible le travail en cours.
Les perturbations acceptables pour chaque type d'entrée vont être examinées.

2.1. Entrées console ou terminal

L'utilisateur d'une console (ASP) ou d'un terminal (CP/67) sait à chaque instant où il en est dans son travail. S'il est interrompu par une panne, il demande seulement de pouvoir reconnecter le terminal qu'il utilisait, ou un autre, le plus tôt possible pour pouvoir reprendre son travail. Il sait que la dernière commande sans réponse n'est peut-être pas traitée, mais dès que la connexion est rétablie, il peut le vérifier.
Dès que la reconnexion est possible, la machine virtuelle de l'utilisateur CP ou la pseudo-console ASP doit être libre pour que le travail puisse reprendre.

2.2. Entrées lecteur de cartes

En cas de panne pendant un transfert de cartes, les dernières cartes lues par le lecteur distant ne sont pas encore parvenues au serveur : ces cartes peuvent se trouver :

- chez l'abonné client qui gère le lecteur de cartes
- dans la ST du client
- dans le réseau de communication
- dans la ST du serveur.

Elles devraient être réémises vers le serveur lorsque la connexion est rétablie. Mais l'utilisateur du lecteur de cartes ignore quelles sont ces cartes qu'il doit relire. S'il en relit trop, certaines cartes seront reçues en double par le serveur. S'il n'en relit pas assez certaines cartes ne parviendront jamais au serveur. Cette solution ne peut donc être retenue : il n'y a aucune certitude sur la réception correcte du paquet de cartes en cours de transfert au moment de la cassure.
La solution retenue consiste, pour l'utilisateur, à relire depuis le début le paquet de cartes dont le transfert a été interrompu. Le serveur doit alors ignorer toutes les cartes qu'il a reçues depuis le début du paquet en cours jusqu'à la cassure, puisqu'elles seront à nouveau reçues après la reconnexion.

2.3. Entrées imprimante

On peut comparer les pannes se produisant pendant le transfert des listes et des cartes. Les dernières lignes émises par le serveur ne sont pas encore imprimées sur l'imprimante distante au moment de la panne. Mais dans le cas d'une imprimante on peut admettre de reprendre le transfert quelques lignes avant le point où la panne s'est produite, plutôt que de retransmettre toute la liste : l'utilisateur de la liste peut s'apercevoir qu'un certain nombre de lignes sont doublées.
Cependant, il faudra, après reconnexion, que le serveur reprenne le transfert de la liste en un point tel qu'aucune ligne ne soit perdue, c'est-à-dire n lignes avant la dernière émise, n étant supérieur au nombre maximum de lignes émises et non imprimées au moment de la panne.
Cette solution,satisfaisante pour la grande majorité des listes, peut être insuffisante dans le cas de listes de résultats sans numérotation de pages ou lignes. Pour ces listes il faut reprendre la transmission depuis le début.

2.4. Remarque

La stratégie de reprise décrite ici est la même que celle employée par la plupart des services de traitement par lots ou de temps partagé en dehors

de tout contexte réseau. On cherche seulement à l'étendre à des services accédés à travers un réseau.
Pour un service hors réseau, les seules pannes à considérer sont celles :
- du matériel utilisé par le service, c'est-à-dire de la machine sur laquelle quelle est réalisé ce service,
- du logiciel qui réalise le service (le système d'exploitation lui-même pour les services qui nous intéressent).

Si ce service est mis sur le réseau, il faut considérer en plus les pannes
- du serveur (abonné et (sous)-système utilisé)
- du service de transport (les ST et le réseau de communication)
- du client.

De même que les services hors réseau du type traitement par lots ou temps partagé sont prévus pour redémarrer de la même façon après une panne du matériel ou du logiciel, les services accédés à travers le réseau doivent redémarrer de la même façon quelle que soit la panne qui a conduit au redémarrage.Vu par l'utilisateur, le comportement du service doit toujours être le même au redémarrage. On s'efforce de suivre ce principe dans la mesure où il n'est pas trop coûteux.

3. SOLUTIONS DE REPRISE DES SERVEURS

3.1. Entrée console ou terminal

En ce qui concerne les terminaux (CP) et les consoles (ASP) la communication entre le serveur et le système hôte est basée sur le même principe. Le serveur associe une unité virtuelle (pseudo-console sous ASP, 1052X sous CP) à chaque console ou terminal distant connecté. Le système (ASP ou CP) communique avec le serveur par l'intermédiaire des unités virtuelles.

Ces unités virtuelles doivent être libérées par le serveur quand il ne les utilise plus. Elles sont en nombre limité et si le serveur ne les libère pas au fur et à mesure, il risque de ne plus en avoir pour les nouvelles connexions.
De plus, sous CP/67, les 1052X sont utilisés pour passer les commandes émises par l'utilisateur distant pour piloter sa machine virtuelle. Si, par suite d'une panne, les communications sont rompues entre l'utilisateur et le 1052X associé, la machine virtuelle ne recevant plus de commande est bloquée et inutilisable, même à partir d'un autre terminal (local ou réseau).

En cas de panne distante, le serveur est prévenu par sa ST et libère l'unité virtuelle. Cette libération, sous CP (DIAGNOSE DISCONNECT), provoque en plus le logout automatique de la machine virtuelle pilotée par le 1052X libéré.

Les pannes locales doivent être séparées en deux types :
- les erreurs du système hôte (ASP ou CP)
- les erreurs de la tâche OS ou de la machine virtuelle du serveur.

Les erreurs du système hôte provoquent automatiquement la libération des unités virtuelles. Au redémarrage du système ces unités sont à nouveau disponibles.
Sous OS, l'erreur de la tâche OS provoque également la libération des pseudo-console.
Sous CP, l'erreur de la machine virtuelle spécialisée ne provoque pas cette libération, mais le module PGMCK de SYNCOP, qui prend le contrôle dans ce cas, effectue lui-même cette libération par l'instruction DIAGNOSE DISCONNECT sur tous les 1052X avant de recharger la machine virtuelle.
Dans tous les cas de panne les unités virtuelles sont donc bien libérées et les utilisateurs peuvent à nouveau travailler dès le redémarrage.

3.2. La file d'attente

3.2.1. La file d'attente OS

Sous OS le serveur ne dispose que d'une voie de communication avec ASP pour transmettre les cartes reçues du réseau et d'une autre voie pour recevoir d'ASP les listes à transmettre vers les imprimantes du réseau (cf [11]).
Pour pouvoir recevoir simultanément des cartes venant de plusieurs lecteurs distants, le serveur utilise une file d'attente sur disque où il range les cartes qu'il reçoit sous forme de plusieurs fichiers simultanément. Les fichiers complets sont ensuite passés un à un à ASP.
En sens inverse les listes sont reçues une à une de ASP et rangées dans la file d'attente. Ces fichiers liste peuvent être lus simultanément par le serveur pour être transmis aux imprimantes distantes quand elles sont connectées.

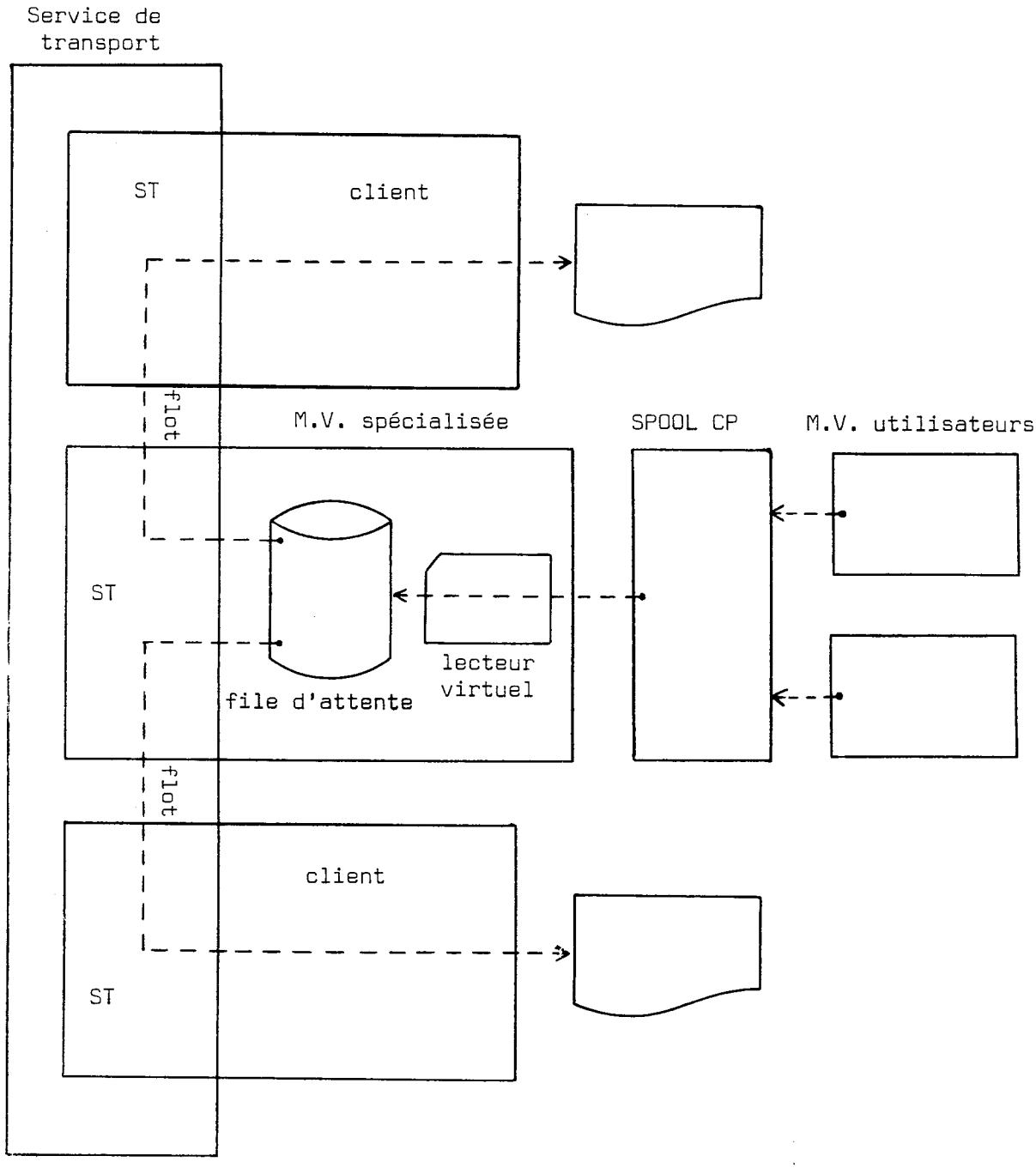
3.2.2. La file d'attente CP

Les cartes reçues par le serveur CP ainsi que les listes qu'il émet passent de la même façon par une file d'attente sur disque.

