UNIVERSITE DE NANCY I

thèse de spécialité informatique

U.E.R. DE MATHEMATIQUES

serveurs et consommateurs dans un protocole full-duplex

Soutenue:

Le 27 février 1978

Par M. H. SABOURIN

Jury:

Président:

M. C. PAIR

Examinateurs:

M. J. C. DERNIAME

M. G. MONSONEGO

UNIVERSITE DE NANCY I

thèse de spécialité informatique

U.E.R. DE MATHEMATIQUES

serveurs et consommateurs dans un protocole full-duplex

Soutenue:

Le 27 février 1978

Par M. H. SABOURIN

Jury:

Président:

M. C. PAIR

Examinateurs:

M. J. C. DERNIAME

M. G. MONSONEGO

Je tiens à remercier:

Monsieur le professeur PAIR, qui m'a fait l'honneur d'accepter de présider ce jury.

Monsieur le professeur DERNIAME, qui m'a proposé ce travail, et qui m'a aidé à le mener à bien, par ses conseils et ses encouragements.

Monsieur le professeur MONSONEGO, qui a bien voulu m'en faciliter les conditions de réalisation.

Messieurs LUTZ et MICHALON pour leur collaboration lors des réalisations matérielles.

Mademoiselle HIPPY et monsieur WEIL, qui m'ont aidé à la présentation de cet ouvrage.

SOMMAIRE

IV - PARTIE: ACCES DES ACTIVITES AU PROCESSEUR ARITHMETIQUE ET LOGIQUE.

A - COLLABORATION DES PROCESSEURS.

I - EVENEMENTS TECHNOLOGIQUES.

II - SYSTEME D'INTERRUPTION.

B - REGULATION DES EVENEMENTS LOGIQUES.

C - REGULATION DES ACTIVITES.

I - TACHES IMBRIQUEES A UN SEUL NIVEAU.

1)- DEROULEMENT DES ACTIVITES ELEMENTAIRES D'ENTREE/SORTIE.

2)- DEROULEMENT DES ACTIVITES ELEMENTAIRES DE LIGNE.

3)- DEROULEMENT DES ACTIVITES ELEMENTAIRES DE COMPRESSION/ DECOMPRESSION.

4)- CAS OU LES PROCESSEURS D'ENTREE/SORTIE SONT LOCAUX.

5)- CAS OU LES TAMPONS IMAGES SONT LIMITES.

II - TACHES IMBRIQUEES A PLUSIEURS NIVEAUX.

1)- DEROULEMENT DES ACTIVITES ELEMENTAIRES DE COMPRESSION/ DECOMPRESSION.

2)- CAS PARTICULIERS.

D - PRIORITE DES ACTIVITES.

I - TACHES IMBRIQUEES A UN SEUL NIVEAU.

1)- FILE MONITEUR.

2)- FILE DE LIGNE.

3)- FILE D'ENTREE/SORTIE.

II - TACHES IMBRIQUEES A PLUSIEURS NIVEAUX.

III - CRITERES D'IMPLANTATION DES METHODES DE REGULATION DES ACTIVITES.

CONCLUSION.

INTRODUCTION

- Pour l'entrée des travaux à distance, UNIVAC a défini un protocole de communication appelé FULL-DUPLEX NINE THOUSAND TERMINAL REMOTE (FD-NTR). Ce protocole permet l'entrée simultanée de plusieurs travaux. La configuration terminale peut se composer de plusieurs processeurs d'entrée/sortie de type lecteur de cartes, imprimante ou perforateur de cartes. Le nombre maximum de ces processeurs d'entrée/sortie est limité à 15.

- L'échange d'informations entre le site central et le terminal est simultané dans les deux sens. Par contre, entre le site central et un processeur d'entrée/sortie, il s'opère à l'alternat. Les procédures de communication, qui mettent en oeuvre ce protocole, sont écrites pour un terminal UNIVAC 9000 connecté par voie téléphonique à un UNIVAC 1100.

- Pour répondre aux besoins de traitements par lots du Groupe de Chambre à Bulles à Hydrogène (C.B.H.), le Centre de Calcul de Strasbourg a pris en charge l'implantation du protocole FD-NTR sur le calculateur DIGITAL PDP 11/45 de ce laboratoire. Dans le cahier des charges qui décrivait ce projet, il était convenu que la procédure permettrait la mise en ligne d'un appareil d'acquisition de données (HPD) ayant un débit d'information de 80 K-bits/seconde. Pour cela, la réalisation de la procédure était prévue en deux étapes.

- Pour la première étape, la procédure devait supporter uniquement un lecteur de cartes (1000 c/mn) et une imprimante (600 l/mn); et devait pouvoir se dérouler simultanément avec le programme d'acquisition de données (les deux programmes étant indépendants de tout système).

- Pour la seconde étape le programme d'acquisition de données devait être intégré à la procédure.

- Ainsi la réalisation de la procédure présentait les trois problèmes suivant:

1)- Son adaptation à un matériel de technologie différente.

2)- Son optimisation pour permettre un débit de transmission 100 K-bits/seconde.

3)- Sa compatibilité avec un autre programme écrit indépendamment.

- Les mêmes problèmes pouvant être posés au centre de calcul pour un autre modèle de calculateur, il était important pour éviter des écritures "artisanales" de la procédure, de trouver pour celle-ci un formalisme de description, qui en clarifie l'expression et permette d'en faire une implantation systématique.

- Le formalisme de description développé à priori dans ce travail repose sur l'idée essentielle que tout problème d'exclusion mutuelle de plusieurs activités à une ressource critique peut se ramener au problème du Serveur et du Consommateur (tout au moins dans une procédure). Or le modèle du Serveur et du consommateur peut être formalisé par un automate synchrone ou asynchrone.

- Ce formalisme simple et d'implantation performante des verrous et des sémaphores conduit à une description facilement lisible de la procédure, au niveau du site terminal et du site central, tout en permettant de redéfinir deux sortes de protocoles de communication.

- La procédure de communication étant ainsi repensée et décrite, son implantation au niveau du calculateur du site central ou du site terminal dépend de la façon dont est géré le système d'interruption de celui-ci. Suivant le moment où nous convenons de le revalider, lors du traitement d'une interruption, nous pouvons distinguer deux méthodes d'implantation pour la procédure.

I - PARTIE

MODELE

DU SERVEUR ET DU CONSOMMATEUR

AVEC CONTRAINTES

- Le transfert d'un fichier d'images de données (images de carte ou images de ligne) nécessite plusieurs activités simultanées qui collaborent, deux à deux, selon le modèle du Serveur et du Consommateur. Afin de formaliser, par la suite, les mécanismes de collaboration de ces activités et leur mise en oeuvre, nous allons redéfinir le modèle du Serveur et du Consommateur.

I - EVENEMENTS ET ETATS:

- Considérons deux activités qui partagent n tampons identiques (n = 2). L'une des activités, nommée serveur, remplit les tampons, alors que l'autre, nommée consommateur, les vide. Chacune des deux activités se décompose en une suite d'activités élémentaires identiques. Pour le serveur, une activité élémentaire consiste à remplir un tampon, alors que pour le consommateur, elle consiste à vider un tampon. Dans chaque cas, elles libérent le tampon aprés l'avoir rempli ou vidé.

- Nous disons qu'une activité se bloque chaque fois que l'une de ses activités élémentaires se termine; et qu'elle est demandée chaque fois qu'elle peut disposer d'un tampon aprés s'être bloquée. Lorsqu'une activité est demandée, si une de ses activités élémentaires est amorcée, nous disons qu'elle est réveillée. Cependant, nous admettons que chaque activité peut être alternativement validée et invalidée, pour une durée indéterminée, par une ou plusieurs activités appelées contrôleurs. Ainsi, le serveur et le consommateur peuvent demeurer bloqués lorsqu'ils sont demandés.

- Dans ces conditions, si nous appelons événement toute cause susceptible d'entrainer le blocage d'une activité, ou de contribuer à son réveil; et si nous appelons état toute valeur, enregistrée par une ou plusieurs variables, qui permet de décider du réveil d'une activité bloquée, nous devons considérer deux sortes d'événements et d'états: les événements et

l'état intrinsèque liés au déroulement du serveur et du consommateur; et les événements et les états extrinsèques attachés aux contrôleurs.

1)- EVENEMENTS ET ETAT INTRINSEQUES:

- Les événements intrinsèques correspondent aux libérations des tampons effectuées par le serveur et le consommateur, et aux demandes de ceux-ci. Suivant leur correspondance, nous les nommons "tampon plein" et "demande serveur" pour le serveur, et "tampon vide" et "demande consommateur" pour le consommateur.

- Lorsqu'une activité libère un tampon, en fait, elle le met à la disposition de l'activité concurrente. Par conséquent, pour procéder aux réveils du serveur et du consommateur, lorsqu'ils se bloquent, il faut tenir à jour, à l'issue de chaque événement "tampon plein" et "tampon vide", les nombres de tampons qui sont respectivement à leur disposition. Comme la somme de ces deux nombres est constante, la connaissance de l'un d'eux est suffisante. Par suite, nous appelons état intrinsèque du serveur et du consommateur, le nombre de tampons dont dispose le serveur à l'issue d'un événement "tampon plein" ou "tampon vide". L'état intrinsèque varie de 0 à n. Nous désignons par I la variable qui l'enregistre.

2)- EVENEMENTS ET ETATS EXTRINSEQUES:

- Les événements extrinsèques correspondent aux validations et invalidations des réveils du serveur et du consommateur par les contrôleurs. Il faut cependant préciser que chaque contrôleur valide et invalide soit le serveur, soit le consommateur. Mais ceux-ci peuvent être asservis par plusieurs contrôleurs indépendants ou non entre eux.

- Pour simplifier la suite de l'exposé, nous nous limitons à deux contrôleurs, l'un pour le serveur et l'autre pour le consommateur. Lorsque le serveur est validé, puis invalidé, nous disons qu'il est initialisé, puis terminé. Nous nommons les événements correspondants "initialiser serveur" et "terminer serveur". Par contre, dans le cas du consommateur nous disons qu'il est résumé et suspendu. Nous nommons les événements correspondants, "résumer consommateur" et "suspendre consommateur".

- Un événement extrinsèque n'agit pas directement sur une activité (serveur ou consommateur). Il valide ou invalide uniquement ses réveils jusqu'à la réalisation de l'événement extrinsèque opposé. Comme la réalisation d'un tel événement coïncide rarement avec une demande de l'activité, il faut le mémoriser. Pour cela, chaque activité (serveur et consommateur) dispose d'autant d'indicateurs qu'il y a de contrôleurs qui l'asservissent. Chacun de ces indicateurs, appelé indicateur de validation, est positionné puis annulé à chaque validation puis invalidation de l'activité qui sont effectuées par le contrôleur qui lui correspond.

- De plus, une activité (serveur ou consommateur) peut être demandée pendant qu'elle est invalidée. Pour rendre son réveil effectif dés qu'elle est validée, l'événement intrinsèque ("demande serveur" ou "demande consommateur") doit également être mémorisé. A cette fin, chaque activité (serveur ou consommateur) dispose d'un indicateur appelé indicateur de réveil. Celui-ci est positionné chaque fois que l'activité est demandée, et annulé dès qu'elle est réveillée.

- Dans ces conditions, nous appelons état extrinsèque d'une activité, toute configuration binaire prise par son indicateur de réveil et ses indicateurs de validation. Par la suite, nous désignerons plus généralement ces indicateurs sous le nom d'indicateurs d'états. Dans notre cas, le serveur et le consommateur disposent l'un et l'autre d'un indicateur de réveil et d'un indicateur de validation. Nous désignons respectivement ces indicateurs par sr et sv pour le serveur, et cr et cv pour le consommateur.

II - EMULATION DES EVENEMENTS:

- Nous appelons émulateur toute activité qui, à l'issue d'un ou plusieurs événements, modifie et consulte le ou les états concernés soit pour demander, soit pour réveiller le serveur ou le consommateur, si ce n'est les deux. Nous disons qu'un tel émulateur émule des événements.

- Si tout événement est mémorisé, par l'activité qui le réalise, sur l'indicateur d'état qui lui est réservé, nous pouvons admettre que la mise à jour et la consultation des états sont assurées, conformément aux valeurs prises par les indicateurs d'états, par un unique émulateur qui est réveillé, indépendamment de la réalisation des événements, à des intervalles de temps réguliers ou non. Toutefois, ceci suppose que le serveur et le consommateur disposent chacun d'un indicateur d'état supplémentaire pour mémoriser respectivement les événements "tampon plein" et "tampon vide". Ces deux indicateurs d'état sont encore appelés indicateurs de blocage. Nous les désignons respectivement par sb pour le serveur et cb pour le consommateur.

- Nous pouvons également admettre que toute activité provoque dès qu'elle réalise un événement, le réveil d'un émulateur qui modifie et consulte uniquement l'état concerné conformément à l'événement réalisé. Toutefois, ceci suppose qu'un émulateur soit associé à chaque type d'événement. Mais dans ce cas, deux émulateurs réveillés à l'issue de deux événements de types différents ne peuvent pas accéder s multanément à un même état, car le serveur et le consommateur peuvent alors rester définitivement bloqués. Par conséquent, pour résoudre les conflits d'accés de plusieurs émulateurs à un même état, ceux-ci doivent être réveillés un à un chaque fois que l'état est accessible. Pour cela, dès qu'une activité réalise un événement, elle range le nom de celui-ci dans une file d'attente d'événements. Nous disons alors que l'activité déclenche un événement. L'émulateur d'un événement ainsi retardé est réveillé, par une activité appelée régulateur d'événements, dès que l'état auquel il doit accéder est disponible. Nous disons alors que le régulateur d'événements libère un événement.

- Nous allons maintenant comparer respectivement les deux mécanismes d'émulation d'événements présentés ci-dessus à un circuit séquentiel synchrone et à un circuit séquentiel asynchrone.

1)- EMULATION SYNCHRONE DES EVENEMENTS:

- Le serveur et le consommateur devant être demandés avant d'être réveillés, l'émulateur consulte, à chacun de ses réveils, d'abord les indicateurs de blocage, puis chacun des indicateurs de réveil.