3.2.2.1. Les listes

L'utilisateur d'une machine virtuelle qui veut envoyer une liste vers une imprimante distante du réseau passe cette liste à la machine virtuelle spécialisée par le spool de CP (cf [18]). La liste est donc disponible sur le lecteur virtuel de la M.V. spécialisée.
Dans la première version du serveur CP (cf [10]), ce fichier restait sur le lecteur virtuel jusqu'à ce que l'imprimante distante destinataire soit connectée. Dès la connexion le fichier était alors lu et transmis sur le réseau. Quand plusieurs utilisateurs produisent de tels fichiers, ces fichiers se trouvent rangés dans le spool de CP dans leur ordre de production. Si l'imprimante destinataire du fichier de tête ne se connectait pas, les fichiers suivants étaient bloqués. En effet le spool de CP ne permet de lire les fichiers que dans l'ordre dans lequel ils ont été produits.

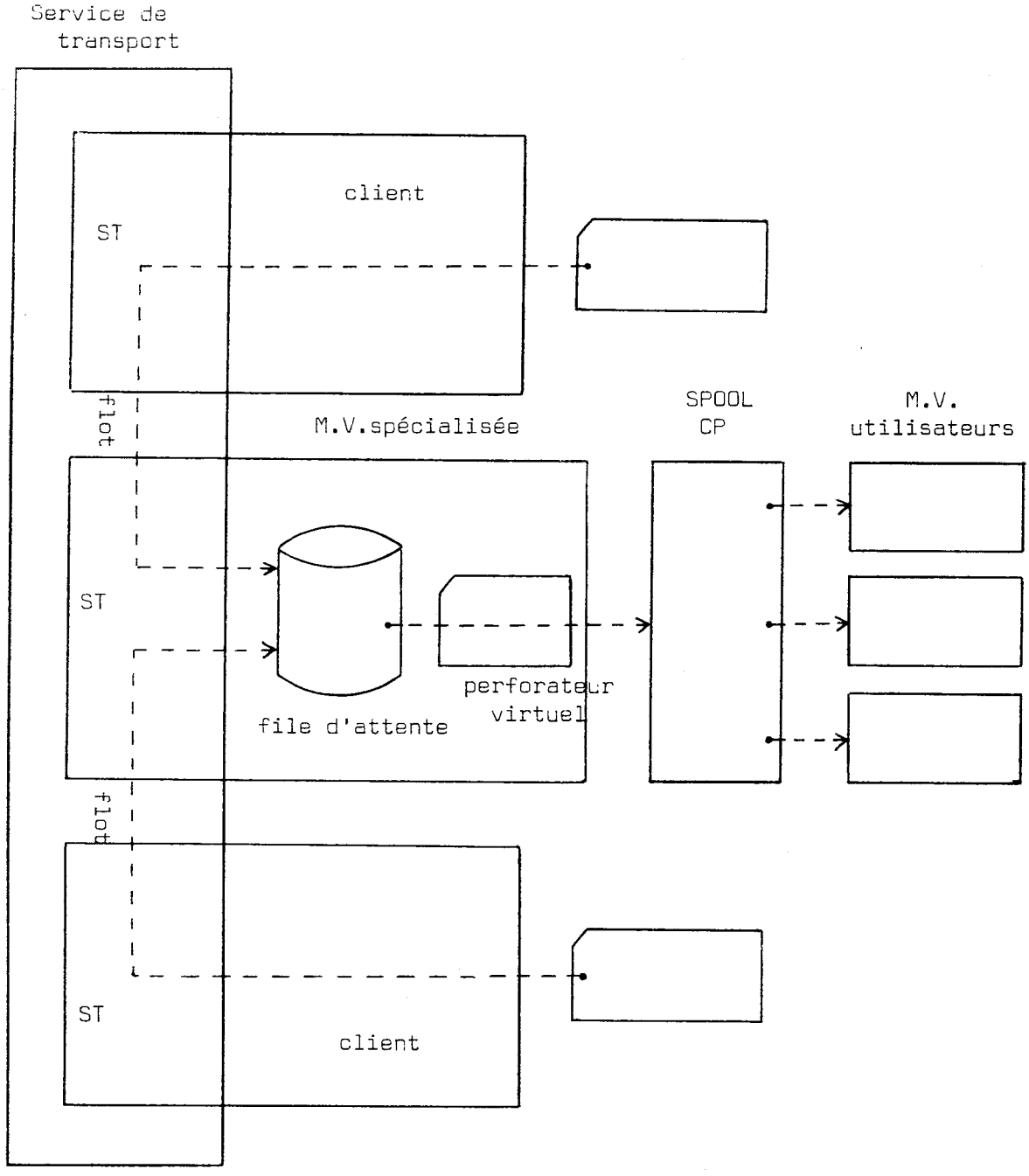
La multiplication des lecteur virtuels de la machine spécialisée a permis de diminuer le risque de blocage de ces fichiers, mais sans l'éliminer .
Si la M.V. spécialisée possède n lecteurs virtuels et que les imprimantes destinataires des n premiers fichiers produits ne se connectent pas, les fichiers suivants sont bloquées jusqu'à ce que l'une des imprimantes destinataires des n premiers fichiers se connecte.
Pour éviter cet inconvénient, la M.V. spécialisée ne possède qu'un lecteur virtuel, mais les fichiers sont lus dès qu'ils sont disponibles dans le spool de CP. Ils sont alors rangés dans la file d'attente. Dès qu'une imprimante distante se connecte, le serveur cherche dans la file d'attente les fichiers qui lui sont destinés et les lui envoie successivement. Plusieurs imprimantes peuvent être connectées simultanément, les fichiers leur sont envoyés en parallèle puisqu'on peut lire plusieurs fichiers simultanément dans la file d'attente.



Transmission des listes vers les imprimantes distantes.
Les transmissions figurées par les flèches peuvent être simultanées.
Le stockage est fait dans la file d'attente, le séjour dans le SPOOL est très court (temps de l'écriture + temps de la lecture).

3.2.2.2. Les cartes

Les cartes reçues des lecteurs de cartes du réseau sont passées aux M.V. des utilisateurs destinataires par le spool de CP. Le perforateur virtuel de la M.V. spécialisée permet de les ranger dans le spool où elles sont disponibles pour les M.V. destinataires.
On peut envisager, une fois un lecteur distant connecté au serveur, de recevoir les cartes qui arrivent et de les ranger au fur et à mesure dans le spool de CP. Pour que plusieurs lecteurs distants puissent envoyer des cartes simultanément, on peut disposer d'autant de perforateurs virtuels qu'il y a de connexions.
Mais la file d'attente, nécessaire pour les listes, peut également être utilisée pour les cartes. Ceci améliore la fiabilité des transferts de cartes et un seul perforateur virtuel est nécessaire à la machine virtuelle spécialisée.
Les cartes reçues du réseau sont rangées dans la file d'attente. Plusieurs fichiers peuvent être rangés en parallèle. Ces fichiers sont ensuite envoyés dans le spool de CP un à un par l'intermédiaire du perforateur virtuel. De cette façon les transferts de la file d'attente au spool se font très rapidement (transfert de disque à disque). Seule une panne locale peut interrompre ce transfert. Ces pannes sont les plus rares et le transfert est rapide. La probabilité d'incident à ce moment est donc très faible. Elle est plus forte pendant le rangement des cartes reçues du réseau dans la file d'attente. Ce transfert est plus lent (vitesse du réseau) et peut être interrompu par une panne distante ou locale. Mais on peut prévoir la gestion de la file d'attente en conséquence, alors qu'il est très délicat de modifier la gestion du spool de CP.



Transmission des cartes vers les machines virtuelles.
Toutes les transmissions figurées par les flèches peuvent être simultanées, le stockage est fait dans le spool CP, le séjour dans la file d'attente est très court (temps de l'écriture et de la lecture).

3,3, Entrées lecteur de cartes et imprimante

La gestion de la file d'attente a été définie en fonction de son utilisation par les entrées lecteur de cartes et imprimante et des réactions que doit avoir le serveur en cas de panne.

3,3,1, Imprimante

Les listes qui se trouvent dans la file d'attente sont à l'abri de toute panne. Seules les pannes survenant pendant le transfert des listes entre le système hôte (ASP ou CP) et la file d'attente ou entre la file d'attente et le réseau peuvent avoir des conséquences.

Panne pendant le rangement dans la file d'attente

Seules des pannes locales peuvent interrompre le rangement dans la file d'attente. Après une erreur du système hôte, au redémarrage, le fichier en cours de transfert, sous OS comme sous CP est à nouveau disponible intégralement dans le système hôte. Le serveur le retransmettra à nouveau vers la file d'attente, depuis le début du fichier. La partie qui avait déjà été rangée avant l'erreur devra avoir disparu. Pour cette raison tout fichier de type liste en écriture dans la file d'attente n'y est effectivement enregistré que s'il est fermé. Ainsi, après une panne survenant avant la fin du transfert d'un fichier imprimante et donc sa fermeture, le fichier n'existe pas.
Tous les fichiers imprimante communiqués par le système hôte sont donc rangés dans la file d'attente et sans répétition, même partielle.
La seule exception à cette règle est un redémarrage à froid de CP/67 après une panne. Dans ce cas tous les fichiers contenus dans le spool de CP sont perdus. Notons cependant que les fichiers imprimante sont rangés dans la file d'attente par le serveur dès qu'ils sont disponibles dans le spool. Le risque de perte est donc très faible. Avec la première version ce risque était plus grand : les fichiers restaient dans le spool jusqu'à la connexion de l'imprimante destinataire.

Panne pendant l'envoi sur le réseau

La ST prévient le serveur en cas de panne distante. Celui-ci doit alors supprimer de la file d'attente les lignes déjà transmises du fichier en cours à l'exception des dernières. Il adresse à la gestion de la file (FA) d'attente une requête ABORT* qui provoque le maintien du fichier dans FA en supprimant tous les enregistrements lus sauf ceux de la piste en cours et de la piste précédente. Ceci représente au moins 20 lignes d'imprimante, ce qui est supérieur au nombre de lignes en cours d'expédition.
Quand l'imprimante distante se reconnecte, le même fichier lui est donc envoyé, sans les lignes qui avaient été transmises avant la panne.
En cas de panne locale, le comportement du serveur est différent. Il n'est pas prévenu de la panne et son travail s'interromp brutalement. Il ne peut donc adresser aucune requête à la gestion de la file d'attente et le fichier qu'il lisait reste intact, depuis le début. Il redémarrera donc en retransmettant la totalité du fichier dès que l'imprimante sera reconnecté. Ceci vient du fait que les enregistrements lus ne sont pas supprimés de la file d'attente au fur et à mesure de leur lecture. Il faudrait, pour les supprimer, faire une opération d'entrée-sortie supplémentaire sur le disque à chaque fois qu'une piste est entièrement lue. Ceci a été jugé trop coûteux.

3.3.2. Lecteur de cartes

Panne pendant le rangement dans le file d'attente.

Une panne distante ou locale peut interrompre le rangement des cartes reçues du réseau dans la file d'attente.
Sur détection de panne distante, le serveur adresse à la gestion de la file d'attente une requête ABORT pour supprimer le fichier en cours d'écriture. Quand le lecteur distant se reconnectera il retransmettra tout le fichier.
Comme les lignes d'imprimantes venant du système hôte, les cartes venant du réseau sont rangées au fur et à mesure de leur réception dans la file d'attente. Mais le directory sur disque ne contient un fichier en écriture

---

*Ou CLOSE qui provoque le maintien du fichier dans FA. A la reconnexion, le fichier sera donc intégralement réémis.

que lorsqu'il est entièrement écrit. Ainsi après une panne locale survenant alors que le fichier n'est pas entièrement reçu, donc écrit, le fichier ne figurera pas dans la file d'attente et devra être réémis par le lecteur distant.

Panne pendant l'envoi au système hôte

Les fichiers de type cartes en cours de lecture dans la file d'attente restent sur le disque jusqu'à ce qu'une requête ABORT soit adressée à FA. En cas de panne avant la fin du transfert du fichier, donc avant une requête ABORT, le fichier pourra être retransmis intégralement.
Pour ASP, un fichier qu'il reçoit et qui n'a pas été fermé n'existe pas. La panne ne communiquera donc pas à ASP un fichier incomplet.
Le spool de CP ne se comporte pas de la même façon. Il simule des appareils réels. Dans le cas qui nous intéresse, il simule un perforateur de cartes. Si, pendant la perforation de cartes, une panne se produit, les cartes perforées sur un perforateur réel sont bien perforées. C'est pour cette raison qu'après une panne pendant l'utilisation du perforateur virtuel, les cartes déjà passées dans le spool y restent et constituent un fichier même si le fichier en cours de perforation n'a pas été fermé. Cet inconvénient pour le serveur CP ne peut pas être éliminé sans changer la philosophie de CP/67.
Après une panne pendant le transfert d'un fichier de cartes de la file d'attente au spool, le spool contient donc un fichier représentant les cartes déjà perforées au moment de la cassure. Après le redémarrage, le fichier est toujours présent dans la file d'attente et sera retransmis au spool qui contiendra alors deux fichiers : le premier incomplet, le second complet. L'utilisateur destinataire de ces cartes pourra lire les deux fichiers dans sa machine virtuelle et supprimer le premier.
Notons qu'un seul fichier cartes peut subir ce traitement puisque le serveur ne dispose que d'un perforateur virtuel. Si le serveur n'utilisait pas la file d'attente, il lui faudrait plusieurs lecteurs virtuels (un par lecteur distant connecté) et une telle panne pourrait provoquer le même phénomène pour tous les lecteurs virtuels, donc pour tous les fichiers cartes en cours de transfert depuis les lecteurs distants.

4. IMPLÉMENTATION DES SERVEURS

L'implémentation du serveur sous ASP/OS-MVT est décrite dans [11], ainsi que l'implémentation de sa file d'attente.
La file d'attente sous OS a été réalisée d'après les spécifications qui ont été déduites de l'étude précédente. Cette réalisation a servi de base à l'implémentation de la file d'attente pour le serveur CP. Elle ne sera pas décrite en détail ici. La seule différence avec la file d'attente sous OS est la façon de lancer les opérations d'entrée-sortie sur disque.
Sous CP la file d'attente est entièrement gérée par la machine virtuelle du serveur sur un mini-disque CP. Les opérations d'entrée-sortie, exécutées par le superviseur d'entrée-sortie de SYNCOP, se font au niveau SIO.
Sous OS ces opérations sont confiées par le superviseur d'entrée-sortie de SYNCOP ou TELCOM à OS par des macro-instructions EXCP. La file d'attente est un data set OS.
L'implémentation du serveur CP est décrite en annexe.

5. CONCLUSION

5.1. Résultats obtenus

Le serveur sous OS se comporte bien comme nous l'avons décrit dans le paragraphe 2 (à une exception près : b). Sous CP, les buts définis n'ont pas été totalement atteints :

a) On peut perdre des fichiers imprimante après un redémarrage à froid de CP. Ce risque est très faible, mais il existe.

b) Dans le cas d'une panne locale pendant l'envoi d'une liste sur le réseau, après redémarrage le serveur réémet la liste depuis le début.

c) Sous CP, un paquet de cartes peut être reçu en deux fichiers, le deuxième étant complet.

Le cas b n'est pas très gênant : aucune information n'est perdue. Il en est de même pour le cas c. Dans le cas a, par contre, il y a perte d'information. Cette perte est dûe au spool de CP. Pour éviter ce problème, on ne peut envisager d'utiliser d'autre moyen de communication entre la machine virtuelle du serveur et les machines virtuelles des utilisateurs pour transmettre les cartes et les listes. Les autres moyens de communication (cf chapitre suivant) nécessiteraient l'utilisation d'un logiciel spécial et son acaptation dans chaque machine virtuelle qui voudrait envoyer des listes sur le réseau ou recevoir des cartes, alors que le spool peut être accédé avec le logiciel standard qu'utilise toute machine virtuelle (commande XFER de CP et commandes CMS pour les lectures et écritures dans le spool).
Quelle que soit la panne, la reprise est assurée par le serveur. Le client n'a aucune autre action à exécuter que de se reconnecter au serveur.

5.2. Résultats annexes

L'utilisation de la file d'attente permet d'offrir un service plus complet pour les imprimantes connectées. L'utilisateur d'une imprimante peut demander de suspendre le transfert d'une liste en cours de sortie et de supprimer cette liste. Il peut également demander à recevoir plusieurs copies d'une même liste.

La suppression d'une liste en cours de transfert vers l'imprimante distante se fait par une commande à la console opérateur de la machine virtuelle spécialisée (commande CANCEL). L'utilisateur, à partir de son terminal, peut demander à l'opérateur de CP d'envoyer cette commande.
Le fichier en cours de lecture dans la file d'attente n'est pas détruit. Ce cas est le seul où il faudrait détruire un fichier liste en cours de lecture. Pour ne pas alourdir inutilement la gestion de la file d'attente

(cette opération serait rare), le fichier est lu jusqu'à la fin puis est détruit, mais les enregistrements lus ne sont pas envoyés vers l'imprimante.

La demande de copies multiples se fait quand l'utilisateur passe la liste à répéter de sa machine virtuelle à la machine virtuelle spécialisée. La liste contient dans la première ligne l'adresse de la ST destinataire, le nom de la liste (facultatif) et le nombre de copies demandées (facultatif, par défaut 1).
Ces paramètres se retrouvent dans le descripteur du fichier dans la file d'attente et sont connus quand le fichier est ouvert en lecture pour être émis vers l'imprimante. S'il faut envoyer plusieurs copies, le fichier n'existe qu'en un exemplaire dans la file d'attente et quand il a été lu et envoyé vers l'imprimante, il est simplement fermé sans destruction. On peut donc à nouveau le lire et l'envoyer.
Notons que la file d'attente permet d'offrir sur le réseau un service que CP n'offre pas aux utilisateurs locaux. Pour sortir n copies d'une même liste sur une imprimante locale, un utilisateur doit produire n fois cette liste.

5.3. Conséquences sur les performances

Nous avons vu (Chapitre II) que les procédures de reprise au niveau de la ST conduisaient à des dégradations importantes des performances. Ce n'est pas le cas des procédures de reprises des serveurs.
Le principe de la file d'attente étant admis pour gérer plusieurs connexions simultanément, la gestion du disque a été conçue pour permettre les reprises après une panne. Mais aucun traitement coûteux n'est fait pour permettre ces reprises. Le nombre d'accès disque n'a pas été augmenté pour cela. On ne fait qu'un accès disque pour

- ouvrir un fichier en lecture (lecture du directory pour rechercher le fichier)

- lire un enregistrement
- écrire un enregistrement

On fait 4 accès disque pour fermer un fichier après écriture : il faut écrire l'enregistrement final, écrire la table d'allocation des pistes mises à jour, lire le directory pour y introduire le nouveau descripteur de fichier et enfin l'écrire.