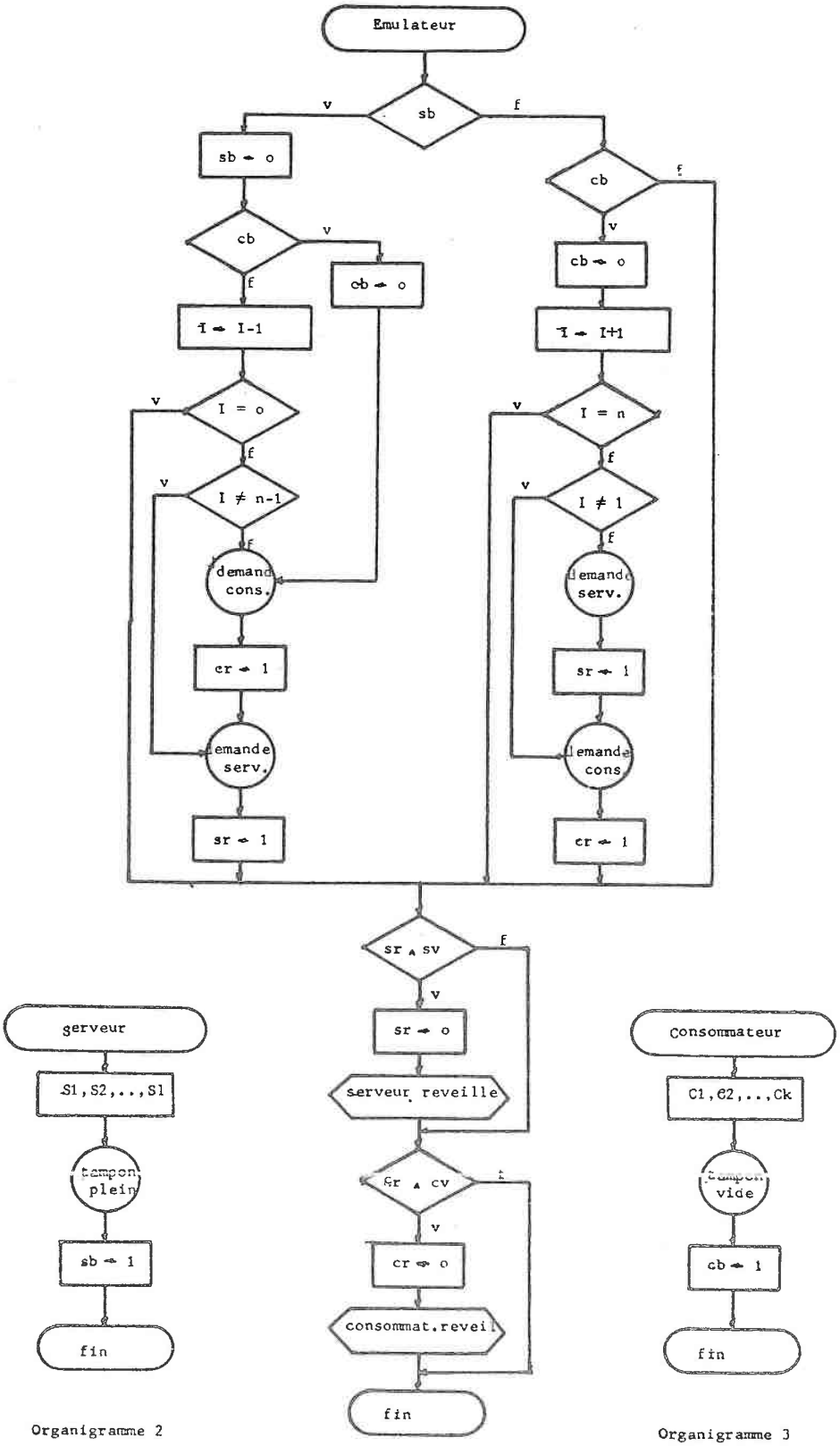
a)- Mise à jour et consultation des états:

- Lorsque le serveur a rempli un tampon sans que le consommateur en ait vidé un (sb positionné et cb annulé), l'émulateur annule l'indicateur sb, puis décrémente l'état intrinsèque. Si celui-ci est alors positif, ce qui signifie que le serveur a au moins un tampon à sa disposition, il mémorise l'événement "demande serveur" (sr positionné). De plus, si l'état intrinsèque obtenu égale n-1, ce qui équivaut à dire que le consommateur était en attente de tampon, il mémorise l'événement "demande consommateur" (cr positionné) - Voir organigramme 1 -.

- Au contraire, lorsque le consommateur a vidé un tampon sans que le serveur en ait rempli un (sb annulé, cb positionné), l'émulateur annule l'indicateur cb, puis incrémente l'état intrinsèque. Si celui-ci est alors inférieur à n, il mémorise l'événement "demande consommateur" (cr positionné). De plus, si l'état intrinsèque obtenu égale 1, il mémorise l'événement "demande serveur" (sr positionné) - Voir organigramme 1 -.

- Par contre, si le serveur et le consommateur ont traité l'un et l'autre un tampon (sb positionné et cb positionné), l'émulateur annule les indicateurs sb et cb, et mémorise les événements "demande serveur"

EMULATION SYNCHRONE DES EVENEMENTS



Organigramme 2

Organigramme 1

Organigramme 3

(sr positionné) et "demande consommateur" (cr positionné), sans modifier l'état intrinsèque - Voir organigramme 1 -.

- Au cas où le serveur et le consommateur n'ont pas traité de tampon (sb annulé, cb annulé), il scrute immédiatement les indicateurs de réveil sr et cr.

- Aprés avoir traité les indicateurs de blocage sb et cb, l'émulateur scrute successivement les indicateurs de réveil sr et cr. Lorsque l'un d'eux est positionné, avant de réveiller l'activité correspondante (serveur ou consommateur), il s'assure que son indicateur de validation (sv ou cv)est positionné. Si c'est le cas, il annule l'indicateur de réveil sr ou cr, et réveille l'activité; sinon il ne modifie pas l'état extrinsèque.

b)- Assimilation à un circuit séquentiel synchrone:

- Ce mécanisme d'émulation des événements est comparable d'une part, pour les demandes du serveur et du consommateur à un circuit séquentiel synchrone, à deux sorties, qui serait susceptible de prendre n+1 états sous l'effet de deux entrées; et d'autre part, pour leurs réveils, à deux circuits combinatoires qui auraient chacun deux entrées et une sortie.

- L'horloge du circuit séquentiel synchrone sert d'émulateur, ses entrées concrétisent les indicateurs de blocage sb et cb,et ses sorties correspondent aux indicateurs de réveil sr et cr.

- Les entrées d'un circuit combinatoire concrétisent l'indicateur de réveil et l'indicateur de validation d'une activité (serveur ou consommateur), et sa sortie correspond au réveil de cette activité.

- Par conséquent, nous allons représenter les demandes du serveur et du consommateur par une matrice des transitions d'états et leurs réveils par deux tables de vérité.

sv: indicateur de validation sr: indicateur de reveil	Activité reveillée
0 0	
0 1	
1 0	
1 1	Serveur

Tableau 2: VALIDATIONS ET INVALIDATIONS DU SERVEUR

I	sb=0 , cb=0	sb=1 , cb=0	sb=0 , cb=1	sb=1 , cb=1
0	0		1 sr ← 1 cr ← 1	
1	1	0	2 cr ← 1	1 sr ← 1 cr ← 1
2	2	1 sr ← 1	3	2 sr ← 1 cr ← 1
3	3	2 sr ← 1 cr ← 1		

Tableau 1: BLOCAGES ET DEMANDES DU SERVEUR ET DU CONSOMMATEUR (n=3)

cv: indicateur de validation cr: indicateur de reveil	Activite reveillée
0 0	
0 1	
1 0	
1 1	Consommateur

Tableau 3: VALIDATIONS ET INVALIDATIONS DU CONSOMMATEUR

- La matrice des transitions d'états est un tableau à double entrée, de n+1 lignes et 4 colonnes. Chaque ligne correspond à un état intrinsèque, et chaque colonne à une configuration binaire des indicateurs de blocage. L'intersection d'une ligne et d'une colonne de ce tableau traduit, pour un état intrinsèque donné et une configuration binaire des indicateurs de blocage, l'état intrinsèque successeur et les valeurs prises par les indicateurs de réveil - Voir tableau 1 -. Nous disons que cette matrice est un automate synchrone.

- Chaque table de vérité est un vecteur à 4 composantes. Le rang d'une composante correspond à un état intrinsèque; et la valeur d'une composante traduit, pour un état extrinsèque donné, le réveil ou non du serveur ou du consommateur - Voir tableaux 2 et 3 -.

2)- EMULATION ASYNCHRONE DES EVENEMENTS:

- Le retardement des événements extrinsèques crée des difficultés du fait qu'ils peuvent être réalisés à tout moment. Il faut, d'une part éviter qu'ils saturent la file d'attente d'événements, et d'autre part veiller à ce qu'ils soient libérés dans l'ordre de leur réalisation. Pour résoudre ces problèmes, nous convenons, à la réalisation d'un événement extrinsèque, d'invalider le contrôleur qui l'a réalisé jusqu'à la mise à jour de l'état extrinsèque concerné. Ceci revient à dire qu'un contrôleur ne peut pas réaliser un événement extrinsèque tant que le précédent n'a pas été acquitté. De cette façon, en limitant la mise en attente des événements extrinsèques à un événement par contrôleur, nous allégeons la tâche du régulateur d'événement. Les événements retardés sont alors indépendants, et leur ordre de libération est lié uniquement à des critères de priorité.

- Pour simplifier la description du mécanisme de retardement et de

libération des événements, nous convenons que ceux-ci sont enregistrés dès leur réalisation, sur des indicateurs que nous appelons indicateurs d'attente. Pour cela, nous assignons un indicateur d'attente à chaque type d'événement intrinsèque et à chaque type de couple d'événements extrinsèques. Ceci revient à concevoir une file d'attente d'événements qui aurait une entrée fixe par type d'événement ou de couple d'événements. Nous pouvons évidemment envisager d'autres sortes de files d'attente (FIFO, LIFO, etc...) et d'autres façons d'enregistrer les événements.

- Le régulateur d'événements scrute alors constamment les indicateurs d'attente dans un ordre précis pour détecter ceux qui sont positionnés. Lorsque l'un d'eux est positionné, le régulateur l'annule et libère l'événement correspondant aprés avoir positionné un verrou, il suspend alors la scrutation des autres indicateurs tant que le verrou est positionné. Celui-ci est annulé aprés chaque mise à jour et consultation de l'état intrinsèque ou des états extrinsèques.

a)- Mise à jour et consultation de l'état intrinsèque:

- A la libération d'un événement "tampon plein", l'émulateur qui est réveillé, décrémente l'état intrinsèque. Si celui-ci est alors positif, il déclenche l'événement "demande serveur", car le serveur dispose au moins d'un tampon. De plus, si l'état intrinsèque obtenu est égal à n-1, il déclenche également l'événement "demande consommateur", car le consommateur était en attente d'un tampon - Voir organigramme 4 -.

- Au contraire, à la libération d'un événement "tampon vide", l'émulateur concerné incrémente l'état intrinsèque. Si celui-ci est inférieur à n, il déclenche l'événement "demande consommateur". De plus, si l'état intrinsèque obtenu est égal à 1, il déclenche aussi l'événement "demande serveur" - Voir organigramme 5 -.

b)- Mise à jour et consultation des états extrinsèques:

- A la libération d'un événement "initialiser serveur", l'émulateur qui est réveillé, positionne l'indicateur de validation sv du serveur et vérifie que l'indicateur de réveil sr, de celui-ci, est positionné. Si c'est le cas, il l'annule et réveille le serveur. Par contre, à la libération d'un événement "terminer serveur", l'indicateur de validation sv est uniquement annulé - Voir organigrammes 6 et 8 -.

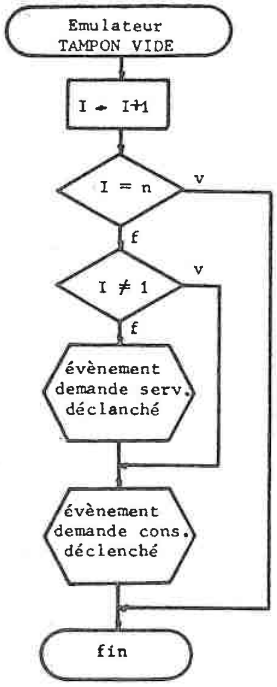
- A la libération d'un événement "demande serveur", l'émulateur concerné réveille le serveur si celui-ci est initialisé (sv positionné); sinon il positionne uniquement l'indicateur de réveil sr du serveur pour signaler qu'il est demandé - Voir organigramme 10 -.

- A la libération d'un événement "résumer consommateur", l'émulateur qui est réveillé, positionne l'indicateur de validation cv du consommateur. Puis il vérifie que l'indicateur de réveil cr, de celui-ci, est positionné. Si c'est le cas, il l'annule et réveille le consommateur. Par contre, à la libération de l'événement "suspendre consommateur", l'indicateur de validation cv est uniquement annulé - Voir organigrammes 7 et 9 -.

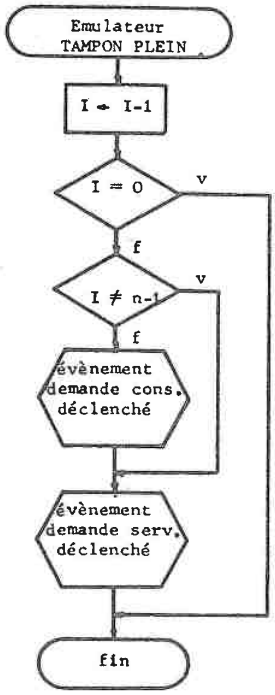
- A la libération d'un événement "demande consommateur", l'émulateur concerné réveille le consommateur s'il est résumé (cv positionné); sinon il positionne uniquement l'indicateur de réveil de celui-ci pour signaler qu'il est demandé - Voir organigramme 11 -.

c)- Assimilation à des circuits séquentiels asynchrones:

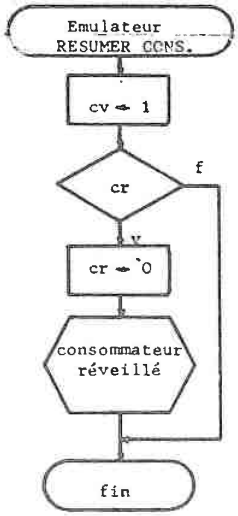
- Ce mécanisme d'émulation des événements est comparable pour les demandes du serveur et du consommateur à un circuit séquentiel asynchrone, à deux sorties, qui serait susceptible de prendre n+1 états sous l'effet d'impulsions soumises non simultanément à deux entrées. Les impulsions de l'une des entrées correspondent à la libération des événements "tampon plein", et celles de l'autre entrée correspondent à la libération des



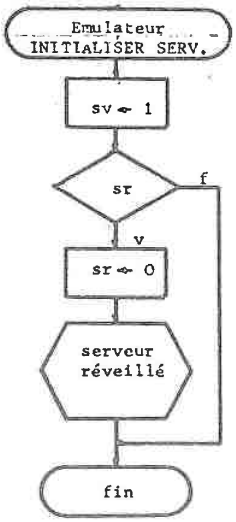
Organigramme 5



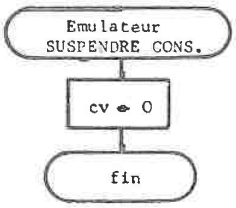
Organigramme 4



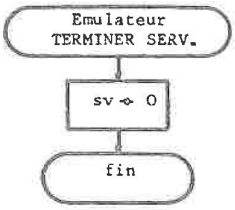
Organigramme 7



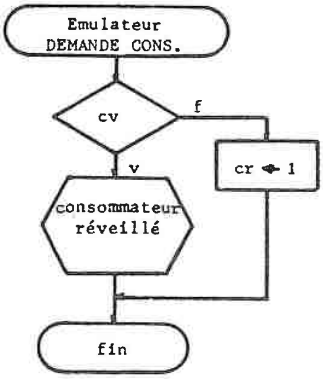
Organigramme 6



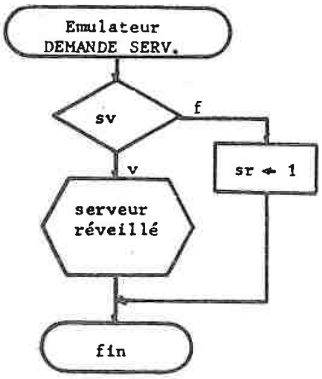
Organigramme 9



Organigramme 8



Organigramme 11



Organigramme 10

événements "tampon vide". Par contre les impulsions de sortie concrétisent la libération des événements "demande serveur" et "demande consommateur".

- De même, nous pouvons assimiler chacun des mécanismes de réveil du serveur et du consommateur à un circuit séquentiel asynchrone à trois états, une sortie et deux entrées. Par conséquent, nous allons représenter chacun de ces trois mécanismes par une matrice des transitions d'états.

- La matrice des transitions d'états, qui décrit les demandes du serveur et du consommateur, est un tableau à double entrée de n+1 lignes et deux colonnes. Chaque ligne correspond à un état intrinsèque, tandis que que l'une des colonnes correspond aux libérations des événements "tampon plein", et l'autre aux libérations des événements "tampon vide". L'intersection d'une ligne et d'une colonne de ce tableau traduit, pour un état intrinsèque donné et la libération d'un événement, l'état intrinsèque successeur et les événements qui sont éventuellement déclenchés - Voir tableau 2 -.

- La matrice des transitions d'états, qui décrit les réveils du serveur, est un tableau à double entrées. Chaque ligne de ce tableau correspond à un état extrinsèque du serveur, c'est a dire, à une configuration binaire de ses indicateurs de réveil et de validation (sr et sv). Par contre, l'une de ses colonnes correspond aux libérations des événements "demande serveur" et l'autre aux libérations alternées des événements "initialiser serveur" et "terminer serveur" - Voir tableau 3 -.

- De même, la matrice des transitions d'états, qui décrit les réveils du consommateur est un tableau à double entrée dont chaque ligne correspond à un état extrinsèque, c'est à dire, à une configuration binaire de ses indicateurs de réveil et de validation (cr et cv). Par contre, l'une de ses colonnes correspond aux libérations des événements "demande consommateur" et l'autre aux libérations alternées des événements "suspendre consommateur" et "résumer consommateur" - Voir tableau 4 -.

- L'intersection d'une ligne et d'une colonne de l'une ou l'autre des

sv: indicateur de validation sr: indicateur de reveil	INITIALISER/TERMINER SERVEUR	DEMANDE SERVEUR
A 0 0	C	B
B 0 1	C reveil serveur	
C 1 0	A	C reveil serveur

Tableau 3: VALIDATIONS ET INVALIDATIONS DU SERVEUR

I	TAMPON VIDE	TAMPON PLEIN
0	1 demande serveur demande consommateur	
1	2 demande consommateur	0
2	3	1 demande serveur
3		2 demande serveur demande consommateur

Tableau 2: BLOCAGES ET DEMANDES DU SERVEUR ET DU CONSOMMATEUR (n=3)

cv: indicateur de validation cr: indicateur de reveil	SUSPENDRE/RESUMER CONSOMMATEUR	DEMANDE CONSOMMATEUR
A 0 0	C	B
B 0 1	C reveil consommateur	
C 1 0	A	C reveil consommateur

Tableau 4: VALIDATIONS ET INVALIDATIONS DU CONSOMMATEUR

matrices de transitions d'états, qui décrivent les réveils du serveur et du consommateur, traduit, pour un état extrinsèque donné et la libération d'un événement, l'état extrinsèque successeur et le réveil éventuel de l'activité, serveur ou consommateur.

- Nous disons que les trois matrices de transitions d'états sont des automates asynchrones.

II - PARTIE

PROTOCOLE DE COMMUNICATION
AVEC
SOLLICITATIONS INDIVIDUELLES
DES
MESSAGES DE DONNEES

A - MECANISMES DE SYNCHRONISATION POUR LE TRANSFERT D'UN FICHIER D'IMAGES DE DONNEES

- Le transfert d'un fichier d'images de données, d'un site à l'autre, se fait par l'intermédiaire des messages de données.

- Un message de données est une chaîne de caractères de longueur fixe qui comprend trois parties: un en-tête, un texte et une fin de texte. Le texte d'un message de données est une suite d'images de données compressées ou non; c'est à dire, une suite d'images de données dont les espaces ont été en partie condensés ou ignorés. De plus, la dernière image de données compressée d'un texte et, de ce fait, la première peuvent être éventuellement incomplètes.

- Par conséquent, quelle que soit la structure de compression utilisée, nous devons envisager, pour chaque fichier d'images de données à transférer d'un site à l'autre, 5 activités simultanées réparties sur les deux sites:

  * Une activité d'entrée et une activité de compression sur le site émetteur.
  * Une activité de décompression et une activité de sortie sur le site récepteur.
  * Une activité ligne qui dépend des deux sites.

- L'activité d'entrée sert des images de données à l'activité de compression qui les consomme pour former des messages de données. L'activité de décompression sert des images de données décompressées à l'activité de sortie qui les consomme pour les éditer. Et l'activité ligne consomme les messages de données, que lui sert l'activité de compression, pour les servir à l'activité de décompression.

- Ainsi les couples d'activités d'entrée et de compression, de compression et de ligne, de ligne et de décompression, et de décompression et de sortie

sont chacun assimilables à un serveur et un consommateur. Par conséquent, nous allons décrire les mécanismes de collaboration de ces activités par des automates asynchrones. Par la suite, nous sous-entendons par couple d'activités l'un ou l'autre des couples cités ci-dessus.

I - TAMPONS IMAGES ET TAMPONS MESSAGES:

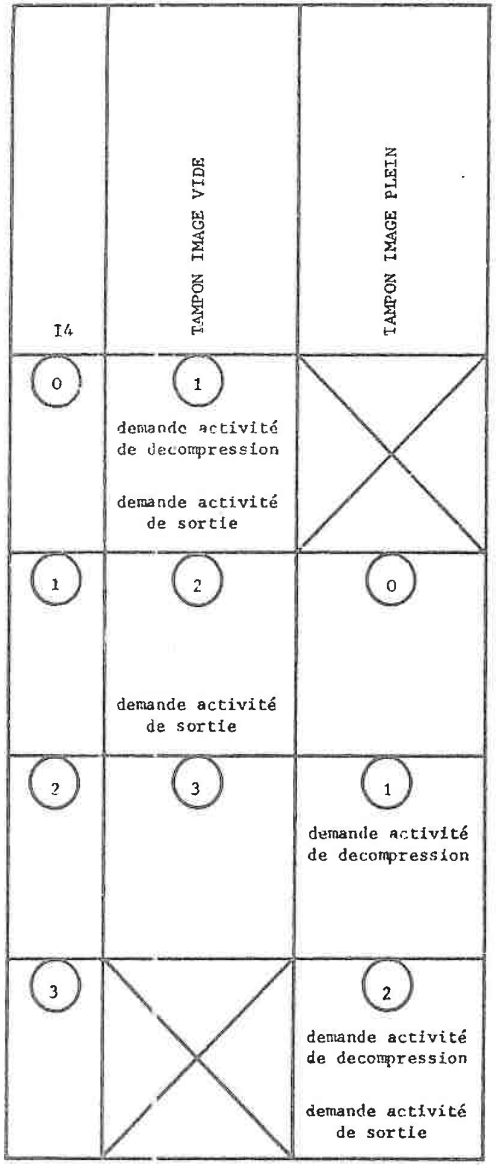
- Nous admettons que chaque couple d'activités C_i, i variant de 1 à 4, dispose de T_i tampons, avec T_i supérieur ou égal à 2.

- Pour les activités d'entrée et de compression, comme pour les activités de décompression et de sortie, chacun des T_i tampons, qu'elles ont en commun, est conçu pour recevoir une image de données. Nous désignons ces tampons sous le nom de tampons images.

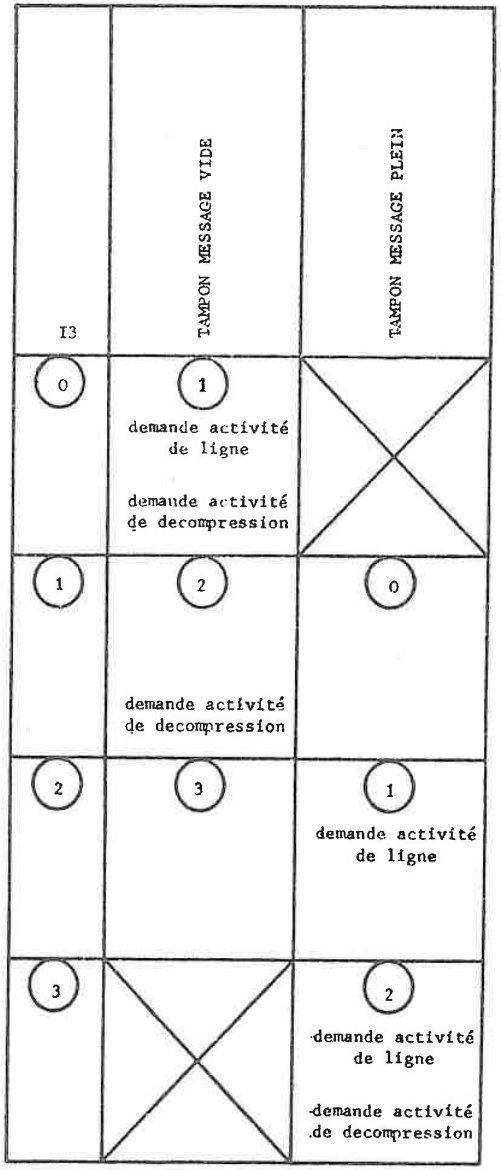
- Par contre, pour les activités de compression et de ligne, comme pour les activités de ligne et de décompression, chacun des T_i tampons, qu'elles ont en commun, est conçu pour recevoir un message de données. Nous désignons ces derniers sous le nom de tampons messages.

II - BLOCAGES ET DEMANDES DES ACTIVITES:

- Chaque couple d'activités se déroulant suivant le modèle du serveur et du consommateur, nous associons à chacun d'eux un automate asynchrone qui traduit les demandes éventuelles de ses deux activités lorsque l'une



BLOCAGES ET DEMANDES
DES ACTIVITES DE DECOMPRESSION ET DE SORTIE (Ti=3)



BLOCAGES ET DEMANDES
DES ACTIVITES DE LIGNE ET DE DECOMPRESSION (Ti=3)

I2	TAMPON MESSAGE VIDE	TAMPON MESSAGE PLEIN
0	1 demande activité de compression demande activité de ligne	
1	2 demande activité de ligne	0
2	3	1 demande activité de compression
3		2 demande activité de compression demande activité de ligne

BLOCAGES ET DEMANDES
DES ACTIVITES DE COMPRESSION ET DE LIGNE (Ti=3)

I1	TAMPON IMAGE VIDE	TAMPON IMAGE PLEIN
0	1 demande activité d'entrée demande activité de compression	
1	2 demande activité de compression	0
2	3	1 demande activité d'entrée
3		2 demande activité d'entrée demande activité de compression

BLOCAGES ET DEMANDES
DES ACTIVITES D'ENTREE ET DE COMPRESSION (Ti=3)

ou l'autre libère un tampon avant de se bloquer. Pour un couple d'activités C_i qui dispose de T_i tampons, i variant de 1 à 4, un tel automate a T_i états, deux entrées et deux sorties. Ses entrées sont assimilées à des événements du type "tampon plein" et "tampon vide", et ses sorties correspondent à des événements du type "demande serveur" et "demande consommateur". Enfin, chacun de ses états représente le nombre de tampons qui sont éventuellement à la disposition de l'activité considérée comme serveur, à l'issue d'un événement de type "tampon plein" ou tampon vide". L'automate asynchrone de chaque couple d'activités est donné ci-aprés avec les événements qui lui sont propres.

III - DESCRIPTION DES ACTIVITES ELEMENTAIRES:

1)- ACTIVITE D'ENTREE:

- Une activité élémentaire d'entrée introduit une image de données dans un tampon image, puis libère celui-ci lorsqu'il est plein (événement "tampon image plein").

2)- ACTIVITE DE COMPRESSION:

- Pour compresser une image de données, l'activité de compression accède simultanément à un tampon image et à un tampon message. Dès que l'un est vide ou l'autre plein, elle doit le libérer (événement "tampon image vide" ou "tampon message plein"). Mais il peut arriver que le tampon image soit vide lorsque le tampon message est plein. Dans ce cas, l'activité de compression peut déclencher l'un et l'autre événements avant de se bloquer.

Mais, ceux-ci ne dépendent pas du même automate asynchrone. Ce qui veut dire que l'activité de compression est demandée deux fois: dès qu'elle peut disposer d'un autre tampon image, et dès qu'elle peut disposer d'un autre tampon message - Voir les automates asynchrones des activités d'entrée et de compression, et des activités de compression et de ligne -. Etant donné que l'activité de compression doit disposer d'un tampon de chaque sorte pour compresser une image de données, les deux automates asynchrones doivent alors être dépendants.

- Cependant, pour éviter d'alourdir l'émulation des événements, il est préférable de maintenir l'indépendance des deux automates asynchrones. Celle-ci peut être obtenue à condition que l'activité de compression conserve un tampon à chacun de ses blocages. Dès lors, si à l'issue de la compression d'une image de données, le tampon image est vide et le tampon message plein, nous convenons qu'elle libère d'abord le tampon message, puis le tampon image dès qu'elle dispose d'un autre tampon message.

- Ainsi, nous avons deux sortes d'activités élémentaires de compression. D'une part, celles qui compressent complètement ou incomplètement une image de données depuis un tampon image vers un tampon message, et qui libèrent le tampon message dès qu'il est plein, sinon le tampon image dès qu'il est vide. Et d'autre part, celles qui libèrent uniquement un tampon image.

3)- ACTIVITE DE LIGNE:

- Pour transférer un message de données d'un site à l'autre, l'activité ligne accède simultanément à un tampon message du site émetteur et à un tampon message du site récepteur. A l'issue du transfert, ceux-ci sont respectivement vide et plein (événements "tampon message vide" et "tampon message plein"). Comme pour l'activité de compression, pour préserver l'indépendance de l'automate asynchrone des activités de compression et de ligne avec celui des activités de ligne et de décompression, l'activité

doit conserver un tampon message à chacun de ses blocages. Dans ces conditions, à l'issue du transfert d'un message de données, nous convenons qu'elle libère d'abord le tampon message du site récepteur, puis celui du site émetteur dès qu'elle dispose d'un autre tampon message sur le site récepteur. De cette façon, l'automate asynchrone du site émetteur suscite essentiellement le transfert des messages de données, tandis que celui du site récepteur entraîne uniquement leur sollicitation.

- Par suite, nous désignons par activité élémentaire d'émission de message, l'activité ligne élémentaire qui transfert un message de données et libère le tampon message du site récepteur. Et nous désignons par activité élémentaire de sollicitation de message, l'activité ligne élémentaire qui libère uniquement le tampon message sur le site émetteur.

- Lorsque l'activité élémentaire de sollicitation de message déclenche l'événement "tampon message vide" pour libérer le tampon message du site émetteur, elle transfere cet événement du site récepteur vers le site émetteur pour le déposer dans une file d'attente d'événement de ce dernier. Un tel événement qui est transféré d'un site à l'autre est communément appelé message de contrôle. Dans le cas de l'activité élémentaire de sollicitation de message, nous le désignons par message de sollicitation.

4)- ACTIVITE DE DECOMPRESSION:

- Pour décompresser une image de données, l'activité de décompression accède simultanément à un tampon message et à un tampon image. Lorsque le tampon message est vide (événement "tampon message vide"), le tampon image peut être plein (événement "tampon image plein"). Par conséquent, pour que l'automate asynchrone des activités de ligne et de décompression et celui des activités de décompression et de sortie soient indépendants, l'activité de décompression doit conserver un tampon à chacun de ses blocages. Dès lors, lorsque le tampon image et le tampon message deviennent simultanément plein

et vide, nous convenons qu'elle libère d'abord le tampon image, puis le tampon message dès qu'elle peut disposer d'un autre tampon image.