On fait 3 accès disque pour supprimer un fichier lu : il faut écrire la table d'allocation mise à jour, lire le directory pour supprimer le descripteur du fichier et enfin écrire le directory.

On ne fait pas d'accès pour ouvrir un fichier en écriture, fermer un fichier lu, détruire un fichier en cours d'écriture.

Les opérations les plus fréquentes (lecture ou écriture d'un enregistrement) se font toujours en un seul accès. Les enregistrements étant en format PAV, chacune de ces opérations permet d'écrire ou de lire plusieurs lignes ou cartes.

CHAPITRE V

CONCLUSION ET PERSPECTIVES

1. RETOUR SUR SYNCOP

SYNCOP peut être comparé à un système d'exploitation multiprogrammé. Il en a les principales fonctions. Mais étant destiné à tourner dans une petite machine, une tâche utilisateur ou une machine virtuelle, il ne possède pas toutes les protections qu'on trouve dans un 'gros' système, ceci pour garantir des performances satisfaisantes.

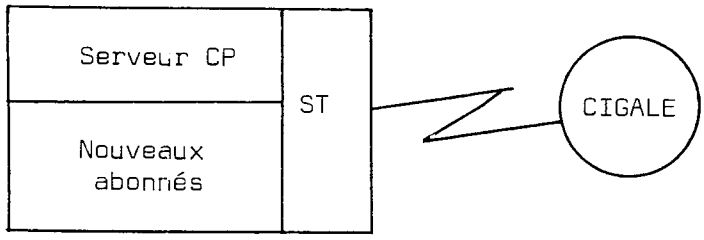
Il est cependant capable de déceler les erreurs des processus qu'il contrôle (interruptions programme) et de supprimer les processus qui provoquent ces erreurs. Parmi ces processus certains sont indispensables (par exemple, les processus de la ST et les processus permanents du serveur). Il n'est pas question de les supprimer. D'autres processus,au contraire, peuvent être supprimés sans inconvénient (de nouveaux abonnés en cours de test par exemple). Cette solution a été adoptée sous CP/67, elle sera utile sous OS et sur les petites machines (ORDOPROCESSEUR, MITRA15).
Si la suppression des processus provoquant une interruption programme permet d'obtenir une amélioration de la fiabilité, cette solution n'est pas suffisante. Un processus peut, par erreur, modifier une zone de mémoire utilisée par un autre processus. C'est ce dernier processus qui risque alors d'être détruit par la faute du premier. On cherchera donc un moyen de séparer les processus peu fiables des processus bien testés.
Sous CP/67 ceci peut être obtenu, dans le cas des applications réseau, en mettant dans des machines virtuelles différentes la ST et les abonnés d'exploitation d'une part et les abonnés en cours de développement d'autre part.

Ceci permet de développer de nouveaux services sans réduire les horaires d'exploitation et sans risquer de perturber les services en exploitation.

2. SÉPARATION DES ABONNÉS ET DE LA ST

2.1. L'organisation initiale

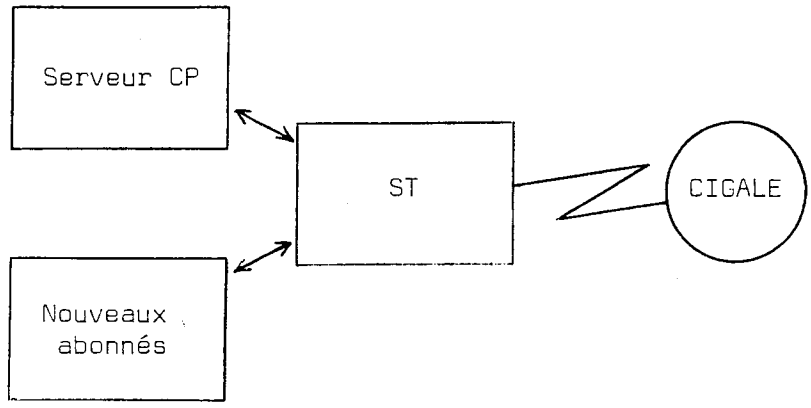
Nous avons décrit l'implémentation du serveur et de la station de transport dans une seule machine virtuelle. L'adjonction de nouveaux abonnés dans la même machine virtuelle mènera à la structure suivante :



Le serveur, la ST et les nouveaux abonnés sont des processus contrôlés par SYNCOP.

2.2. La séparation dans plusieurs M.V.

Si un mécanisme de communications entre machines virtuelles est disponible sous CP/67, on peut séparer la ST des abonnés et les abonnés entre eux dans plusieurs machines virtuelles :



Cette structure est l'une des structures possibles. Un souci d'amélioration des performances peut nous conduire à une organisation différente. Il peut être préférable de garder certains abonnés dans la même machine virtuelle que la ST. Il est aussi possible de répartir le serveur dans deux machines différentes : les entrées terminal, INIT et DOC d'un côté, les entrées lecteur de cartes et imprimante de l'autre, avec la file d'attente.
Deux mécanismes seront disponibles pour communiquer entre machines virtuelles :

- Un mécanisme de communication par la mémoire dont la maquette est décrire dans [10] .
- Un mécanisme matériel se présentant comme un adaptateur de canal à canal.

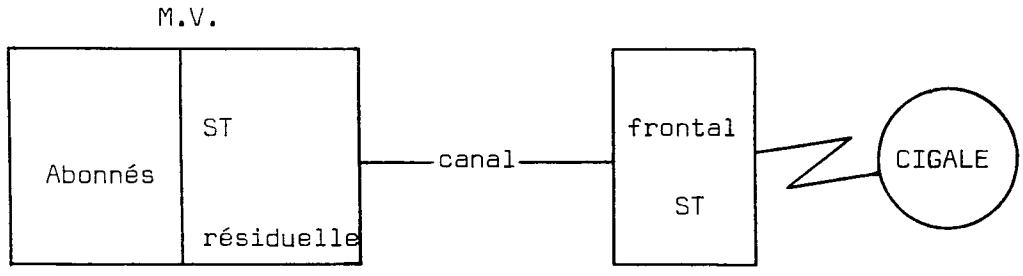
On choisira l'un ou l'autre de ces mécanismes suivant leur fiabilité.
Cette organisation permettra à toute machine virtuelle d'accéder au service de transport.

2.3. Le frontal ST

Un projet de réalisation d'une ST par des microprocesseurs est en cours dans l'équipe 'Architecture d'Ordinateur' de l'ENSIMAG [17]. Ce frontal ser constitué de trois microprocesseurs, un pour chaque fonction :

- gestion de la ligne vers CIGALE (procédure TMM-UCIG)
- service de transport, sauf fragmentation-réassemblage
- communication avec le 360 par le canal.

Une ST résiduelle sera implémentée dans le 360 pour permettre l'accès des abonnés à la ST et pour assurer la fragmentation et le réassemblage qui ne seront pas faits dans le frontal, faute de mémoire.
La structure des services réseau sera alors la suivante :



Cette structure est logiquement équivalente à la structure précédente où on peut considérer la M.V. ST comme un frontal.
Il est possible de séparer les abonnés entre eux en utilisant les mécanismes de communication entre machines virtuelles.
La solution du frontal présente l'intérêt de séparer la ST des abonnés tout en utilisant un moyen de communication fiable, comme dans la solution précédente. De plus le 360 se trouve déchargé de tout le travail en temps réel (gestion de la ligne et time-outs de transport) ainsi que d'une partie importante du service de transport. La ligne est gérée avec une procédure full-duplex, plus efficace que la procédure BSC utilisée actuellement à cause des possibilités du matériel de transmission disponible sur le 360. Les performances doivent s'en trouver améliorées.
Une extension envisageable du frontal est de le raccorder par deux lignes différentes à deux noeuds de CIGALE. Ainsi les pannes d'un noeud ne compromettront plus la connexion du 360 au réseau.

3. LES ORIGINES DES PANNES

Cette étude nous a conduits à porter une grande attention aux pannes du réseau concernant les serveurs du 360.

3.1. Pannes locales

Les pannes locales sont les plus faciles à céceler et peuvent être classées en trois catégories :

a/ Les pannes matérielles du 360 et les erreurs des systèmes hôtes sont les plus fréquentes. Nous n'avons dans ce domaine que peu de moyens d'action.

b/ Les pannes des logiciels réseau du 360 (ST, serveurs) sont plus rares. Sous CP, l'exploitation quotidienne (8 heures par jour) de ce logiciel, n'est perturbée qu'une fois toutes les 3 semaines environ par ce type de pannes. Ce chiffre s'applique à la première version. La seconde version, qui doit avoir une fiabilité accrue, n'est pas encore en exploitation, l'ensemble du réseau n'utilisant pas encore les nouveaux protocoles. Les quelques essais déjà réalisés ne permettent pas de donner de chiffre significatif.

c/ Le matériel de transmission vers le noeud (ligne, modems) présente une très bonne fiabilité. Les pannes sont exceptionnelles. Ceci provient sans doute en partie de la faible longueur de la ligne (quelques mètres).