- Ainsi, il y a deux sortes d'activités élémentaires de décompression. D'une part, celles qui décompressent une image de données complète ou incomplète depuis un tampon message vers un tampon image, et qui libèrent le tampon image dès qu'il est plein, sinon le tampon message dès qu'il est vide. Et d'autre part, celles qui libèrent uniquement un tampon message.

5)- ACTIVITE DE SORTIE:

- Une activité élémentaire de sortie édite une image de données retenue dans un tampon image, puis libère celui-ci lorsqu'il est vide (événement "tampon image vide").

IV - INITIALISATION ET TERMINAISON DES ACTIVITES:

1)- INITIALISATION DES ACTIVITES D'ENTREE ET DE LIGNE:

- Le transfert d'un fichier d'images de données d'un site à l'autre, est amorcé, sur le site émetteur, par deux contrôleurs qui valident respectivement l'activité d'entrée (événement "initialiser entrée"), et l'activité ligne (événement "initialiser ligne"). Le second dépend du premier. Il est réveillé lors de l'émulation de l'événement "initialiser entrée".

- Chaque fois que l'activité ligne est initialisée, nous convenons qu'elle retient un tampon message dont le contenu a déjà été transféré. De cette façon, dès qu'elle dispose d'un tampon message vide sur le site récepteur, elle peut amorcer le transfert du premier message de données, préparé par l'activité de compression, en libérant le tampon message vide du site émetteur. De plus, étant donné que ce tampon message vide inhibe les réveils de l'activité ligne sur le site émetteur tant qu'il n'est pas libéré, l'activité de compression peut construire les premiers messages de données, indépendamment de la disponibilité d'un tampon message sur le site récepteur - Voir automate asynchrone des activités de compression et de ligne -.

- Pour que l'activité ligne soit réveillée sur le site récepteur, l'événement "initialiser ligne" doit parvenir à celui-ci. Nous désignons encore cet événement sous le nom de message d'initialisation.

2)- TERMINAISON DES ACTIVITES D'ENTREE ET DE LIGNE:

- Un fichier d'images de données se termine par une image fin de fichier.

- Lorsque l'activité d'entrée reconnaît une image fin de fichier, elle s'invalide (événement "terminer entrée") avant de libérer le tampon image. De cette façon, dès qu'elle dispose d'un autre tampon, elle ne peut plus être réveillée.

- Lorsque l'activité de compression rencontre à son tour l'image fin de fichier, elle l'insère dans le message de données en cours de formation et libère le tampon message. Puis, dés qu'elle dispose d'un autre tampon message, elle libère le tampon image. Ainsi, elle ne peut plus être réveillée, puisque d'une part elle a libéré le dernier tampon image rempli par l'activité d'entrée, et que d'autre part elle immobilise un tampon message.

- Lorsque l'activité ligne a transféré le message de données avec l'image fin de fichier, elle s'invalide (événement "terminer ligne") avant de libérer le tampon message du site récepteur. De cette façon, lorsqu'elle dispose d'un autre tampon message sur le site récepteur, elle ne libère pas son dernier tampon message sur le site émetteur.

- Lorsque l'activité de décompression prend en compte l'image fin de fichier, elle libère le tampon image et attend de disposer d'un autre tampon image pour libérer le tampon message. Ainsi, elle ne peut plus être réveillée.

- L'activité de sortie se bloque alors faute de tampon.

- L'activité ligne ne peut pas être initialisée tant qu'elle n'est pas terminée. Cependant, le transfert d'un fichier d'images de données peut être amorcé, alors que le précédent est toujours en cours de décompression.

- Sont donnés ci-aprés les automates asynchrones qui décrivent les mécanismes de validation et d'invalidation des activités d'entrée et de ligne.

lv: indicateur de validation lr: indicateur de reveil	INITIALISER/TERMINER LIGNE	DEMANDE ACTIVITE DE LIGNE
A 0 0	C	B
B 0 1	C reveil activité de ligne	
C 1 0	A	C reveil activité de ligne

INITIALISATIONS ET TERMINAISONS
DE L'ACTIVITE DE LIGNE

ev: indicateur de validation er: indicateur de reveil	INITIALISER/TERMINER ENTREE	DEMANDE ACTIVITE D'ENTREE
A 0.0	C	B
B 0.1	C reveil activité d'entrée	
C 1.0	A	C reveil activité d'entrée

INITIALISATIONS ET TERMINAISONS
DE L'ACTIVITE D'ENTREE

B - ACCES AUX PROCESSEURS D'ENTREE/SORTIE ET DE LIGNE

- Le déroulement d'une activité élémentaire dépend d'un ou plusieurs processeurs. Ainsi,une activité élémentaire de compression/décompression dépend uniquement d'un processeur arithmétique et logique; tandis qu'une activité élémentaire d'entrée/sortie dépend en plus d'un processeur d'entrée sortie. Par contre, une activité ligne élémentaire dépend d'un processeur ligne et de deux processeurs arithmétiques et logiques.

- Nous distinguons deux sortes de processeurs: ceux, tels que les processeurs d'entrée/sortie et de ligne, qui peuvent dérouler une seule activité élémentaire à la fois; et ceux, tels que les processeurs arithmétiques et logiques, qui peuvent imbriquer le déroulement de plusieurs activités élémentaires. Nous disons que les premiers sont à accés séquentiel; tandis que les seconds sont à accés parallèle.

- Si les activités élémentaires d'une même activité sont les seules à accéder à un processeur, nous disons que celui-ci est local à l'activité. Par contre, si les activités élémentaires de plusieurs activités accèdent à un même processeur, nous disons que celui-ci est global à ces activités.

- Si un processeur à accés séquentiel est global à plusieurs activités, nous disons que celles-ci sont en exclusion mutuelle pour accéder à ce processeur.

I - EXCLUSION MUTUELLE DES ACTIVITES D'ENTREE/SORTIE:

- Un processeur d'entrée/sortie peut être local à une activité d'entrée/sortie (lecteur de cartes, imprimante), comme il peut être global à plusieurs activités d'entrée/sortie (unité de disques).

- Pour résoudre les conflits d'accés de m activités d'entrée/sortie ($m \geq 2$) à un processeur d'entrée/sortie, leurs réveils doivent être retardés chaque fois que celui-ci n'est pas disponible. Pour cela, dès qu'une activité validée est demandée, le nom de l'activité élémentaire qui doit être amorcée est rangé dans une file d'attente d'activités à m entrées, appelée file d'entrée/sortie. Nous disons alors que l'activité est déclenchée. Les activités d'entrée/sortie, ainsi retardées, sont réveillées une à une par un régulateur d'activités appelé régulateur d'entrée/sortie. Chaque fois qu'une activité élémentaire se termine, celui-ci consulte la file d'entrée/sortie pour en amorcer éventuellement une autre.

- En donnant ainsi l'accés au processeur d'entrée/sortie à tour de rôle aux activités d'entrée/sortie pour entrer ou sortir une image de données, le régulateur imbrique l'entrée ou la sortie des m fichiers d'images de données. Par suite, nous disons que ceux-ci sont entrés ou sortis en parallèle.

II - EXCLUSION MUTUELLE DES ACTIVITES DE LIGNE:

- Pour le transfert des fichiers d'images de données d'un site à un autre, nous admettons que nous disposons d'un seul processeur ligne. Celui-ci est succeptible d'être accédé simultanément dans les deux sens. Mais pour un sens donné, il est à accés séquentiel.

- Pour transférer un message de données, une activité de ligne accède deux fois séquentiellement au processeur ligne. Une première fois dans le sens émetteur-récepteur (activité élémentaire d'émission de message), et une seconde fois dans le sens récepteur-émetteur (activité élémentaire de sollicitation de message). Le processeur ligne est alors dit être accédé à l'alternat par l'activité ligne.

- Par conséquent, si nous voulons acheminer en même temps d'un site à l'autre, p fichiers d'images de données ($p \geq 1$) dans un sens, et q fichiers d'images de données ($q \geq 1$) dans l'autre sens, le processeur ligne sur chaque site est global à $p+q$ activités de ligne

- Pour éviter les conflits d'accés des $p+q$ activités de ligne au processeur ligne, une file d'attente d'activités à $p+q$ entrées, appelée file de ligne, leur est assignée sur chaque site pour retarder leurs réveils. Chacune des deux files de ligne est gérée par un régulateur d'activités appelé régulateur de ligne.

- Le régulateur de ligne d'un site donné imbrique les transferts des fichiers d'images de données destinés à l'autre site, en donnant à tour de rôle aux $p+q$ activités de ligne l'accés au processeur ligne pour qu'elles transfèrent un message de données ou de contrôle. Nous disons que ces fichiers d'images de données sont transmis en parallèle.

III - REGULATEURS D'ACTIVITES:

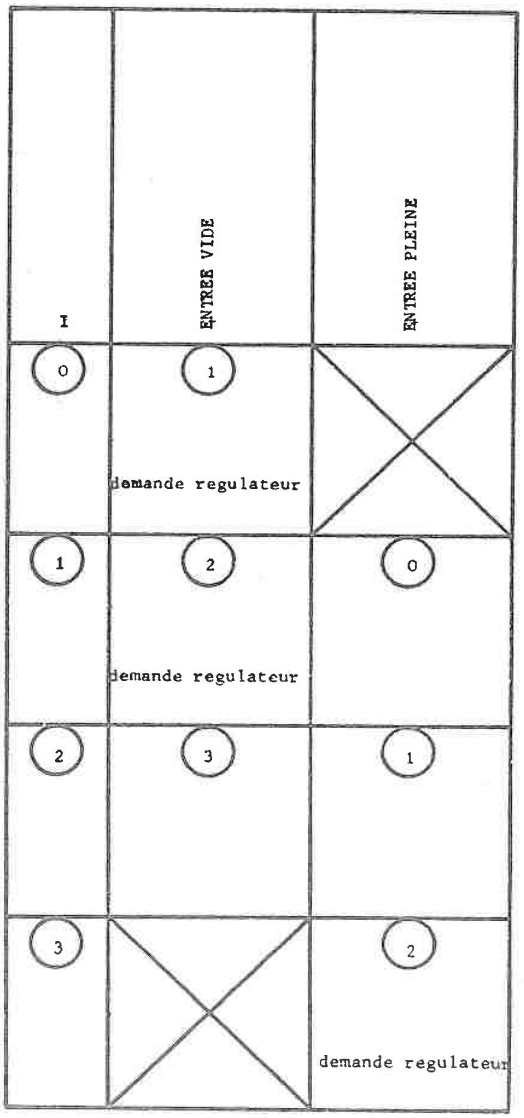
- Considérons une file d'attente d'activités à n entrées ($n \geq 2$), telle que la file de ligne et les files d'entrée/sortie. Ses entrées sont d'une part remplies par les émulateurs qui déclenchent les activités, et d'autre part vidées par le régulateur qui réveille les activités. Si nous admettons

que les événements sont libérés par un unique régulateur d'événements, les émulateurs accèdent alors un à un aux entrées de la file d'attente d'activités.

- Dans ces conditions, les activités élémentaires d'émulation et les activités élémentaires de régulation peuvent être considérées comme étant respectivement les activités élémentaires d'un serveur et d'un consommateur qui auraient en commun les n entrées de la file d'attente d'activités. Toutefois, dans ce modèle les libérations des entrées (événements "entrée pleine" et "entrée vide") entraînent uniquement les demandes du consommateur; c'est à dire, du régulateur d'activités.

- Nous représentons les mécanismes de collaboration de ce serveur et de ce consommateur par un automate asynchrone à n+1 états, deux entrées et une sortie. Les entrées correspondent aux événements "entrée pleine" et "entrée vide" et la sortie aux demandes du régulateur. Chaque état représente le nombre d'entrées vides dont disposent éventuellement les émulateurs à l'issue d'un événement "entrée pleine" ou "entrée vide".

- Cet automate est donné ci-aprés. De plus, nous disons que le régulateur d'activités est un régulateur asynchrone.



BLOCAGES ET DEMANDES
DU REGULATEUR (n=3)

C - TERMINAISONS ANORMALES DES ACTIVITES ELEMENTAIRES

- Une activité élémentaire d'entrée/sortie ou de ligne peut se terminer anormalement par suite d'incidents survenus au processeur d'entrée/sortie ou de ligne, au cours de son déroulement. Dans ce cas, elle doit être réamorcée.

I - TERMINAISONS ANORMALES DES ACTIVITES ELEMENTAIRES D'ENTREE/SORTIE:

- La terminaison anormale d'une activité élémentaire d'entrée/sortie se concrétise par une image de données incorrectement entrée ou sortie (carte lue avec perforations multiples, magasin d'alimentation d'un lecteur de cartes vide, etc...).

- Dans ce cas, l'activité élémentaire, avant de se terminer, déclenche l'événement "terminaison entrée/sortie anormale" sans libérer son tampon image. La libération de cet événement entraîne l'émission d'un message d'erreur à la console opérateur. Lorsque l'opérateur répond, pour signifier que le processeur d'entrée/sortie a été remis en état de fonctionnement, l'événement "demande activité d'entrée/sortie" est déclenché. Il faut noter que les frappes de reprise pour un processeur d'entrée/sortie ne doivent être validées que lorsqu'il est en erreur.

II - TERMINAISONS ANORMALES DES ACTIVITES ELEMENTAIRES DE LIGNE:

- La terminaison anormale d'une activité de ligne élémentaire se concrétise soit par la détérioration (bruit de ligne), soit par la perte (coupure de ligne) d'un message de données ou de contrôle. Dans le premier cas, l'activité de ligne élémentaire se déroule complètement et signale l'incident. Par contre, dans le second cas, elle se déroule incomplètement et l'incident n'est pas signalé. De ce fait, un site considéré comme récepteur doit pouvoir détecter, dès que possible, les activités de ligne élémentaires qui se déroulent incomplètement, afin de demander au site émetteur de réamorcer leur déroulement.

1)- DETECTION PAR LE SEQUENCEMENT DES ACTIVITES ELEMENTAIRES:

- Si plusieurs fichiers d'images de données sont transmis en parallèle d'un site (1) vers un site (2), les activités de ligne élémentaires accédant l'une aprés l'autre au processeur ligne peuvent être numérotées séquentiellement sur le site (1). Dans ces conditions, si deux activités élémentaires, qui se terminent consécutivement sur le site (2), ne sont pas numérotées séquentiellement, la terminaison incomplète d'une ou plusieurs activités élémentaires est mise en évidence, et le site (2) peut en avertir le site (1).

- Pour cela, chaque régulateur de ligne affecte à chacune des activités de de ligne élémentaires, avant d'amorcer son déroulement, un numéro d'accés au processeur ligne (ce qui se traduit par la numérotation des messages de contrôle et de données).

- Dans la mesure où elle se déroule normalement, avant de libérer le tampon message du site récepteur, une activité de ligne élémentaire compare

son numéro d'accés au processeur ligne avec un numéro témoin situé sur le site récepteur. Si les numéros sont identiques, elle incrémente le numéro témoin et libère le tampon message du site récepteur. Si les numéros sont différents, elle procède comme une activité de ligne qui se termine complètement mais anormalement; c'est à dire, qu'elle déclenche l'événement "terminaison ligne anormale" sans libérer le tampon message.

- L'événement "terminaison ligne anormale" a pour effet, lorsqu'il est libéré, d'interrompre sur le site récepteur le transfert des fichiers d'images de données en réception comme en émission. Ceci se concrétise par l'invalidation du déclenchement des événements des activités de ligne, et l'invalidation du régulateur de ligne uniquement pour ces mêmes activités. L'invalidation du régulateur de ligne n'est pas impérative. Nous justifierons ce choix par aprés. De plus, cet événement provoque le déclenchement du contrôleur de début de reprise en ligne.

a)- Contrôleur de début de reprise en ligne:

- Lorsqu'il est réveillé par le régulateur de ligne, le contrôleur de début de reprise en ligne déclenche, depuis le site récepteur vers le site émetteur, l'événement "séquence interrompue". Cet événement est accompagné du numéro de la dernière activité de ligne élémentaire qui s'est terminée normalement sur le site récepteur.

- Sur le site émetteur, la libération de l'événement "séquence interrompue" provoque, avant le déclenchement du contrôleur de fin de reprise en ligne, la remise en file de ligne de toutes les activités de ligne qui se sont bloquées anormalement sur le site récepteur, et la réactualisation des numéros d'accés au processeur ligne.

- Le message de contrôle correspondant à l'événement "séquence interrompue" est appelé message de début de reprise en ligne.

b)- Contrôleur de fin de reprise en ligne:

- Réveillé en priorité par le régulateur de ligne, le contrôleur de fin de reprise en ligne déclenche depuis le site émetteur vers le site récepteur, l'événement "reprise séquence". Celui-ci est également accompagné du numéro de la dernière activité de ligne élémentaire qui s'est terminée normalement sur le site émetteur.

- La libération de l'événement "reprise séquence", sur le site récepteur, provoque, au niveau de celui-ci, la remise en file de ligne des activités de ligne qui se seraient éventuellement bloquées anormalement sur le site émetteur, avec la réactualisation des numéros d'accés au processeur ligne. De plus, elle entraîne la revalidation du transfert des fichiers d'images de données, en réception comme en émission.

- Le message de contrôle correspondant à l'événement "reprise séquence" est appelé message de fin de reprise en ligne.

2)- DETECTION SUR DEMANDE DU SITE RECEPTEUR:

- La détection de la terminaison incomplète d'une activité de ligne élémentaire sur un site récepteur, telle que nous l'avons vue ci-dessus, est efficace dans la mesure où une autre activité de ligne élémentaire se termine normalement aprés celle-là.

- Supposons que n fichiers d'images de données ($n \geq 1$) soient transférés en parallèle d'un site à l'autre. Le régulateur du site émetteur a, au plus, à un moment donné, à amorcer le déroulement de n activités de ligne élémentaires. Dans ce cas, le site récepteur n'a plus de message de données à solliciter. Si les déroulements de ces n activités élémentaires se terminent alors tous incomplètement, la transmission des n fichiers d'images de données est définitivement bloquée.

- En effet, le site récepteur n'ayant reçu aucun message de données n'en sollicite pas d'autre, et par suite le site émetteur est bloqué. En particulier, lors du transfert d'un seul fichier d'images de données, ce type de blocage se produit systématiquement dès qu'une activité élémentaire de ligne se termine incomplétement.

- Pour que le site récepteur puisse, dans ce cas, détecter les terminaisons incomplètes, chaque activité élémentaire de ligne, qui se termine normalement, initialise un délai sur le site récepteur pour l'activité de ligne suivante. Si ce délai vient à s'écouler, avant qu'une activité de ligne ait pu le réinitialiser, l'événement "terminaison ligne anormale" est déclenché. Ce délai est fonction du temps maximum que peut mettre le site émetteur pour remplir un tampon message; c'est à dire, qu'il dépend du processeur d'entrée le plus lent et du taux de compression maximum des images de données.

- Dans l'hypothèse, où les processeurs en entrée seraient homogènes en performance, nous pourrions envisager la reprise des terminaisons anormales essentiellement sur délai. Dans ce cas, les activités élémentaires de ligne n'auraient plus la possibilité de déclencher l'événement "terminaison ligne anormale". Toute terminaison anormale serait considérée comme étant une terminaison incomplète. Mais, si les processeurs en entrée ne sont pas homogènes en performance, en cas de terminaison anormale, les plus rapides seront pénalisés par les plus lents. Reste évidemment à estimer le taux de fiabilité du processeur ligne.

III - TERMINAISONS ANORMALES DES CONTROLEURS DE REPRISE EN LIGNE:

- Les contrôleurs de début et de fin de reprise en ligne peuvent aussi se dérouler anormalement.

1)- DETERIORATION DE L'EVENEMENT "SEQUENCE INTERROMPUE":

- Si un événement "séquence interrompue" est détérioré lors de son transfert d'un site (1) vers un site (2), le transfert des fichiers d'images de données, sur ce dernier site, est invalidé en réception comme en émission; puis le contrôleur de début de reprise en ligne est déclenché pour qu'il achemine l'événement "séquence interrompue" vers le site (1). Dès lors sur le site (1), le transfert des fichiers d'images de données est revalidé, et le contrôleur de fin de reprise en ligne est déclenché.

- Lorsque l'événement "reprise séquence" parvient alors au site (2), accompagné du numéro de la dernière activité élémentaire de ligne qui s'est terminée normalement sur le site (1), les activités de ligne, qui se sont bloquées anormalement sur le site (1), sont remises dans la file de ligne du site (2), comme elles l'auraient été si l'événement "séquence interrompue" n'avait pas été détérioré.

- Par conséquent, en invalidant ainsi, sur un site considéré comme récepteur, le transfert des fichiers d'images de données en émission, lorsqu'une activité élémentaire de ligne s'y termine anormalement, le transfert des messages de début de reprise en ligne et de fin de reprise en ligne est réduit, ainsi que l'émission inutile des autres messages.

2)- PERTE DES EVENEMENTS "SEQUENCE INTERROMPUE" ET "REPRISE SEQUENCE":

- En prévision que l'événement "séquence interrompue" ou "reprise séquence" puisse être perdu, le contrôleur de début de reprise en ligne initialise un délai dès qu'il est réveillé. Lorsque le délai est écoulé (événement "délai ligne écoulé"), si le transfert des fichiers d'images de données est toujours invalidé, le contrôleur de début de reprise en ligne est redéclenché.

- Ainsi, sur un site (1), dès le moment où le contrôleur de début de reprise en ligne est réveillé, suite à la terminaison anormale d'une activité élémentaire de ligne, il peut être réveillé consécutivement, plusieurs fois, sur délai. Dans ce cas, à chacun de ses réveils consécutifs les événements "séquence interrompue", qu'il transfère du site (1) vers le site (2), doivent être numérotés en séquence, et les événements "reprise séquence", en réponse à chacun des précédents, doivent porter le même numéro, sinon le site (1) peut être conduit à accepter plusieurs messages de contrôles et de données identiques.

- En effet, supposons que le site (2) ait tardé à fournir au site (1) l'événement "reprise séquence", et qu'au moment où celui-ci est acheminé, le site (1) transfère vers le site (2) un second événement "séquence interrompue". Les deux événements "séquence interrompue" sont accompagnés du numéro de la dernière activité élémentaire de ligne qui s'est terminée normalement sur le site (1). Ce numéro est le même pour les deux événements. Par suite, sur le site (2), les activités de ligne, qui se sont bloquées anormalement sur le site (1), sont remises deux fois dans la file de ligne. Si entre les deux événements "séquence interrompue" l'une ou plusieurs d'entre elles ont été réveillées, le site (1) reçoit deux fois les mêmes messages.

- Par conséquent, le site(1) ne doit accepter des messages de contrôle et de données, aprés avoir émis consécutivement plusieurs événements "séquence interrompue", que lorsqu'il reçoit un événement "reprise séquence" portant un numéro identique à celui que portait le dernier événement "séquence interrompue" qu'il a émis.

D - PROTOCOLE DE COMMUNICATION

- L'ensemble des messages de contrôle et de données, qui ont été précédemment définis avec leur chronologie dans le temps, constitue un protocole de communication.

- Dans ce protocole, tout message de données, construit sur un site considéré comme émetteur, est transféré uniquement aprés avoir été explicitement sollicité par le site récepteur.

- Comme les divers couples d'activités de ligne et de décompression, qui procèdent chacun au transfert d'un fichier d'images de données, disposent les uns et les autres de leurs propres tampons messages, les sollicitations des messages de données, pour des fichiers d'images de données différents, sont indépendantes. Elles se font au rythme suivant lequel, les messages de données sont décompressés, et plus précisément au rythme suivant lequel, les images de données décompressées peuvent être éditées.

- Par conséquent, ce protocole est particulièrement adapté lorsque les processeurs de sortie ne sont pas homogènes en performance. De plus, sa mise en oeuvre est simple. cependant, il présente l'inconvénient d'échanger au moins autant de messages de contrôles que de messages de données. Le trafic des premiers ralentit celui des seconds, lorsque des fichiers d'images de données sont transmis, d'un site à l'autre, dans les deux sens de communication.

III - PARTIE

PROTOCOLE DE COMMUNICATION
AVEC
SOLLICITATIONS GLOBALES
DES
MESSAGES DE DONNEES

- En admettant que toute activité de ligne, aprés avoir transféré un message de données, ne peut pas libérer son tampon message sur le site émetteur tant qu'elle ne dispose pas d'un autre tampon message sur le site récepteur, nous avons construit un protocole de communication dans lequel les messages de données sont sollicités individuellement.

- Si maintenant, nous admettons que toute activité de ligne doit pouvoir libérer son tampon message sur le site émetteur dès qu'elle libère celui du site récepteur, nous pouvons envisager de construire un autre protocole dans lequel les messages de données peuvent être sollicités globalement.

- Pour cela, nous supposons à priori qu'une seule file de tampons messages est à la disposition des divers couples d'activités de ligne et de décompression.

I - SOLLICITATION SANS CONTRAINTE DES MESSAGES DE DONNEES:

- Considérons n couples d'activités de ligne et de décompression ($n \geq 2$) qui ont en commun T tampons messages avec T supérieur ou égal à 2n. Les activités de ligne accédant consécutivement au processeur ligne, ces tampons messages sont remplis un à un.

- Lorsqu'une activité de ligne a déposé un message de données dans l'un d'eux, il est mis à la disposition de l'activité de décompression avec laquelle elle collabore. Dès ce moment, l'activité ligne peut être déclenchée

uniquement si un autre tampon message vide peut lui être réservé. Mais normalement, le nombre de tampons messages vides doit alors être supérieur ou égal à n.

- En effet, supposons qu'il reste exactement n tampons messages vides. La libération de chacun des T-n autres tampons, qui sont à la disposition des activités de décompression, a été accompagnée du déclenchement de l'activité de ligne qui l'avait rempli. Par conséquent, les n activités de ligne sont alors validées pour transférer chacune un message de données; et dès que l'une d'elles remplit l'un des n tampons, elle ne peut pas en disposer d'un autre, car ils sont tous réservés.

- Par suite, si nous voulons que chaque activité de ligne soit déclenchée dès qu'elle libère un tampon message sur le site récepteur, nous devons envisager, chaque fois que le nombre de tampons messages vides passe de n+1 à n, de suspendre les n activités de ligne, sur le site émetteur, avant qu'elles aient terminé de transmettre chacune un message de données.

1)- CONTROLEURS D'ARRET TOTAL DE TRAFIC ET DE REPRISE TOTALE DE TRAFIC:

- Pour suspendre à temps les activités de ligne, dès que le nombre de tampons messages vides passe de n+1 à n, le contrôleur d'arrêt total de trafic est déclenché sur le site récepteur pour qu'il transfère l'événement "trafic totalement suspendu".

- Lorsque l'événement "trafic totalement suspendu" est libéré sur le site émetteur, le régulateur de ligne, sur celui-ci, est invalidé pour réveiller les activités de ligne.

- Lorsque le nombre de tampons messages vides repasse de n à n+1, le contrôleur de reprise totale de trafic est déclenché à son tour sur le site récepteur. Dès qu'il est réveillé, il déclenche l'événement "trafic

totalement résumé" pour revalider le régulateur de ligne du site émetteur.

- Les messages de contrôle qui correspondent aux événements "trafic totalement suspendu" et "trafic totalement résumé" sont appelés message d'arrêt total de trafic et message de reprise totale de trafic.

- Il faut cependant préciser que le contrôleur d'arrêt total de trafic doit être réveillé avant que ne le soient l'activité de ligne qui a provoqué son déclenchement, et toute activité de ligne qui se bloque après celle-ci, car les n activités de ligne doivent être suspendues sur le site émetteur avant qu'elles ne libèrent chacune leur tampon message sur celui-ci. Ceci peut être simplement réalisé, si la file de ligne est gérée par le régulateur de ligne suivant le principe du "premier entré premier sorti". Le contrôleur d'arrêt total de trafic est alors déclenché avant que ne le soit l'activité ligne qui a provoqué son déclenchement.

2)- CONTROLEURS D'ARRET DE TRAFIC ET DE REPRISE DE TRAFIC:

- Maintenant, pour éviter qu'une activité de décompression puisse disposer de trop de tampons messages au détriment des autres activités de décompression, nous pouvons admettre que toute activité de ligne l_i, i variant de 1 à n, doit être suspendue chaque fois que l'activité de décompression d_i, qui lui est associée, retient au moins T_i tampons messages. Chaque nombre limite T_i de tampons messages est supérieur ou égal à 2, et leur somme est égale à T.

- Lorsqu'un $T_i^{ième}$ tampon message est mis à la disposition d'une activité de décompression d_i, le contrôleur d'arrêt de trafic est déclenché, sur le site récepteur, pour qu'il transfère l'événement "trafic suspendu".

- Lorsque l'événement "trafic suspendu" est libéré sur le site émetteur, l'activité de ligne l_i, associée à l'activité de décompression d_i, est suspendue.

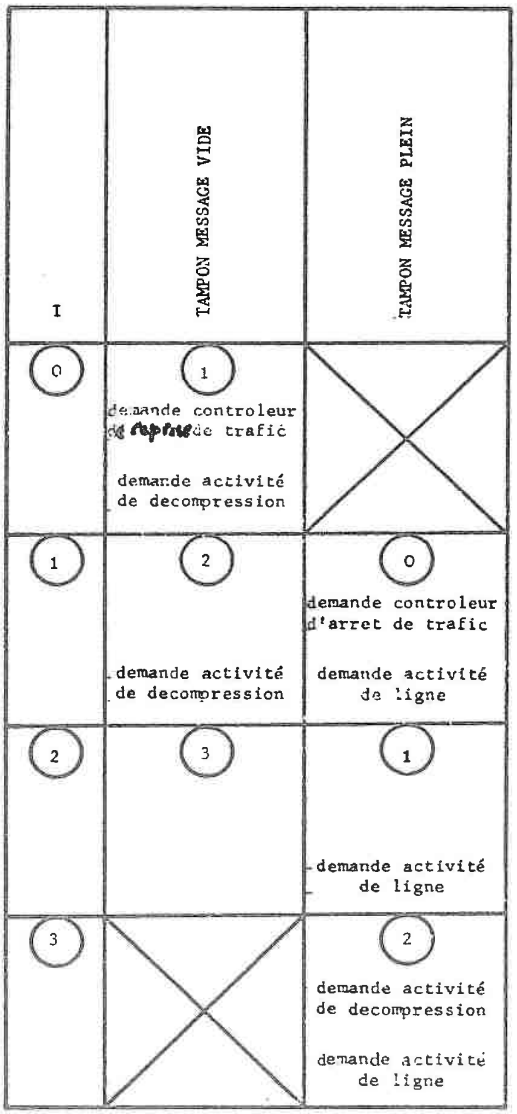
- Si le nombre de tampons messages, dont dispose une activité de décompression d_i, passe de T_i à T_i-1, lorsqu'elle en libère un, le contrôleur de reprise de trafic est déclenché à son tour. Dès qu'il est réveillé, il déclenche l'événement "trafic résumé" pour revalider sur le site émetteur l'activité de ligne l_i associée à l'activité de décompression d_i.