3.2. Les pannes distantes

Au début de l'exploitation les pannes distantes ont été les plus fréquentes.

a/ Les pannes de CIGALE et plus particulièrement du noeud auquel est connecté le 360 provenaient d'un manque de tests de la procédure BSC. Cette procédure n'était utilisée que pour le 360 et, étant moins exploitée que la procédure TMM-UCIG, elle a été moins rapidement mise au point.
Tous les noeuds de CIGALE sont téléchargés par un centre de contrôle unique pour le réseau. En cas de panne de ce centre de contrôle, tout le réseau se trouve paralysé. Ces pannes, bien que très rares, ont des conséquences catastrophiques. Une bonne solution pourrait être de disposer d'un centre de contrôle de secours ou à défaut de prévoir la possibilité de charger chaque noeud localement en cas de panne du centre de contrôle.

b/ Les pannes des concentrateurs de terminaux ont été les plus gênantes. Ces concentrateurs sont les points d'accès du réseau qu'utilisent tous les usagers. Leurs pannes, très fréquentes au début de l'exploitation du réseau, ont été immédiatement perçues par les utilisateurs. Cette expérience montre qu'il est très important de développer au maximum la fiabilité de ces concentrateurs pour attirer les utilisateurs vers le réseau et ne pas les décourager.
Il est également important de permettre à chaque utilisateur d'un terminal du réseau de localiser les pannes qui interrompent son travail. Le premier outil à mettre à sa disposition serait une commande envoyant des ECHO. L'utilisateur pourrait ainsi tester les différents composants qu'il utilise, et ainsi se renseigner directement sur les conditions de redémarrage de l'élément défaillant.
Les concentrateurs sont réalisés sur de petites machines qui ne sont pas surveillées en permanence. En cas de panne , l'intervention n'est pas immédiate. On pourrait leur appliquer la même solution que pour les noeuds de CIGALE: les noeuds sont surveillés en permanence à partir du centre de contrôle du réseau. Les pannes sont immédiatement détectées et, s'il

ne s'agit pas de pannes matérielles, les noeuds sont rechargés par le centre de contrôle. Ainsi l'indisponibilité d'un noeud est très brève. Cette solution conviendrait très bien aux concentrateurs de terminaux.

4. GÉNÉRALISATION

4.1. Application à d'autres réseaux

Toute cette étude est axée sur le réseau CYCLADES. Elle est cependant assez générale pour être applicable à d'autres réseaux.
Le 360/67 doit être connecté sous CP/67 au réseau européen EIN. Ce réseau possède la même architecture que CYCLADES : un sous-réseau de communication et des centres participants connectés à ce sous-réseau par l'intermédiaire d'une station de transport. Les protocoles ne sont pas identiques, mais très voisins.La plupart des réseaux en cours de développement ou prévus reprennent cette architecture et des protocoles semblables.
La connexion à EIN se fera de la même façon que la connexion à CYCLADES. On adaptera seulement la ST au protocole de transport EIN et le serveur de terminaux au protocole appareil virtuel. On utilisera SYNCOP, puis les mécanismes de communication entre machine virtuelles.
L'analyse qui a été faite au chapitre II s'applique également au cas de EIN. On retrouve dans le protocole de transport les caractéristiques nécessaires pour assurer une bonne fiabilité : le contrôle d'erreur et la possibilité de détecter dans tous les cas les pannes distantes.
Il est probable que, quand des applications utiliseront des lecteurs de cartes ou des imprimantes, la solution de la file d'attente pourra être adoptée.

4.2. Généralisation à d'autres applications

Les applications décrites ici sont les premières applications développées sur tous les réseaux : accès à un service interactif ou à un service de traitement par lots. Ces applications ne sont pas fondamentalement nouvelles, l'accès distant à de tels services pouvant se faire sans un réseau de ce type. Dans ces applications le réseau sert seulement à remplacer plusieurs lignes de transmissions entre les terminaux et les services.

Des applications typiques des réseaux sont en cours d'étude : bases de données réparties, langages de commande réseau ...
Pour ces applications l'outil réseau est indispensable pour avoir accès simultanément à plusieurs sites. Dans ces applications d'un type nouveau, les problèmes de fiabilité pourront se poser de façon différente. Notons toutefois qu'ils pourront être résolus au niveau de l'application elle-même, le réseau offrant les outils nécessaires dans le protocole de transport: les pannes distantes pouvant être décelées, l'application pourra prévoir les moyens de réagir.

Les outils présentés ici pourront être utilisés pour développer de nouvelles applications dans de bonnes conditions de fiabilité. Par exemple, l'interpréteur destiné au langage de commande réseau, en cours d'implémentation sous CP/67, utilisera SYNCOP et les mécanismes de communication entre machines virtuelles.

ANNEXES

A N N E X E I

ALLOCATION MEMOIRE

CALCUL DE LA TAILLE OPTIMALE DES CELLULES

Soient :

T la taille totale de la mémoire libre (table d'allocation et cellules)

t la taille d'une cellule

b la taille moyenne des blocs demandés

Une cellule demande t octets, plus un pour la table d'allocation. Le nombre total de cellules est donc T/t+1.

La table d'allocation, qui comporte un octet par cellule, a donc pour taille: T/t+1.

La taille de la zone des cellules est la taille totale moins la taille de la table d'allocation :

$$T(1 - \frac{1}{t+1}) = \frac{Tt}{t+1}$$

Pour chaque bloc alloué, on perd entre 0 (si la taille du bloc est un multiple de t) et t-1 octets. La perte moyenne sur un bloc est donc $\frac{t-1}{2}$ (la distribution des tailles des blocs est uniforme).

La taille prise par un bloc est en moyenne $b + \frac{t-1}{2}$ (la taille du bloc plus la place perdue) = $\frac{2b + t-1}{2}$

Quand la mémoire est pleine, le nombre de blocs est :

$$\frac{Tt}{t+1} \times \frac{2}{2b + t-1}$$ (taille de la zone des cellules/taille d'un bloc)

La perte sur l'ensemble des blocs est : $\frac{t-1}{2} \times \frac{Tt}{t+1} \times \frac{2}{2b+t-1}$

La perte totale est la perte sur l'ensemble des blocs plus la taille de la table d'allocation : $T \frac{t(t-1) + 2b + t-1}{(t+1)(2b+t-1)}$

Le rapport de la place perdue à la place totale est :

$$R(t) = \frac{t^2 + 2b - 1}{t^2 + 2bt + 2b - 1}$$

C'est ce rapport qu'il faut minimiser en choisissant la taille des cellules (t). La dérivée de R(t) est :

$$R'(t) = 2b(t^2 - 2b + 1)$$

Elle s'annule pour $t = \sqrt{2b-1}$
R(t) décroît quand t augmente jusqu'à la valeur $\sqrt{2b-1}$, puis croît. La valeur optimale de t est donc :

$$t = \sqrt{2b-1}$$

t doit être un multiple de 8 pour assurer que chaque cellule est alignée sur une frontière de double-mot.
Une cellule de 8 octets est l'optimum pour des blocs de 32 octets
" 16 octets " " " 128 octets
" 24 octets " " " 288 octets

La valeur la plus proche de la taille moyenne des blocs utilisés dans les applications sous SYNCOP est 128 octets. Nous avons donc choisi une taille de cellules de 16 octets.

ANNEXE II

IMPLEMENTATION DU SERVEUR CP - 67

1. ARCHITECTURE DU SERVEUR CP

Le serveur, pour l'entrée terminal, reprend la même organisation que dans la première version [10]. Il a été adapté à SYNCOP, à la ST2 [22] et au nouveau protocole protocole appareil virtuel [7].

Pour les entrées lecteur de cartes et imprimante, il a été entièrement reconçu à partir de l'étude du chapitre IV.
Le serveur est constitué d'un ensemble de processus gérés par SYNCOP.
L'ensemble SYNCOP + ST2 + Serveur est chargé initialement dans la machine virtuelle spécialisée par NESTOR [23] .

Un processus de contact (PCONT) est chargé de recevoir toutes les demandes de connexions venant du réseau. Pour chaque demande, il crèe un processus chargé de gérer la connexion.

- Pour chaque connexion à une entrée terminal il crèe un processus PITPAV qui crèe à son tour un processus PITCP.
- Pour chaque connexion à l'entrée lecteur de cartes, il crèe un processus SERVRDR.
- Pour chaque connexion à l'entrée imprimante, il crèe un processus SERVPTR.

Trois autres processus sont créés initialement :

- Un processus FA qui gère la file d'attente
- Un processus READER qui fait les transferts des listes entre le spool de CP et la file d'attente par l'intermédiaire d'un lecteur virtuel
- Un processus PUNCHER qui fait les transferts des cartes entre la file d'attente et le spool de CP par l'intermédiaire d'un perforateur virtuel.

2. LE PROCESSUS PCONT

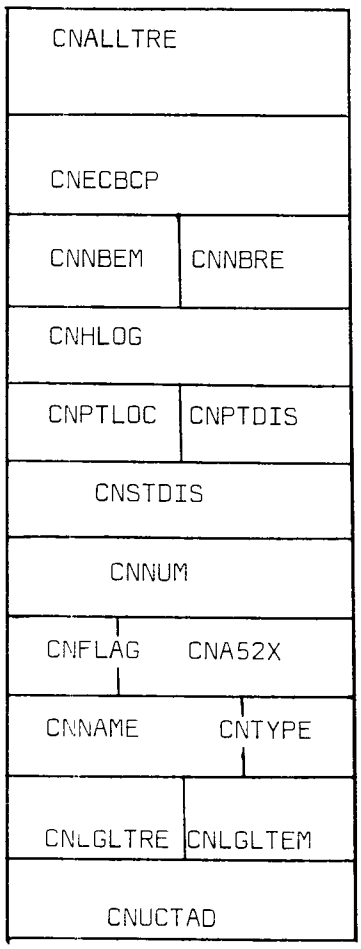
Le processus PCONT créé à l'initialisation, ouvre 5 portes : une pour chacune des entrées du serveur :

- entrée terminal (TERM)
- entrée INIT
- entrée DOC
- entrée lecteur de cartes (RDR)
- entrée imprimante (PTR)

Les entrées INIT et DOC sont des cas particuliers de l'entrée TERM. Une fois connecté à INIT, l'utilisateur reçoit sur son terminal un mode d'emploi du serveur CP, sans aucune action de sa part. L'entrée DOC procède de même pour envoyer une liste des documentations disponibles. Une fois cette liste reçue, l'utilisateur peut demander de lister sur son terminal ou son imprimante ces documentations. Une seule connexion est possible sur chacune des portes INIT ou DOC.
Une fois les 5 portes ouvertes, PCONT attend les demandes de connexion sur ces portes. Dès qu'une demande arrive (FL-INIT), PCONT la traite. Il vérifie d'abord qu'il s'agit bien d'une demande avec contrôle de flux et contrôle d'erreur. Sinon il refuse la demande et attend la suivante. Il cherche une entrée libre dans la table des connexions. Le nombre de connexions simultanées est limité et une table (CNTBLT dans le module AREA) contient autant d'entrées (CNTBL) que de connexions possibles.