- Les messages de contrôle qui correspondent aux événements "trafic suspendu" et "trafic résumé" sont appelés respectivement message d'arrêt de trafic et message de reprise de trafic.

- Au moment où le contrôleur d'arrêt de trafic est déclenché, l'activité de ligne l_i, i variant de 1 à n, qui a provoqué son déclenchement en libérant un tampon message, est également déclenchée. Si dans chacun de ces cas, le contrôleur d'arrêt de trafic est réveillé avant l'activité de ligne, l'activité de décompression d_i associée à cette dernière n'immobilisera jamais plus de T_i tampons messages. En effet, de cette taçon, l'activité de ligne est suspendue sur le site émetteur avant qu'elle libère son tampon message sur celui-ci.

- Dans ces conditions, les contrôleurs d'arrêt total de trafic et de reprise totale de trafic deviennent inutiles, et chaque couple d'activités de ligne et de décompression peut disposer de sa propre file de tampons messages. La gestion de ceux-ci se trouve alors simplifiée.

- Par suite, nous associons toujours à chaque couple C_i d'activités de ligne et de décompression, i variant de 1 à n, un automate asynchrone à T_i+1 états, deux entrées et deux sorties. Chaque état représente le nombre de tampons messages vides qui sont éventuellement à la disposition de l'activité de ligne l_i, aprés la libération d'un tampon message. Ses entrées correspondent aux événements "tampon message plein" et "tampon message vide". Et ses sorties sont assimilées aux demandes de l'activité de ligne avec éventuellement celles du contrôleur d'arrêt de trafic, et aux demandes de l'activité de décompression avec éventuellement celles du contrôleur de reprise de trafic. Un tel automate est donné ci-aprés.



BLOCAGES ET DEMANDES
D'UN COUPLE D'ACTIVITES DE LIGNE ET DE DECOMPRESSION (Ti=3)

- En cas de terminaison anormale, les activités de ligne élémentaires sont réamorcées comme précédemment. Par contre, pour amorcer le transfert d'un fichier d'images de données, il n'est plus nécessaire d'initialiser l'activité de ligne sur le site récepteur.

II - SOLLICITATION GLOBALE DES MESSAGES DE DONNEES:

- Considérons plusieurs activités de ligne qui accèdent consécutivement au processeur ligne sur un même site. Celles qui acheminent un message de données sont déclenchées, sur le site récepteur, dans l'ordre où elles se bloquent en libérant leur tampon message sur celui-ci.

- Si les activités de ligne sont alors réveillées dans l'ordre de leur déclenchement, les tampons messages, sur le site émetteur, sont libérés dans l'ordre suivant lequel elles avaient été réveillées sur celui-ci.

- Mais dans ces conditions, au lieu que chaque activité ligne libère son second tampon message, nous pouvons envisager que ceux-ci soient libérés globalement par une unique activité que nous nommons activité d'acquittement.

1)- ACTIVITE D'ACQUITTEMENT:

- Chaque activité élémentaire d'émission de message, comme toute activité élémentaire, a reçu, pour se dérouler, un numéro d'accés au processeur ligne, dans l'éventualité où elle se terminerait anormalement. Ce numéro d'accés, comme tout numéro d'accés, est conservé sur le site émetteur avec le nom de l'activité élémentaire à laquelle il est affecté.

- Comme le relâchement de tout numéro d'accés entraîne le relâchement de tous ceux qui lui sont inférieurs, l'activité d'acquittement détermine les tampons messages qu'elle doit libérer, sur le site émetteur, en comparant le plus petit numéro d'accés conservé sur celui-ci avec le numéro d'accés de la dernière activité de ligne élémentaire qui s'est terminée normalement, sur le site récepteur, avant qu'elle y soit réveillée. Pour cela, elle transfère ce second numéro d'accés, du site récepteur vers le site émetteur, par l'intermédiaire d'un message de contrôle appelé message d'acquittement.

2)- PERTE DES MESSAGES D'ACQUITTEMENT:

- Considérons deux activités de ligne élémentaires qui se sont terminées sur un même site, et désignons par p et q les numéros d'accés au processeur ligne qu'elles ont respectivement reçus sur l'autre site, en sachant que p est inférieur à q. En admettant, qu'une activité élémentaire d'acquittement ait été amorcée à la terminaison de chacune d'elles, la perte éventuelle du premier message d'acquittement ne présente aucun inconvénient.

- En effet, le numéro d'accés q, qui est retenu dans le second message d'acquittement, sous entend implicitement que l'activité de ligne élémentaire de numéro d'accés p s'est terminée normalement.

- Par suite, les activités élémentaires d'acquittement ne reçoivent pas de numéro d'accés au processeur ligne lorsqu'elles sont amorcées.

3)- REVEILS DE L'ACTIVITE D'ACQUITTEMENT:

- Sur un site considéré comme récepteur, l'activité d'acquittement doit être demandée au moins chaque fois qu'une activité ligne libère un tampon

message sur celui-ci, aprés y avoir déposé un message de données. Toutefois, elle ne peut pas être déclenchée chaque fois qu'elle est demandée, car ceci reviendrait à déclencher les activités de ligne pour qu'elles libèrent elles mêmes leur second tampon message.

- Par conséquent, si l'activité d'acquittement a été demandée par une ou plusieurs activités de ligne, elle est directement réveillée par le régulateur de ligne dès que la file de ligne est vide.

4)- ACQUITTEMENT GENERALISE:

- Pour que les tampons messages du site émetteur soient malgré tout rapidement libérés, nous admettons en plus que chaque activité de ligne élémentaire, qui est amorcée sur le site récepteur, peut procéder également à leur libération. Pour cela, le numéro d'accés de la dernière activité de ligne élémentaire, qui s'est terminée normalement sur le site récepteur, est inséré par le régulateur de ligne dans le message de contrôle ou de données qu'elle transfère.

III - CONTRAINTES D'UTILISATION DE LA SOLLICITATION GLOBALE:

- En permettant à chaque activité de ligne de libérer les tampons messages des autres activités de ligne, le transfert des messages de données entre les deux sites est optimisé. Toutefois, cette méthode n'est concevable qu'à condition que chaque activité de ligne puisse, aprés avoir transféré un message de données, libérer son tampon message sur le site émetteur, dès qu'elle libère celui du site récepteur. Or ce principe n'est applicable

que si chaque_processeur_de_sortie est au_moins_aussi_performant que le processeur_en entrée_qui_lui_est_associé; sinon, l'activité de ligne, qui procède au transfert d'un fichier d'images de données de l'un vers l'autre, serait continuellement suspendue et résumée. L'alternance des messages d'arrêt et de reprise de trafic serait plus coûteuse que les messages de sollicitation.

- L'arrêt et la reprise d'une activité de ligne ne doivent être envisagés qu'en cas d'incident sur le processeur de sortie. Pour cette raison, nous admettons que toute activité de décompression doit disposer au moins de deux tampons messages.

- Les fichiers d'images de données, lus à partir d'un ou plusieurs lecteurs de cartes d'un terminal, sont rangés au niveau du site central sur des mémoires de masse. Par contre ceux, qui sont acheminés vers le terminal, sont édités sur des imprimantes ou des perforateurs de cartes non homogènes en performances. Dans ces conditions, il est judicieux d'utiliser le protocole, avec sollicitations individuelles des messages de données, dans le sens terminal-ordinateur central, et celui, avec sollicitations globales des messages de données, dans l'autre sens.

IV - PARTIE

ACCES DES ACTIVITES
AU PROCESSEUR
ARITHMETIQUE ET LOGIQUE

A - COLLABORATION DES PROCESSEURS

I - EVENEMENTS TECHNOLOGIQUES:

- Lors du déroulement d'une activité élémentaire d'entrée/sortie, le prélèvement ou le dépot d'une image de données, dans un tampon image, peuvent être assuré par le processeur d'entrée/sortie, en collaboration avec le processeur arithmétique et logique. Dans ce cas, le processeur d'entrée/sortie est dit être à accés indirect mémoire. L'image de données est transférée caractère par caractère. Ceux-ci transitent par un registre, dit registre d'interface, commun aux deux processeurs (nous n'envisageons pas l'éventualité de plusieurs registres).

- Pour transférer un caractère chaque processeur exécute une procédure (suite d'opérations logiques programmées, micro-programmées ou cablées): l'une pour déposer un caractère dans le registre d'interface, et l'autre pour l'y prélever. Nous désignons l'exécution d'une telle procédure sous le nom de tâche.

- Le déplacement d'une image de données peut également être réalisé uniquement par le processeur d'entrée/sortie. Dans ce cas, il est dit être à accés indirect mémoire. Il remplit ou vide le tampon image en une seule tâche.

- Quel que soit le mode d'accés du processeur d'entrée/sortie, toute tâche sur celui-ci est validée par une tâche du processeur arithmétique et logique. Toutefois, la validation des tâches est différente en entrée et en sortie. Les tâches d'un processeur d'entrée sont validées globalement, dès la première tâche déroulée par le processeur arithmétique et logique (un lecteur de cartes est initialisé pour lire les 80 caractères d'une carte

perforée). Par contre, les tâches d'un processeur en sortie sont validées une à une par chacune des tâches du processeur arithmétique et logique, à l'exception de la dernière qui libère le tampon image. (une imprimante édite les caractères au fur et à mesure qu'elle les reçoit).

- Chaque fois qu'une tâche se termine sur un processeur d'entrée/sortie, elle libère, suivant le mode d'accés de celui-ci, le registre d'interface ou le tampon image. Cet événement, appelé événement technologique, est concrétisé par un signal qui est positionné par le processeur d'entrée/sortie. A l'issue d'un tel événement, le processeur arithmétique et logique déroule la tâche complémentaire.

- Ainsi une activité élémentaire d'entrée/sortie se décompose en $2p+1$ tâches ($p \geq 1$) qui doivent se dérouler alternativement sur les deux processeurs.

- Cependant, le processeur arithmétique et logique peut perdre les événements technologiques d'un processeur d'entrée à accés indirect mémoire. Ceci tient au fait que les tâches de celui-ci sont validées globalement. Elles se déroulent consécutivement, sans attendre que les caractères soient prélevés du registre d'interface. Par contre, il a toujours le contrôle des événements technologiques d'un processeur d'entrée à accés direct mémoire. Lorsqu'une tâche se termine sur un tel processeur, la tâche complémentaire sur le processeur arithmétique et logique consiste à libérer le tampon image. Or une nouvelle activité élémentaire d'entrée ne peut pas être amorcée tant que celui-ci n'est pas libéré.

- Le processeur ligne peut également accéder directement ou indirectement à un tampon message. La réception ou l'émission d'un message de contrôle ou de données s'opèrent de la même façon que l'entrée ou la sortie d'une image de données. Un processeur arithmétique et logique collabore avec le processeur ligne, comme avec un processeur d'entrée/sortie. Toutefois, lors de la réception d'un message, les tâches du processeur ligne sont validées par celles du processeur arithmétique et logique du site émetteur. Par conséquent, le processeur arithmétique et logique du site récepteur n'a jamais le contrôle

des événements technologiques en réception. Il peut les perdre, même si le processeur ligne est à accés direct mémoire.

- Nous appelons respectivement, tâche de début d'entrée/sortie et tâche de fin d'entrée/sortie, la première et la dernière tâche d'une activité élémentaire d'entrée/sortie, qui sont déroulées par le processeur arithmétique et logique; et nous appelons respectivement, tâche de début d'émission en ligne et tâche de fin d'émission en ligne, la première et la dernière tâche d'une activité élémentaire de ligne, qui sont déroulées par le processeur arithmétique et logique d'un site considéré comme émetteur. La dernière tâche, d'une activité élémentaire de ligne, qui est déroulée sur le processeur arithmétique et logique, d'un site considéré comme récepteur, est appelée tâche de fin de réception en ligne.

II - SYSTEME D'INTERRUPTION:

- Les événements technologiques sont gérés par un système d'interruption. Celui-ci se compose d'une file d'attente d'événements et d'un régulateur d'événements cablés.

- Ce système d'interruption peut être validé et invalidé, pour libérer les événements technologiques, par toute tâche qui se déroule sur le processeur arithmétique et logique.

- Au moment où il libère un événement, le système d'interruption interrompt la tâche en cours de déroulement, sur le processeur arithmétique et logique, pour amorcer la tâche de début d'interruption. Dès lors, il est automatiquement invalidé.

- La tâche de début d'interruption range dans une file d'attente d'activités,

appelée file des activités interrompues, l'adresse de l'instruction interrompue, le mot d'état du processeur et éventuellement le contenu des registres. Lorsque ces paramètres de déroulement, appelés encore le vecteur d'état de la tâche interrompue, sont sauvegardés, elle amorce la tâche associée à l'événement libéré.

- La tâche interrompue (ou une autre) est alors réamorcée à la fin de la tâche qui traite l'événement, ou à la fin des tâches que cette dernière peut éventuellement entraîner. Pour cela, la dernière tâche, qui se termine, amorce la tâche de fin d'interruption.

- Quand la tâche de fin d'interruption réamorce une tâche interrompue, aprés avoir restauré son vecteur d'état, le système d'interruption est automatiquement revalidé.

- Nous appelons, tâche d'interruption, l'ensemble des tâches qui s'imbriquent dans une autre tâche, suite à la libération d'un événement technologique.

- La file des activités interrompues ne peut être modifiée que par une tâche à la fois. Pour cela, le système d'interruption reste ou est invalidé pendant le déroulement des tâches de début d'interruption et de fin d'interruption. Nous disons alors qu'elles se déroulent en mode protégé.

- Ainsi, les tâches de début d'interruption d'une part, et de fin d'interruption d'autre part, se déroulent selon le modèle d'un serveur et d'un consommateur qui auraient en commun les entrées de la file des activités interrompues.

- Compte tenu des restrictions imposées pour les tâches de début et de fin d'interruption, nous pouvons convenir que le système d'interruption est revalidé, aprés la libération d'un événement, uniquement lorsqu'une tâche interrompue est réamorcée, ou au cours du déroulement de la tâche d'interruption. Dans le premier cas, les tâches déroulées sur le processeur arithmétique et logique sont imbriquées à un seul niveau, et par suite la file des activités interrompues a une seule entrée. Dans le second cas,

ces tâches sont imbriquées à plusieurs niveaux, et la file des activités interrompues est à entrée multiple.

- Nous allons voir, pour chacune de ces deux possibilités, comment peuvent s'effectuer les réveils des activités.

B - REGULATION DES EVENEMENTS LOGIQUES

- Pour éviter que les deux entrées d'un automate asynchrone soient accédées simultanément, nous avons convenu de déclencher tout événement logique réalisé par une activité élémentaire. De cette façon, lorsque l'un d'eux est libéré, son émulateur peut modifier l'état de l'automate et réaliser, au besoin, d'autres événements sans être interrompu par un émulateur concurrent.

- Etant donné que, sur un processeur arithmétique et logique, une tâche peut être interrompue, au profit d'une autre tâche, uniquement par le système d'interruption, nous pouvons éviter de déclencher les événements logiques, si celui-ci est invalidé au cours du déroulement des émulateurs. Le système d'interruption assure ainsi le retardement des événements logiques et leur régulation.

C - REGULATION DES ACTIVITES

- Pour tout couple d'activités d'entrée/sortie et de compression/décompression, l'émulation d'un événement "tampon image plein" ou "tampon image vide" peut provoquer le réveil de l'activité d'entrée/sortie avec celui de l'activité de compression/décompression. Dans ce cas, les deux activités élémentaires doivent être amorcées l'une aprés l'autre sur le processeur arithmétique et logique. Comme une activité élémentaire d'entrée/sortie se décompose en $2p+1$ tâches ($p \geq 1$), qui se déroulent alternativement sur le processeur arithmétique et logique et un processeur d'entrée/sortie, il convient d'amorcer à priori celle-ci, si nous voulons que les deux processeurs soient actifs simultanément. Dans ces conditions, l'unique tâche de compression/décompression est amorcée lorsque la tâche de début d'entrée/sortie se termine. De même, une activité de ligne élémentaire est amorcée en priorité lorsqu'elle est en conflit avec une activité de compression/décompression.

- Si nous admettons que tout processeur d'entrée/sortie est global à plusieurs activités d'entrée/sortie, celles-ci doivent être déclenchées lorsqu'elles sont demandées à l'issue d'un événement logique.

- Ainsi, lorsqu'une activité d'entrée/sortie est demandée et validée, le nom de sa procédure de début d'entrée/sortie est rangé dans la file d'entrée/sortie du processeur d'entrée/sortie auquel elle doit accéder. Si cette file était vide auparavant, le régulateur d'entrée/sortie amorce alors sa tâche de début d'entrée/sortie sinon, il attend que le processeur soit disponible.

- Par contre, lorsqu'une activité de compression/décompression est demandée, nous pouvons envisager qu'elle soit immédiatement réveillée, aprés le déclenchement éventuel de l'activité d'entrée/sortie. Mais dans ce cas, une tâche d'interruption peut devenir importante, car si une activité

de compression/décompression dispose de plusieurs tampons images et messages, ses activités élémentaires peuvent se succéder jusqu'à l'épuisement de ceux-ci. Par conséquent, cette méthode ne peut être retenue que si les tâches sur le processeur arithmétique et logique sont imbriquées à plusieurs niveaux. Toutefois, nous verrons que, dans ce cas, le système d'interruption doit être suffisamment évolué.

- Pour que les tâches d'interruption soient aussi courtes que possible, nous devons envisager de déclencher les activités de compression/décompression, lorsqu'elles sont demandées. Nous allons voir comment procéder à leur déclenchement, suivant que les tâches sur le processeur arithmétique et logique sont imbriquées à un ou plusieurs niveaux.

I - TACHES IMBRIQUEES A UN SEUL NIVEAU:

- Si nous convenons que les tâches sur le processeur arithmétique et logique sont imbriquées à un seul niveau, toute tâche d'interruption doit se dérouler entièrement en mode protégé.

1)- DEROULEMENT DES ACTIVITES ELEMENTAIRES D'ENTREE/SORTIE:

- Chaque tâche, d'une activité élémentaire d'entrée/sortie, amorcée à l'issue d'un événement technologique se déroule en mode protégé.

- Lorsque la tâche de fin d'entrée/sortie se termine, le tampon image est libéré avant l'entrée de la file d'entrée/sortie qui était réservée à l'activité élémentaire. Ainsi, les activités d'entrée/sortie et de compression/

décompression sont éventuellement déclenchées avant que le régulateur d'entrée/sortie réveille une nouvelle activité d'entrée/sortie.

- Lorsque l'entrée est libérée, si la file d'entrée/sortie est vide, la tâche interrompue est réamorcée; sinon le régulateur d'entrée/sortie amorce une nouvelle activité élémentaire. La tâche interrompue est alors réamorcée à la fin de la tâche de début d'entrée/sortie. Par suite, lorsqu'une activité élémentaire d'entrée/sortie est amorcée à l'issue d'un événement technologique, elle se déroule, sur le processeur arithmétique et logique, entièrement en mode protégé.

2)- DEROULEMENT DES ACTIVITES DE LIGNE ELEMENTAIRES:

- Chaque tâche, d'une activité de ligne élémentaire, qui est amorcée à l'issue d'un événement technologique, se déroule également en mode protégé.

- Lorsqu'une activité de ligne élémentaire est amorcée par le régulateur de ligne d'un site donné, l'entrée qui lui est réservée dans la file de ligne est libérée dès que sa tâche de fin d'émission en ligne se termine. Par contre, son tampon message ne peut être libéré que par une tâche de fin de réception en ligne.

- Lorsque l'entrée est libérée, si la file de ligne est alors vide, la tâche interrompue est réamorcée; sinon le régulateur de ligne amorce une nouvelle activité de ligne élémentaire. La tâche interrompue est alors réamorcée à la fin de la tâche de début d'émission en ligne. Par suite, lorsqu'une activité de ligne élémentaire est amorcée à l'issue d'un événement technologique, elle se déroule sur le processeur arithmétique et logique entièrement en mode protégé.

3)- DEROULEMENT DES ACTIVITES ELEMENTAIRES DE COMPRESSION/DECOMPRESSION:

- Pour leur déclenchement à l'issue d'une tâche de fin d'entrée/sortie ou de fin de réception en ligne, nous associons aux activités de compression/décompression, une file d'attente d'activités et un régulateur d'activités que nous appelons respectivement file moniteur et moniteur.

- Pour que les tâches d'interruption soient courtes, les tâches de compression/décompression doivent se dérouler en mode non protégé. Par suite une activité de compression/décompression, qui est déclenchée, ne peut être réveillée que lorsque la tâche interrompue est réamorcée. Pour cela, le moniteur est amorcé chaque fois qu'une tâche interrompue se termine.

- Lorsqu'une tâche de compression/décompression se termine, si son tampon image est libéré, l'activité de compression/décompression peut être demandée avec l'activité d'entrée/sortie. Par contre, si son tampon message est libéré, elle peut être demandée avec l'activité de ligne. Dans ces cas, l'activité d'entrée/sortie ou de ligne est déclenchée et éventuellement immédiatement réveillée. Or nous avons convenu qu'une tâche interrompue est réamorcée à la fin d'une tâche de début d'entrée/sortie ou de début d'émission en ligne. Pour rester cohérents avec ce principe, nous admettons qu'une tâche de compression/décompression se termine en déclenchant par programme un événement technologique. Lorsque celui-ci est libéré, le nom de la procédure qui décrit le moniteur est rangé dans la file des activités interrompues; puis le tampon image ou message est libéré. De cette façon, l'activité de compression/décompression est systématiquement déclenchée chaque fois qu'elle est demandée, et le moniteur est alors amorcé, que les activités soient déclenchées ou non, comme une tâche interrompue. Par suite, les activités élémentaires d'entrée/sortie et de ligne se déroulent toutes,entièrement en mode protégé.

- Etant donné que les activités élémentaires d'une activité de compression/décompression se déroulent en séquence, la file moniteur doit avoir autant d'entrées qu'il y a d'activités de compression et de décompression.

- Si la file moniteur a une entrée fixe par activité de compression et de décompression ou si elle est gérée suivant le principe du "premier entré premier sorti" le moniteur peut se dérouler en mode non protégé. Dans ces conditions, lorsque le moniteur est amorcé, si la file moniteur est vide, nous admettons qu'il scrute en permanence ses entrées dans l'attente qu'une activité de compression ou de décompression soit déclenchée.

4)- CAS OU LES PROCESSEURS D'ENTREE/SORTIE SONT LOCAUX:

- Admettons maintenant que chaque processeur d'entrée/sortie soit local à une activité d'entrée/sortie. Dans ce cas, les files d'entrée/sortie avec leur régulateur sont inutiles.

- Par conséquent, nous pouvons envisager d'amorcer les activités élémentaires des activités d'entrée/sortie en mode non protégé. Pour cela, les activités d'entrée/sortie sont toujours déclenchées; mais les noms des procédures de début d'entrée/sortie sont rangés dans la file moniteur.

- Les entrées de la file moniteur peuvent alors être regroupées deux à deux. Chaque couple d'entrées est assigné à un couple d'activités d'entrée/sortie et de compression/décompression. Le moniteur scrute alors les couples d'entrées un à un. Pour un couple d'entrées donné, il consulte en priorité l'entrée réservée à l'activité d'entrée/sortie. De cette façon, pour chaque couple d'activités, les activités élémentaires d'entrée/sortie sont amorcées avant celles de compression/décompression, lorsqu'elles sont en concurrence.

- Comme les taches de début d'entrée/sortie se déroulent en mode non protégé, le moniteur est amorcé dès qu'elles se terminent. De ce fait, les tâches de compression/décompression ne déclenchent plus d'événement technologique. Le moniteur est amorcé dès que leur tampon image ou message est libéré.

- De cette façon, les tâches d'interruption sont plus courtes et moins nombreuses.

5)- CAS OU LES TAMPONS IMAGES SONT LIMITES:

- De plus, admettons que chaque couple d'activités d'entrée/sortie et de compression/décompression dispose uniquement de deux tampons images.

- Dans ce cas, lorsqu'une activité d'entrée/sortie libère un tampon image, elle ne peut pas être déclenchée, tant que l'activité de compression/décompression n'a pas libéré le sien. De même, l'activité de compression/décompression est tributaire de l'activité d'entrée/sortie.

- Dans ces conditions, si nous convenons qu'une activité élémentaire d'entrée/sortie peut amorcer une activité élémentaire de compression/décompression, à l'issue de la tâche début d'entrée/sortie, nous pouvons réserver une seule entrée de la file moniteur par couple d'activités.

- Le moniteur est alors amorcé uniquement à la fin des tâches de compression/décompression. Ainsi, il est amorcé deux fois moins que précédemment, et son rôle est réduit du fait que les entrées de la file moniteur ont été diminuées de moitié.

- Cependant une activité de compression/décompression est toujours déclenchée seule, aprés qu'elle se soit bloquée en libérant un tampon message. Une activité d'entrée/sortie est également déclenchée seule lorsque l'une de ses activités élémentaires se termine anormalement. Si chaque couple d'activités ne dispose que d'une entrée dans la file moniteur, le déclenchement de l'une peut alors annuler le déclenchement de l'autre.

- Pour résoudre ce conflit d'accés aux entrées de la file moniteur, nous convenons que toute activité élémentaire d'entrée/sortie libère son tampon

image, quelle que soit sa terminaison. La terminaison anormale d'une activité élémentaire d'entrée/sortie est alors détectée et signalée par celle qui lui succède. Dans ce cas, cette dernière n'amorce pas l'activité de compression/décompression.

II - TACHES IMBRIQUEES A PLUSIEURS NIVEAUX:

- Si nous convenons que les tâches sur le processeur arithmétique et logique peuvent être imbriquées à plusieurs niveaux, toute tâche d'interruption peut être interrompue au profit d'une autre.

- Dans l'hypothèse, où chaque processeur d'entrée/sortie est global à plusieurs activités d'entrée/sortie, nous admettons que les tâches interrompues sont réamorcées comme précédemment par les activités d'entrée/sortie et de ligne.

1)- DEROULEMENT DES ACTIVITES ELEMENTAIRES DE COMPRESSION/DECOMPRESSION:

- Chaque fois qu'une activité de compression/décompression est demandée à la terminaison d'une tâche de fin d'entrée/sortie ou de fin de réception en ligne, le nom de sa procédure de compression/décompression est rangée dans la file des activités interrompues.

- Ce mécanisme de déclenchement de l'activité de compression/décompression revient à simuler la libération d'un événement technologique qui suspendrait le réveil de la tâche de compression/décompression pour demander l'activité d'entrée/sortie. Dans ces conditions, déclencher l'activité de compression/

décompression équivaut à interrompre la tâche d'interruption. Par conséquent, une tâche interrompue doit être réamorcée à l'issue de la tâche de compression/décompression ainsi retardée.

- Comme précédemment, lorsqu'une tâche de compression/décompression se termine, l'activité de compression/décompression peut être demandée soit avec l'activité d'entrée/sortie, soit avec l'activité de ligne. Comme nous avons convenu qu'une tâche interrompue est réamorcée à la fin d'une tâche de début d'entrée/sortie ou de début d'émission en ligne, nous admettons que l'activité de compression/décompression est systématiquement déclenchée chaque fois qu'elle est demandée.

- Ainsi, une tâche interrompue est toujours réamorcée à l'issue d'une tâche de compression/décompression, à moins qu'une tâche de début d'entrée/sortie ou de début d'émission en ligne soit alors amorcée.

- Par cette méthode, chacune des tâches des activités élémentaires se déroule sur le processeur arithmétique et logique à l'intérieur d'une tâche d'interruption. De ce fait, chaque fois que la file des activités interrompues est vide, une tâche dite d'attente est amorcée.

2)- CAS PARTICULIERS:

- Si un processeur d'entrée/sortie est local à une activité d'entrée/sortie, celle-ci est immédiatement réveillée chaque fois qu'elle est demandée.

- Si maintenant tout couple d'activités dispose uniquement de deux tampons images ou de deux tampons messages, nous pouvons admettre qu'une activité élémentaire de compression/décompression puisse être amorcée, soit par une activité élémentaire d'entrée/sortie à la fin de la tâche de début d'entrée/sortie, soit par une activité de ligne élémentaire à la fin de la tâche de début d'émission en ligne. En effet, dans ce cas, l'activité de compression/

décompression est toujours demandée soit avec l'activité de ligne, soit avec l'activité d'entrée/sortie. Une tâche interrompue est alors réamorcée à l'issue d'une tâche de compression décompression.

- Si une activité élémentaire d'entrée/sortie se termine anormalement, le tampon image n'est pas libéré, et la tâche de début d'entrée/sortie est réamorcée sans que la tâche de compression/décompression le soit. Il en est de même pour les activités de ligne élémentaires qui se terminent anormalement.

- En évitant le déclenchement des activités de compression/décompression, la tâche de fin d'interruption est moins souvent amorcée et son rôle est réduit du fait qu'elle a moins d'entrées à consulter dans la file des activités interrompues pour réamorcer une activité.

- Il faut aussi noter que les processeurs d'entrée/sortie peuvent être indifféremment locaux ou globaux.

D - PRIORITE DES ACTIVITES

- Si les processeurs d'entrée/sortie ne sont pas homogènes en performance, les couples d'activités d'entrée/sortie et de compression/décompression, qui dépendent des plus rapides, doivent se dérouler en priorité.

- Dans ce cas, les événements technologiques sont généralement regroupés en classes d'événements suivant les caractéristiques des processeurs d'entrée/sortie qui les génèrent. Si m classes d'événements sont ainsi formées ($m > 1$), celles-ci sont ordonnées de 0 à m-1 suivant l'ordre croissant des performances des processeurs d'entrée/sortie. La classe d'événements de rang i, i variant de 0 à m-1, est dite de niveau de priorité i.

- Lorsque le système d'interruption est validé, si plusieurs événements technologiques sont en attente, il libère l'un des événements de priorité la plus élevée.