PCONT cherche un CNTBL libre et vérifie en même temps qu'il n'y a pas déjà une connexion identique (même porte locale, même porte distante) ou, s'il s'agit de INIT ou DOC, qu'il n'y a pas déjà une connexion sur cette entrée. S'il n'y a plus de place ou qu'il existe déjà une connexion identique, la demande de connexion est refusée et PCONT attend la demande suivante.
Si la connexion est possible, PCONT initialise le CNTBL trouvé avec les champs : CNPTLOC, CNPTDIS, CNSDIS, CNNUM, CNTYPE, CNLGLTEM, CNLGLTRE.

Structure du CNTBL



Adresse de la liste des lignes reçues destinées à PITCP ou nombre de copies pour une liste (entrée imprimante).

ECB d'attente de PITCP

Compteur d'émission/compteur de réception

Heure de connexion

Identificateurs porte locale/porte distante (PT-ID)

Adresse de la ST distante

Numéro de la connexion

Flag/adresse de l'extension de l'UCT

Nom du 1052X/type de connexion (TERM,INIT,DOC,RDR, PTR)

Longueur des lettres reçues/émises

Adresse de l'UCT du 1052X associé (Initialisé au début de PCONT).

PCONT crèe ensuite un processus PITPAV en lui communiquant l'adresse du CNTBL s'il s'agit des types TERM, INIT ou DOC, ou un processus SERVRDR pour le type RDR ou SERVPTR pour le type PTR. Il envoie ensuite un message à la console opérateur signalant la demande de connexion et attend d'autres demandes de connexion.

3. LE PROCESSUS PITPAV

PITPAV est chargé de l'ouverture du flot et de la réception sur le flot. Il commence par demander à la ST l'ouverture du flot avec le terminal qui a fait la demande. Si cette ouverture échoue, le CNTBL est libéré et PITPAV se supprime (EXIT) après avoir envoyé un message à la console opérateur. Si le flot est ouvert correctement, un message est envoyé à la console opérateur puis un processus PITCP est créé. PITPAV attend la fin de son initialisation.

S'il s'agit d'une connexion de type INIT ou DOC, PITPAV prépare le login de la machine virtuelle correspondante. Pour cela, il passe une attention à PITCP (indicateur ATTN dans l'extension de l'UCT et post de l'ECPB CNECBCP) puis met dans la liste d'entrée de PITCP (pointée par CNALLTRE) la commande login, le mot de passe, les commandes ipl cms et guide. L'exécution de la commande guide (commande EXEC) par la machine virtuelle INIT ou DOC fera sortir le manuel d'utilisation ou la liste de la documentation.

PITPAV demande alors à la ST de retirer les télégrammes et les lettres reçues sur le flot et attend une lettre ou un télégramme. Sur réception d'un télégramme PITPAV ne fait rien sauf si ce télégramme indique que le flot est fermé (voir plus bas).

Les lettres reçues doivent avoir le format PAV. Elles sont analysées. S'il s'agit d'une fonction (message de contrôle du dialogue), elle est traitée comme une attention et signalée à PITCP (indicateur ATTN dans l'extension de l'UCT et post de CNECBCP).

S'il s'agit d'un message de contrôle d'application, les lignes de texte qu'il contient sont enfilées dans la liste d'entrée de PITCP, ligne par ligne. Si le message contient A-VOUS, PITCP en est prévenu (indicateur A-VOUS remis à zéro dans l'extension de l'UCT et post de CNECBCP). Les autres messages reçus sont ignorés.

Une fois la lettre ou le télégramme traité, PITPAV attend la lettre ou le télégramme suivant sur le flot.

Si, au moment d'un retrait de lettre ou de télégramme, la ST signale que le flot est fermé, PITPAV le signale à PITCP (indicateur DISC1 dans l'extension de l'UCT et post de CNECBCP) et se supprime (EXIT).

4. Le processus PITCP

PITCP, créé par PITPAV est chargé de gérer le 1052X [24] et d'émettre sur le flot. Il crée sa liste d'entrée et complète le CNTBL : CNA52X,CNNAME,CNALLTRE CNHLOG. Ayant fini son initialisation, il réveille PITPAV et envoie une attention sur le 1052X qu'il gère (DIAGNOSE ATTN).
Il attend alors :
- soit que le 1052X change d'état
- soit que PITPAV le réveille (CNECBCP)
- soit qu'il y ait une ligne dans sa liste d'entrée.

Changement d'état du 1052X

Si le 1052X est en lecture, PITCP cherche une ligne dans sa liste d'entrée. S'il y en a une il satisfait la lecture avec et attend un autre évènement. S'il n'y en a pas, il envoie la lettre A-VOUS, marque qu'il n'a plus la main et attend un autre évènement.
Si le 1052X est en écriture et si le serveur a la main, il analyse le buffer de sortie, le met en format PAV et l'envoie sur le flot. Il attend l'évènement suivant.

Réveil par PITPAV

Si PITPAV demande la connexion (DISC1), PITCP teste si le 1052X est dans l'état inactif (l'utilisateur a fait logout). S'il est actif, il fait une diagnose DISCONNECT pour forcer le logout. Il envoie ensuite un message de déconnexion à la console et se supprime (EXIT). Si PITPAV demande une attention (ATTN), PITCP la reflète au 1052X (DIAGNOSE ATTN) et attend l'évènement suivant.

Une ligne en entrée

Si le 1052X est en lecture, la lecture est satisfaite avec la ligne reçue et on attend un autre évènement. Sinon on attend seulement l'évènement suivant.

Rupture du flot

Si, au cours de l'envoi d'une lettre sur le flot, la ST signale que le flot est fermé, PITCP supprime le processus PITPAV associé et fait le même traitement que pour une demande de déconnexion venant de PITPAV.

5. LE PROCESSUS FA

Le processus FA gère le disque de la file d'attente. Il possède une liste en entrée dans laquelle les processus utilisant la file d'attente déposent leurs requêtes. Ces requêtes peuvent être de cinq types :

- OPEN : Ouverture d'un fichier. Précise le type du fichier (cartes ou listes), le mode d'accès (lecture ou écriture) et l'identification du destinataire (listes uniquement).
- WRITE : écriture d'un enregistrement
- READ : lecture d'un enregistrement
- CLOSE : fermeture d'un fichier avec destruction totale si le fichier est ouvert en écriture, des pistes lues s'il est ouvert en lecture.

6. LE PROCESSUS SERVRDR

Créé par PCONT, SERVRDR gère un flot recevant des cartes et range ces cartes dans la file d'attente.
SERVRDR demande d'abord à la ST l'ouverture d'un flot avec le lecteur de cartes distant. Si l'ouverture du flot échoue, un message prévient l'opérateur et SERVRDR se supprime (EXIT).
Le flot étant ouvert, un message le signale à l'opérateur et SERVRDR adresse à FA une requête OPEN pour un fichier cartes en écriture. Il attend alors les lettres et télégrammes arrivant sur le flot.
Les seuls télégrammes traités sont ceux qui signalent la fermeture du flot. (Voir plus loin leur traitement). Les autres sont ignorés et SERVRDR se remet en attente de réception.
Les lettres contenant des messages de contrôle d'application PAV sont écrites sans modification dans la file d'attente (WRITE). Ces lettres sont toutefois analysées pour détecter la fin du paquet de cartes (code nouvelle page). Si le paquet de cartes est entièrement reçu, le fichier en cours d'écriture dans la file d'attente est fermé (requête CLOSE) et SERVRDR ouvre un nouveau fichier (requête OPEN) pour recevoir le prochain paquet de cartes. Les autres lettres sont ignorées.
En cas de rupture du flot, signalée par la ST, le fichier en cours d'écriture dans la file d'attente est détruit (requête ABORT), un message de déconnexion est envoyé à la console, et le processus SERVRDR se supprime (EXIT).

7. LE PROCESSUS PUNCHER

Le processus PUNCHER fait passer les cartes de la file d'attente au spool de CP par l'intermédiaire du perforateur virtuel.
PUNCHER adresse à FA une requête OPEN pour un fichier de type cartes en

lecture et attend qu'un tel fichier soit disponible. Dès qu'un fichier est prêt, PUNCHER lit les enregistrements (requête READ), les décode et en extrait les cartes. La première carte doit être une carte ID standard. Elle contient le nom de la M.V. destinataire. PUNCHER ne la perfore pas, mais envoie à CP/67 une commande XFER pour lui communiquer le nom de la M.V. destinataire. Les autres cartes sont perforées sur le perforateur virtuel. Elles sont ainsi rangées dans le spool.
Quand FA signale la fin de fichier sur une requête READ, une commande de fermeture du perforateur virtuel est envoyée à CP et le fichier lu dans la file d'attente est détruit (requête ABORT). La M.V. destinataire du fichier est prévenue par CP qu'un fichier de cartes est prêt dans le spool.
PUNCHER adresse alors à FA une requête OPEN pour lire un nouveau fichier et reprend le même traitement.