- Dans ces conditions, nous classons les couples d'activités d'entrée/sortie et de compression/décompression, et par suite les couples d'activités de compression/décompression et de ligne, sur le même modèle que les événements technologiques. Ainsi, lorsque plusieurs activités élémentaires d'entrée/sortie, de compression/décompression ou de ligne doivent être amorcées ou réamorcées, elles le sont suivant leur priorité. Mais ceci suppose que les files d'attente d'activités soient organisées.

- Nous allons voir l'organisation des files d'attente d'activités suivant que les tâches sur le processeur arithmétique et logique sont imbriquées à un niveau ou à plusieurs niveaux.

I - TACHES IMBRIQUEES A UN SEUL NIVEAU:

1)- FILE MONITEUR:

- La file moniteur est divisée en sous files. Chacune de celles-ci est réservée aux activités d'un niveau de priorité donné. De ce fait, elles sont implicitement hiérarchisées.

- Chaque fois que le moniteur est amorcé, il consulte chacune des sous-files moniteur dans l'ordre décroissant de leur niveau de priorité. Dès qu'il en rencontre une qui est non vide, il amorce l'une des tâches qu'elle retient.

- Pour que le moniteur soit aussi performant que possible, il est préférable qu'il gère chaque sous-file moniteur suivant le principe du "premier entré premier sorti". Nous disons alors que chaque sous-file a une structure de silo.

2)- FILE DE LIGNE:

- Compte tenu de la compression des images de données, si nous admettons que le processeur ligne est au moins aussi performant que tous les processeurs d'entrée réunis, la file de ligne n'a pas besoin d'être décomposée en sous-files.

- Cependant, pour que les activités de ligne soient réveillées dans l'ordre où elles sont déclenchées, il est souhaitable que la file de ligne ait une structure de silo, d'autant plus que cette structure facilite la reprise des activités élémentaires de ligne qui se terminent anormalement.

- En effet, le régulateur de ligne peut alors donner, pour numéro d'accés

au processeur ligne, à toute activité élémentaire qu'il amorce, le numéro de l'entrée du silo qui retient le nom de sa procédure de début d'émission en ligne. Ainsi, si l'activité élémentaire se termine anormalement, le régulateur, pour la remettre en file de ligne, doit seulement réajuster le pointeur de début de silo.

- Si le processeur ligne est moins performant que tous les processeurs d'entrée réunis, la file de ligne est décomposée en sous-files. Chaque sous-file a une structure de silo,pour que les activités de ligne soient réveillées dansl'ordre où elles sont déclenchées.

- Le régulateur de ligne accède à ces sous-files de ligne, comme le moniteur accède aux sous-files moniteur. Toutefois pour prévenir les terminaisons anormales des activités élémentaires de ligne, il dispose d'un silo supplémentaire dans lequel il mémorise le nom des procédures de début d'émission en ligne au fur et à mesure qu'il les amorce.

3)- FILES D'ENTREE/SORTIE:

- Une file d'entrée/sortie est réservée à des activités d'entrée/sortie qui accèdent à un même processeur. Ces activités ayant par conséquent le même niveau de priorité, elle peut avoir une structure de silo ou de pile, c'est à dire, être gérée suivant le principe du "dernier entré premier sorti". Cette dernière structure n'a pas été envisagée pour les sous-files moniteur, car le moniteur devrait alors se dérouler en mode protégé.

II - TACHES IMBRIQUEES A PLUSIEURS NIVEAUX:

- Dans ce cas, la file des activités interrompues est divisée en sous-files comme la file moniteur. Mais du fait que les tâches de début et de fin d'interruption se déroulent en mode protégé, chaque sous-file d'activités interrompues peut avoir une structure de pile ou de silo.

- La file de ligne et les files d'entrée/sortie sont organisées comme précédemment.

III - CRITERES D'IMPLANTATION DES METHODES DE REGULATION DES ACTIVITES:

- Des diverses méthodes présentées pour réguler les activités, laquelle mettre plus particulièrement en oeuvre? Ceci dépend des facilités de programmation qu'offre le processeur arithmétique et logique utilisé et du système d'interruption dont il est équipé.

- Par exemple, si au système d'interruption sont associés deux jeux de registres de travail, l'un pour les tâches en mode protégé et l'autre pour les tâches en mode non protégé, il y a avantage à imbriquer les tâches à un seul niveau.

- Par contre, si au moment de la libération d'un événement technologique, le vecteur d'état de la tâche interrompue est automatiquement sauvegardé et la tâche d'interruption automatiquement amorcée par micro-programme, il est préférable d'imbriquer les tâches à plusieurs niveaux. Les tâches interrompues sont alors, le plus souvent, rangées au sommet d'une pile définie par programme. Dans ce cas, il faut définir autant de piles qu'il y a de niveaux de priorité

pour les activités.

- Maintenant, sur certains processeurs arithmétiques et logiques, chaque tâche a la possibilité d'invalider la libération de tous les événements technologiques,de niveau de priorité inférieur ou égal à un niveau de priorité donné. Dans ce cas, la file des activités interrompues n'a plus besoin d'être décomposée,puisque chaque activité se déroule en se protégeant contre les activités de priorité inférieure ou égale. De plus les activités de compression/ décompression peuvent être réveillées dès qu'elles sont demandées.

CONCLUSION

- Dans l'introduction, la procédure devait répondre aux critères de PERFORMANCE, d'ADAPTATION, et de COMPATIBILITE. Comment ces objectifs sont ils atteints maintenant?

I - PERFORMANCE:

- Dans l'hypothèse où les processeurs d'entrée et de sortie du site terminal sont moins performants que ceux du site central, le protocole de communication doit être asymétrique, pour que les échanges d'information entre les deux sites soient optimisés. C'est le cas de FD-NTR.

- Que les tâches soient imbriquées à un ou plusieurs niveaux, la gestion des files d'activités est réduite au maximum lorsque les processeurs d'entrée et de sortie sont locaux, et que chaque couple d'activités dispose uniquement de deux tampons images ou messages qui lui sont propres. Ces propriétés sont appliquées dans la procédure de l'UNIVAC 9000 et dans celle du DIGITAL PDP 11.

- Les processeurs d'entrée ou de sortie les plus performants ne sont pas pénalisés par les moins performants, si chaque couple d'activités est affecté d'une priorité d'accés au processeur arithmétique et logique. Cette propriété est appliquée dans la procédure du DIGITAL PDP 11. Sa mise en oeuvre a été facilitée par le fait que toute tâche, sur ce matériel, a la possibilité d'invalider la libération des événements technologiques de priorité inférieure ou égale à une priorité donnée.

II - ADAPTATION:

- Suivant le modèle de processeur arithmétique et logique utilisé, l'adaptation de la procédure, à celui-ci, s'opère uniquement au niveau de la file des activités interrompues.

1)- TACHES IMBRIQUEES A PLUSIEURS NIVEAUX:

- Dans le cas du DIGITAL PDP 11, l'imbrication des tâches à plusieurs niveaux est possible du fait que le vecteur d'état, de toute tâche interrompue par le système d'interruption, est sauvegardé automatiquement par micro-programme. Celui-ci le range en mémoire au sommet d'une pile qui a été définie par programme. La restitution d'un vecteur d'état se fait également par micro-programme. Ce dernier est appelé par l'instruction RTI.

2)- TACHES IMBRIQUEES A UN SEUL NIVEAU:

- L'UNIVAC 9000 ayant deux jeux de registre, l'un pour les tâches en mode protégé et l'autre pour les tâches en mode non protégé, la file des activités interrompues est à une seule entrée. Par suite, une file moniteur est nécessaire pour les activités de compression et de décompression.

- Toutefois,dans ce cas, l'émulation des événements,de type "tampon plein" ou "tampon vide",peut être synchrone. C'est ce qui est fait sur l'UNIVAC 9000. Chaque entrée de la file moniteur est assignée à un processeur d'entrée/sortie. Mais ces entrées, au lieu de mémoriser des noms de procédure, mémorisent des

des indicateurs de blocage - Voir régulation synchrone des événements chapitre I -. Cette méthode a l'avantage de réduire au maximum les tâches d'interruption, mais elle présente l'inconvénient d'alourdir le rôle du moniteur, et de façon générale la programmation de la procédure.

- Pour une configuration qui comprend un lecteur de cartes, une imprimante et un perforateur de cartes, la procédure de l'UNIVAC 9000 occupe 12 K-octets de mémoire, et sa vitesse de transmission est garantie jusqu'à 50 K-bauds; tandis que pour la même configuration, celle du DIGITAL PDP 11 occupe 8 K-octets de mémoire, et sa vitesse de transmission est garantie jusqu'à 100 K-bauds.

III - COMPATIBILITE:

- La compatibilité de la procédure avec un autre programme est obtenue facilement lorsque les tâches sont imbriquées à plusieurs niveaux.

- Il suffit, d'une part, de retirer la tâche d'attente qui se résume à une instruction d'attente (WAIT), suivie d'une instruction de branchement à cette dernière, et, d'autre part, d'ajuster les priorités des couples d'activités en fonction des priorités nécessaires à l'autre programme.

- La procédure a été ainsi rendue compatible avec le programme d'acquisition de données. Comme autre exemple de coexistence, nous pouvons concevoir qu'un mini-ordinateur assure les fonctions de terminal de traitement par lots et de concentration de terminaux conversationnels - Voir figures 1 et 2 -.

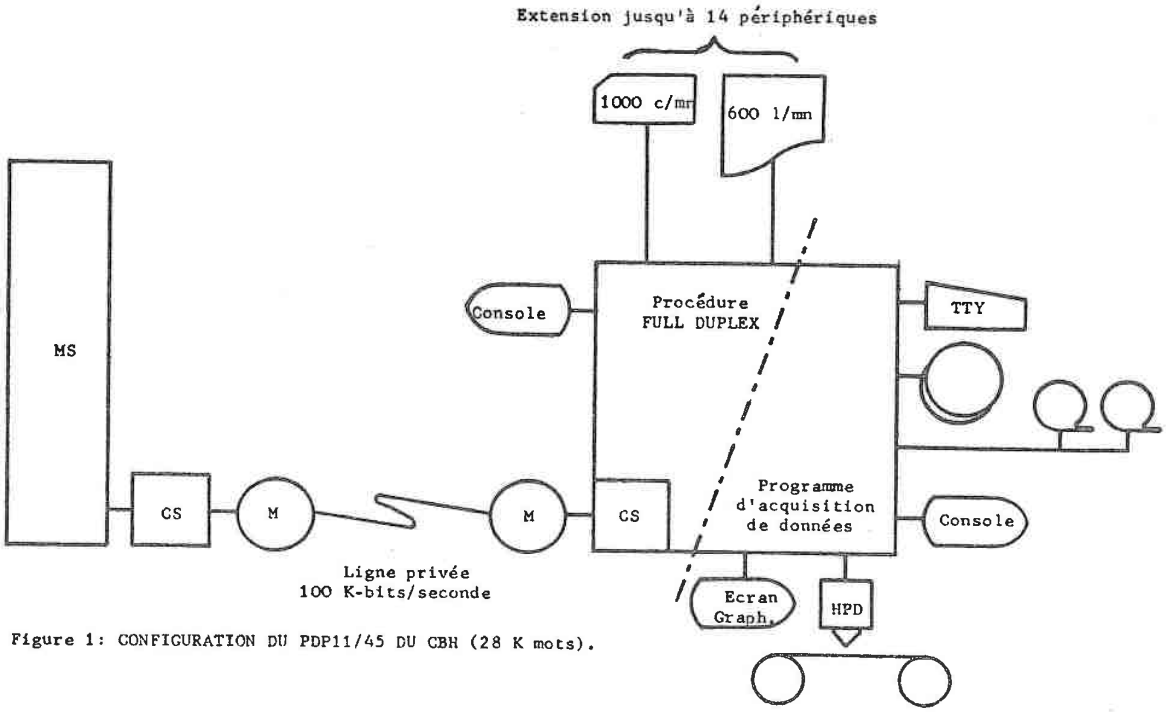


Figure 1: CONFIGURATION DU PDP11/45 DU CBH (28 K mots).

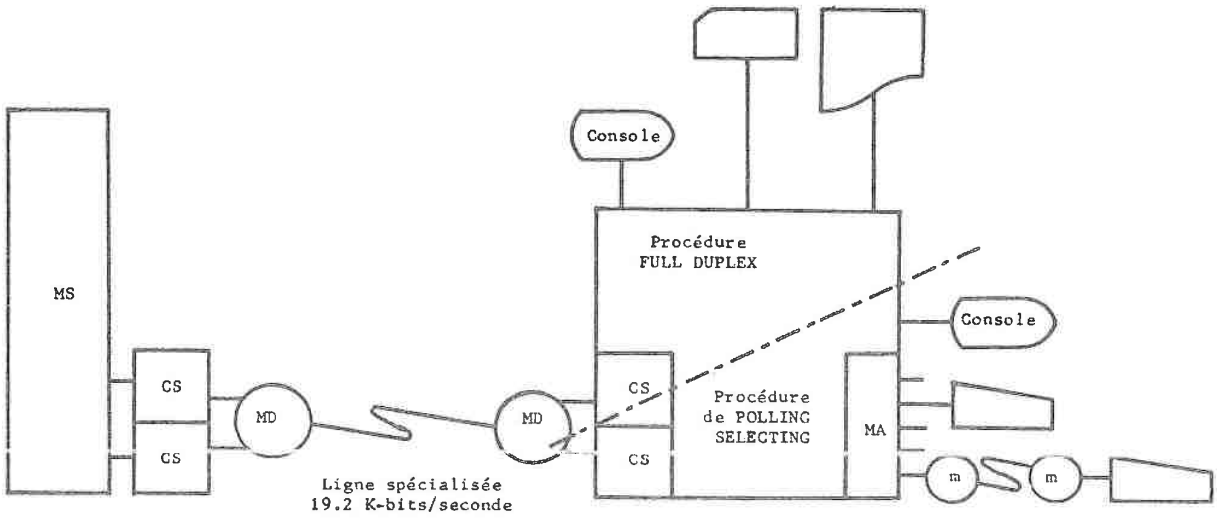


Figure 2: REGROUPEMENT DES FONCTIONS TERMINAL LOURD ET CONCENTRATION DE TERMINAUX COVERSATIONNELS.

m: Modem asynchrone.
M: Modem synchrone.
MD: Modem synchrone duplexeur.

CS: Coupleur de transmition synchrone.
MA: Multiplexeur de terminaux asynchrones.
MS: Multiplexeur de terminaux synchrones.

IV - AUTRE POSSIBILITE:

- Bien que la méthode d'implantation avec les tâches imbriquées à plusieurs niveaux se prête à une intégration dans un système d'exploitation, celle-ci suppose néanmoins une bonne connaissance de ce dernier. Toutefois, si le calculateur, qui doit être connecté dispose,au moins d'une procédure de communication qui lui est propre, les problèmes de connexion peuvent être délimités de la façon suivante:

- Considérons un terminal en connexion avec un calculateur central sous un protocole du type FD-NTR. Nous avons admis,en particulier,qu'un processeur d'entrée/sortie peut être global,à plusieurs activités d'entrée/sortie. Et dans ce cas, nous avons envisagé comme processeur d'entrée/sortie une unité de disques. Mais ce peut être aussi un second processeur ligne. Dans ces conditions, la connexion de deux calculateurs, sous le protocole de type FD-NTR,peut se faire par l'intermédiaire du terminal. L'adaptation des procédures se fait alors au niveau de ce dernier. Par suite, le terminal sert d'adaptateur de transmission synchrone.