8. Le processus READER

Le processus READER fait passer les listes du spool de CP à la file d'attente par l'intermédiaire du lecteur virtuel.
Il lance une lecture sur le lecteur virtuel. S'il n'y a rien à lire (unit check) il attend qu'un fichier soit disponible.
Il lit alors les enregistrements qui représentent des lignes d'imprimante. La première ligne indique la destination du fichier, son nom et le nombre de copies demandées (ligne //PTR) et permet d'ouvrir un fichier de type ligne en écriture dans la file d'attente (requête OPEN). Si la ligne //PTR est absente, READER envoie un message à l'opérateur, ferme le fichier du spool et attend le fichier suivant.
Les lignes suivantes sont analysées, mises au format PAV (messages de contrôle d'application) et écrites dans la file d'attente (requête WRITE).

Quand le fichier du lecteur virtuel est vide, on le ferme ainsi que le fichier de la file d'attente (requête CLOSE) et on envoie un message à l'opérateur indiquant qu'un fichier est prêt à être émis vers l'imprimante distante. On attend ensuite le fichier suivant sur le lecteur virtuel.

9. LE PROCESSUS SERVPTR

Créé par PCONT, le processus SERVPTR est chargé d'envoyer les listes qui sont dans la file d'attente vers l'imprimante distante.
SERVPTR ouvre d'abord le flot avec l'imprimante distante. Si cette ouverture échoue, l'opérateur est prévenu par un message et SERVPTR se supprime (EXIT).
Une fois le flot ouvert, SERVPTR adresse une requête OPEN à FA pour obtenir un fichier de type liste destiné à l'imprimante distante à laquelle il est connecté. Si aucun fichier n'est prêt, il attend l'un des deux évènements :

- un fichier prêt dans la file d'attente
- fermeture du flot

Dès qu'un fichier est prêt, FA communique à SERVPTR le nom du fichier et le nombre de copies demandées. Une ligne d'en-tête contenant le nom du fichier est envoyée sur le flot vers l'imprimante distante et le nombre de copies demandées est rangé dans le CNTBL (champ CNALLTRE).
SERVPTR lit alors les enregistrements de ce fichier dans la file d'attente (requête READ). Chaque enregistrement lu est envoyé sur le flot (il est déjà au format FAV), sauf si l'indicateur CNFLAG a été positionné par CONS. Dans ce cas (une commande CANCEL a été émise par l'opérateur), les enregistrements lus ne sont pas envoyés, jusqu'à la fin du fichier.

La fin du fichier est signalée par FA sur une requête READ. Si l'indicateur CNFLAG est positionné, une requête ABORT est envoyée à FA pour supprimer le fichier lu et SERVPTR demande à FA un autre fichier (requête OPEN). L'indicateur CNFLAG a été supprimé.
Si on a déjà envoyé le nombre de copies demandées, le traitement est le même.
S'il reste des copies à envoyer, le compteur de copies (CNALLTRE) est décrémenté et une requête CLOSE est envoyée à FA. Le fichier qu'on vient de lire reste donc dans la file d'attente et une requête OPEN permet d'obtenir à nouveau ce fichier. Le compteur de copies ne sera pas réinitialisé quand le fichier sera prêt.

En cas de fermeture du flot (détectée au moment d'une émission ou pendant l'attente d'un fichier de FA), la partie déjà lue du fichier (sauf la piste courante et la précédente) est supprimée (requête ABORT). Si on était en attente de fichier ABORT annule seulement la demande d'ouverture.
Un message de déconnexion est envoyé à la console opérateur et le processus SERVPTR se supprime (EXIT).

10. Le processus CONS

Le processus CONS gère la console opérateur de la machine virtuelle. Il est créé à l'initialisation du système.
Il crèe d'abord une liste dans laquelle les différents processus pourront mettre les messages qu'ils veulent sortir à la console. Puis il attend que l'opérateur provoque une attention ou qu'un message arrive dans la liste.
S'il y a un message dans la liste, il l'écrit à la console, libère la zone mémoire qu'il occupait et attend à nouveau.
Si l'opérateur provoque une attention, il lance une lecture sur la console et attend la fin. La commande de l'opérateur est alors analysée et traitée (voir la liste des commandes) et CONS se remet en attente.

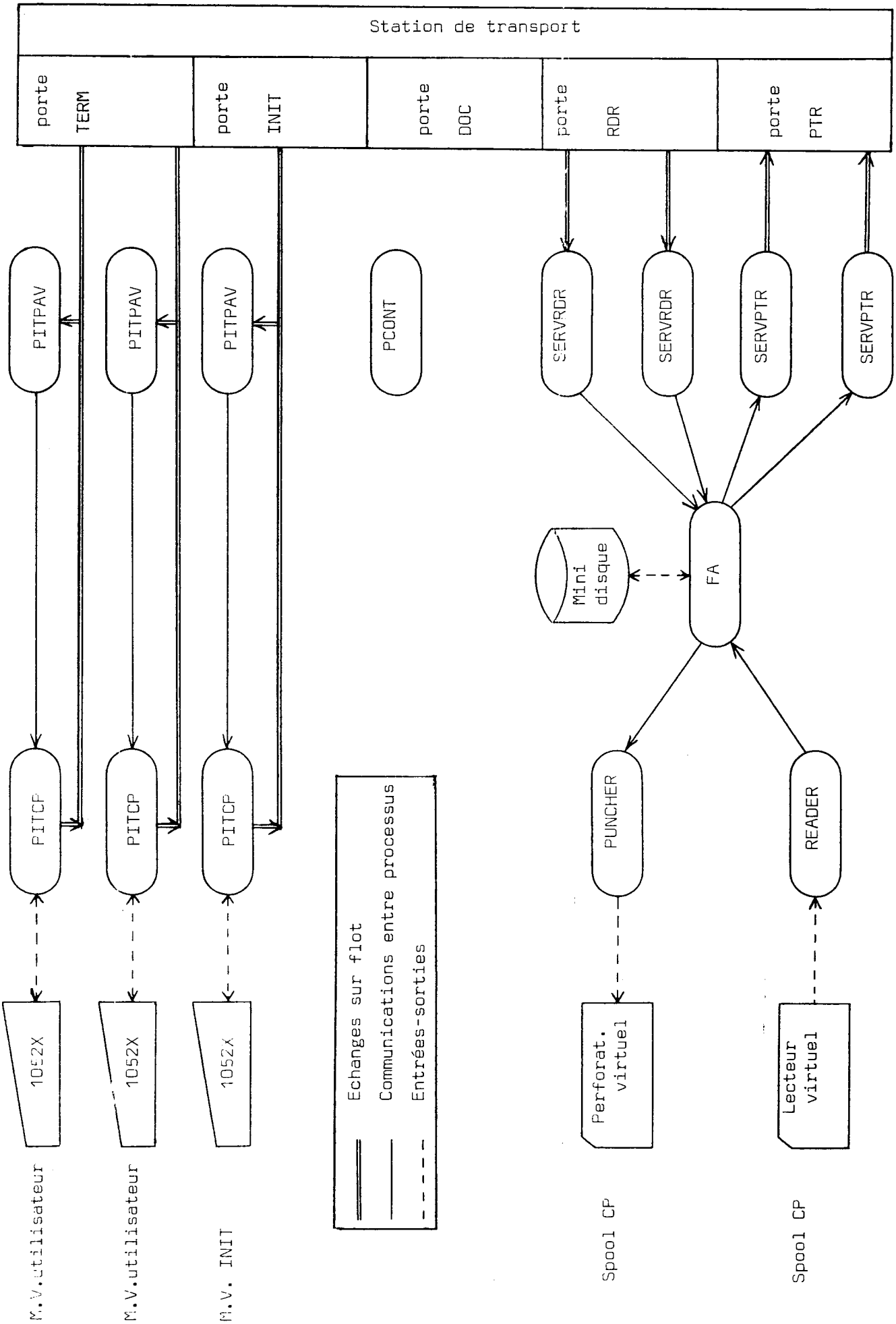


Schéma du serveur CP

A N N E X E III

MESSAGES ENVOYES A LA CONSOLE PAR LE SERVEUR CP

1. RÉCEPTION D'UNE DEMANDE DE CONNEXION

EEEENNN : ppppssss

EEEE : nom de l'entrée (TERM, INIT, DOC, RDR, PTR)

NNN : numéro de la connexion (décimal)

pppp : identificateur de la porte distante (hexadécimal)

ssss : adresse de la ST distante (hexadécimal)

2. ETABLISSEMENT D'UNE CONNEXION

EEEENNN:CONNEXION ETABLIE[,UNIT=aaa]

aaa:adresse du 1052X associé à la connexion. Uniquement pour TERM, INIT et DOC. Permet, par demande à CP, de connaître la machine virtuelle utilisée à travers la connexion.

3. ECHEC A L'ÉTABLISSEMENT D'UNE CONNEXION

EEEENNN : ECHEC CONNEXION

4. FERMETURE D'UNE CONNEXION

EEEENNN : CONNEXION FERMEE

5. STATISTIQUES D'EXPLOITATION D'UNE CONNEXION APRÈS SA FERMETURE

{TERM}
{INIT} NNN:UNIT=aaa,MINUTES=mmmm,EM=eeee,REC=rrrr
{DOC }

mmmm : durée de la connexion en minutes

eeee : nombre de lignes émises

rrrr : nombre de lignes reçues

RDRNNNN : MINUTES=mmmm,RE=rrrr,FI=ffff

rrrr : nombre de lettres reçues sur la connexion RDR

ffff : nombre de fichiers cartes reçus

PTRNNNN : MINUTES=mmmm,EM=eeee,FI=ffff

eeee : nombre de lettres émises sur la connexion PTR

ffff : nombre de listes émises

6. RÉCEPTION DE LISTES D'UNE AUTRE MACHINE VIRTUELLE

Chaque fois qu'une machine virtuelle donne au serveur une liste à émettre sur le réseau, l'un des deux messages suivants sort à la console :

FICHIER PTR POUR ssss

Le fichier est pris en compte. ssss est l'adresse de la ST destinataire. Le fichier sera émis dès qu'une imprimante de cette ST sera connectée.

CARTE //PTR ILLEGALE

La carte //PTR est omise ou n'a pas le bon format. Le fichier est perdu.

A N N E X E IV

COMMANDES OPERATEUR

MISE AU POINT

- Commande : DUMP
  Réponse : **OK**
  Provoque la sortie, sur une imprimante locale, d'un dump des 16 registres généraux et de la mémoire de la machine virtuelle spécialisée.

- Commande : CP message
  Réponse : réponse de CP et **OK**
  Permet de passer une commande à CP et d'obtenir la réponse.

- Commande : FDUMP
  Réponse : **OK**
  Provoque la sortie, sur une imprimante locale, du dump du F-espace contenant les snaps.

SYNCOP

- Commande : CORE
  Réponse : n OCTETS JAMAIS UTILISES
  Donne la taille de la mémoire gérée par SYNCOP qui n'a jamais été utilisée depuis le début de la session.

- Commande : SIZE

  Réponse : n TROUS, m OCTETS LIBRES, TROU MAX = p

  Permet de connaître la fragmentation de la mémoire :

  nombre de trous (n)

  taille totale des trous (m)

  taille du plus grand trou (p)

LIGNE VERS CIGALE

- Commande : START LMTR

  Réponse : **OK**

  Active la ligne

- Commande : STOP LMTR

  Réponse : **OK**

  Désactive la ligne

- Commande : HISTORI

  Réponse : **OK**

  Permet de passer en mode historique : tous les paquets émis et reçus sont 'snapés' dans le F-espace.

- Commande : NHIST

  Réponse : **OK**

  Permet de quitter le mode historique.

- Commande : STAT LMTR

  Réponse :

TRANSMISSIONS	n	COMMAND REJECT	n
INT/VENTION REQ	n	BUS - OUT CHECK	n
EQUIPEMENT CHEC	n	DATA CHECK	n
OVERRUN	n	LOST DATA	n
TIME-OUT	n	UNIT EXCEPTION	n

  Donne des statistiques sur la ligne depuis le début de la session.

Réponses : FA WARM START
ou FA COLD START
ou FORMATTING FA
ou FA ALREADY STARTED
Permet de démarrer la file d'attente sur disque
en la formattant ($FORMAT*)
en supprimant tous les fichiers (%COLD*)
en conservant tous les fichiers (WARM)

- Commande : CANCEL pt-id/st-id
Réponse : **OK**
Permet de supprimer un fichier imprimante en cours de transfert
vers la porte pt-id de la ST st-id.

ANNEXE V

PERFORMANCES DE SYNCOP

Les chiffres donnés ici ont été calculés à partir du temps d'exécution des instructions sur le 360/67. Pour ces calculs certaines hypothèses ont été faites :

- 40 processus répartis également sur 4 niveaux de priorité
- les fréquences des évènements réalisés et non réalisés lors des attentes sont les mêmes
- l'attente multiple concerne un évènement parmi 4.
- quand on poste un évènement, il a autant de chances d'être déjà attendu que de ne pas l'être
- la mémoire comporte 10 trous, le cinquième dans l'ordre des adresses croissantes peut satisfaire une demande de mémoire
- quand une demande de réveil ou d'arrêt d'un réveil est émise par un processus, il y a déjà 20 demandes en cours
- les unités d'entrée-sortie gérées par SYNCOP sont au nombre de 16
- quand un processus enfile un élément dans une liste, la liste peut aussi bien être vide que non vide, avec la même probabilité
- quand un processus retire un élément d'une liste cette liste peut aussi bien être pleine que non pleine, avec la même probabilité.

Ces chiffres sont applicables à un 360/67 réel. Sous CP/67, il faut tenir compte du temps pris par les instructions privilégiées qui sont simulées (SVC, SSM, SIO, LPSW). Il faut notamment ajouter pour la plupart des services les temps des instructions qui masquent et démasquent les interruptions (SVC + SSM). Pour le distributeur (DISPAT), il faut ajouter le temps de l'instruction LPSW.

Temps d'exécution des principaux services de SYNCOP (en µs)

DISPAT	210
WAIT	40
WAITM	140
CHECK	110
POST	80
GTMN	230
FRMN	140
STIMER	450
RTIMER	240
EXTINT	300
PUTLST	200
GETLST	170
EXCP	100
IOINT	230

Temps de simulation des instructions privilégiées (en µs)

SSM + SVC	130
LPSW	147

G L O S S A I R E

ABONNE : programme utilisateur du service de transport.

CALCULATEUR PARTICIPANT : calculateur connecté au réseau.

CLIENT : logiciel permettant à un utilisateur d'accéder aux serveurs par le réseau.

FLOT : voie logique entre deux portes, par laquelle des abonnés peuvent échanger des lettres et des télégrammes dans les deux sens. Synonymes : voie virtuelle, liaison.

LETTRE : unité d'information échangée entre abonnés. La taille des lettres est variable et limitée à quelques milliers d'octets.

PAQUET : unité d'information transmise en un bloc par le réseau de communication. La taille des paquets est limitée (255 octets pour Cyclades).

PAV : protocole d'appareil virtuel. Définit un appareil standard pour le réseau.

PORTE : point d'accès pour un abonné au service de transport.

PROTOCOLE : ensemble des règles définissant le mode d'échange des informations entre correspondants, ainsi que le format et la signification de ces informations.

RESEAU DE COMMUNICATION : ensemble de calculateurs (noeuds) interconnectés par des lignes de communications, assurant le transport des paquets. Synonymes : sous-réseau, machine de commutation de paquets.

SERVEUR : logiciel permettant l'accès à un service depuis le réseau.

SERVICE DE TRANSPORT : service permettant aux abonnés de communiquer entre eux par lettres et télégrammes. Le service de transport utilise le réseau de communication.

STATION DE TRANSPORT (ST) : représentant du service de transport pour les abonnés d'un calculateur participant. La ST réalise le protocole de transport pour le calculateur participant où elle est implémentée.

TELEGRAMME : information (2 octets) échangée entre abonnés sur un flot. Les télégrammes circulent sur le flot indépendamment des lettres.

- Commande : LOCAL
  Réponse : **OK**
  Permet de passer en mode local : tous les paquets à émettre et destinés à la ST CP/67 sont passés directement au processus de réception de la ST.

- Commande : NLOCAL
  Réponse : **OK**
  Permet de quitter le mode local : tous les paquets passent par la ligne.

SERVEUR CP/67

- Commande : QUERY C
  Réponse : pt-id/st-id type adr.
  Donne la liste de toutes les connexions en cours .
  Pour chaque connexion :
  pt-id/st-id : adresse de la porte distante et de sa ST.
  type : nom de l'entrée
  T = terminal
  D = doc
  I = init
  P = imprimante
  R = lecteur de cartes
  adr : adresse du 1052X associé à la connexion (seulement pour les types T, D et I.

- Commande : PAINIT { %FORMAT* / %COLD* / WARM }

DISPONIBILITE : probabilité qu'un élément d'un système, ou un système, soit opérationnel à un instant donné.

FIABILITE : la fiabilité d'un système à un instant t est la probabilité que le système soit opérationnel entre les instants 0 et t.

B I B L I O G R A P H I E

[1] Présentation du Réseau CYCLADES.
L.POUZIN. GAL506.

[2] The Cyclades Network. Present State and Development trends.
L.POUZIN. RES 505.1.

[3] CIGALE implementation, tools and techniques. J.L.GRANGE.
SEAS Dublin. Sept.1975.

[4] Transport Protocol. Standard end-to-end protocol for heterogeneous computer Networks. H.ZIMMERMANN M.ELIE. Mai 1973.
SCH 519.2.

[5] Spécifications fonctionnelles de la station de transport du réseau CYCLADES. Protocole ST-ST. Mai 1973. SCH 502.3.

[6] Proposal for a Virtual Terminal Protocol (VTP). H.ZIMMERMANN.
Janvier 1976.

[7] Virtual Terminal Protocol (VTP). Proposed specifications.
A.VIVIER. H.ZIMMERMANN. TER 503.1.

[8] Projet CRIC. CSCII Grenoble. Janvier 1973.

[9] TELCOM. Un support de terminaux lourds sous un système à temps partagé. Z.PAPACHRISTODOULOU. Thèse USMG. Octobre 1974.

[10] Système intéractif dans un environnement réseau. J.P.ANSART. Thèse USMG. Février 1976.

[11] Le service de traitement par lots dans un réseau hétérogène. R.FOURNIER. Thèse USMG à soutenir.

[12] SYNCOP. Implémentation sous CP/67. N.X.DANG, R.FOURNIER, V.QUINT. Septembre 1975.

[13] SYNCOP. Macro-instructions sous CP/67.

[14] SYNCOP. Spécifications de réalisation IRIS 80. J.SEGUIN, G.SERGEANT. Septembre 1975.

[15] Proposal for a Basic File Management Protocol. M.GIEN. Mai 1976. DAT 514.1.

[16] Insertion d'une station de transport dans un système d'exploitation. H.ZIMMERMANN. Janvier 1975. SCH 546.

[17] On the decentralisation of a transport station of the Cyclades network. R.SAETTONE. DEA USMG Juin 1976.

[18] Control Program 67. Program Logic Manual. IBM GY20 - 0590.

[19] Control Program 67/Cambridge Monitor System User's Guide IBM GH20-0859.

[20] Mécanismes de base et réalisation de fonctions pour l'utilisation intéractive d'un réseau d'ordinateurs. A.ZHIRI. Thèse USMG Décembre 1973.

[21] Espaces virtuels et gestion de fichiers. X.de LAMBERTERIE. Thèse USMG. Juin 1973.

[22] Implémentation de la station de transport ST2 sous le système CP/67. N.X.DANG. Janvier 1976.

[23] NESTOR, le chargeur de SYNCOP. J.du MASLE.

[24] A Virtual Terminal for CP/67. M.REY. SEAS 1974. Zurich.

[25] Multiprogramming sub-system. J.du MASLE. Flaine 1975.