La communication dans un système réparti : construction du "service de communication" de Chorus

Alain Caristan

N° d'ordre :

THÈSE

présentée à

L' UNIVERSITÉ SCIENTIFIQUE ET MÉDICALE DE GRENOBLE

pour obtenir

LE DIPLÔME DE DOCTEUR DE 3ème CYCLE

par

Alain CARISTAN

Sujet de la thèse :

LA COMMUNICATION DANS UN SYSTÈME RÉPARTI, CONSTRUCTION DU "SERVICE DE COMMUNICATION" DE CHORUS

Soutenue le 24 mai 1984 devant la Commission composée de :

MM. **S. KRAKOWIAK** Président

M. GUILLEMONT
J. MOSSIERE Examinateurs
G. PUJOLLE

THESE

présentée à

L'UNIVERSITE SCIENTIFIQUE ET MEDICALE DE GRENOBLE

pour obtenir

Le GRADE DE DOCTEUR DE 3ème CYCLE

par

Alain CARISTAN

Sujet de la thèse:

LA COMMUNICATION DANS UN SYSTEME REPARTI, CONSTRUCTION DU "SERVICE DE COMMUNICATION" DE CHORUS

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES MATIERES

PREFACE ET REMERCIEMENTS

PREFACE

Un clin d'oeil en guise d'hommage, d'un modeste écrivain complètement inconnu (l'auteur), à un humoriste, chansonnier, écrivain et humaniste, qui avait une vision des hommes et de leur société si riche et prolifique qu'elle le nourri (au propre et au figuré) sa vie durant.

Pierre Dac avait un avis sur tous et sur toutes choses. Il n'a pas été difficile de découvrir dans son oeuvre une réflexion sur les affaires de science et les hommes de même nom.

"Qu'on le veuille ou non et qu'on s'en rende compte ou pas, la Science, en ses disciplines qui en font pourtant la force principale, au même titre que la Sobriété qui fait la force principale de la Sécurité et que la Contestation qui fait la force principale des Manifestations, la Science donc, laisse parfois planer quelque doute quant à l'exactitude de ses définitions et la justesse de ces précisions.

Ce principe, qu'on le veuille ou non et qu'on l'approuve ou pas, n'est pas plus contestable que révocable en doute, et réciproquement, car en ce qui concerne les sciences, à part les mathématiques plus ou moins élémentaires et plus ou moins spéciales, les autres, en leur ensemble plus ou moins composite, ont moins bonne réputation et marquent plutôt une nette tendance à l'approximation en ce qui concerne la relative exactitude ou inexactitude de leurs disciplines respectives, lesquelles, pour autant, n'en sont pas moins dignes de profond respect et de considération distinguée. Car, tant en effet qu'effectivement, nombre de disciplines scientifiques, pour ne pas dire toutes, ne

reposent que sur un postulat.

Or, en vérité scientifique, qu'est-ce qu'un postulat, sinon un principe premier indémontrable et conséquemment non démontré, mais dont, toutefois et cependant, et réciproquement, l'admission est nécessaire pour établir une démonstration suffisante en ses effets démonstratifs plus ou moins convaincants, poil à ..., en admettant, naturellement, que le caractère de la démonstration soit d'ordre suffisant pour que le nécessaire soit fait en ce qui concerne son admissibilité, mais à condition, cependant et toutefois, que l'admissibilité de l'ordre suffisant en l'absolu de son nécessaire en tant qu'élément de première nécessité soit en mesure d'établir son exact degré d'accessibilité.

Comment s'y prendre pour démontrer un principe indémontrable, donc ne pouvant être démontré par suite d'impossibilité de démonstration, tant théorématique que de marques d'affectation, poils aux ...?"

Après ces quelques lignes, suivant qu'il sera pessimiste ou optimiste, conforté dans son aversion de la science ou irréductiblement curieux, ennemi farouche ou ami indulgent, le lecteur poursuivra ce mémoire dans lequel il trouvera, je l'espère, le moins d'arguments possibles susceptibles d'étayer le propos précédant.

A ce propos, qu'il me soit permis de remercier ceux qui, faisant fi de ces remarques et avec beaucoup de conscience professionnelle, se sont donné la Peine de me lire et de m'assister pour la réalisation de ce mémoire.

Monsieur le Professeur Sacha KRAKOWIAK, qui m'a fait l'honneur d'accepter de présider le jury de thèse, ses conseils ont contribué à l'amélioration de la clarté de cet ouvrage,

Monsieur Marc Guillemont, qui a accepter de participé au jury et compagnon depuis les premiers jours dans l'aventure Chorus,

Monsieur Jacques MOSSIERE, qui a accepter de participer au jury,

Monsieur Guy PUJOLLE, qui a accepter de participer au jury.

Je dois ajouter que toute ma reconnaissance va aux membres de l'équipe Chorus, passés, présents et même futurs puisque grâce à eux nous conservons l'espoir de ne pas avoir travaillé en vain.
J'emettrais une attention particulière pour Monsieur Hubert Zimmermann, qui m'a encouragé dans cette étude des mécanismes de communication. Le temps passé à cet étude s'avère être, tout au moins sur le plan personnel, une époque particulièrement riche, du point de vue de ma connaissance de ce domaine pas toujours bien balisé.

Pour conclure cette préface, j'exprimerais un souhait que vous me ferez la grâce de croire exempt de toute fatuité; puisse la lecture de ce mémoire aider les gens qui s'y interresserons autant que sa rédaction a pu m'apporter d'enseignements.

C'est à P. DAC [redacted]quel reviendra le mot de la fin (que n'auraient sans doute pas renié les gens de Monsieur Jacques de CHABANNES, seigneur DE LA PALICE):

"Bref, en résumé scientifique et pour conclure scientifiquement, avant la science, ce n'est pas la science, après la science, ce n'est plus science la science, la science c'est la science. Poil aux objecteurs de conscience. C.Q.F.D."

CHAPITRE 1: LES SYSTEMES REPARTIS ET LES COMMUNICATIONS

INTRODUCTION

Les systèmes informatiques répartis ont hérité des études poussées sur les systèmes centralisés et les réseaux de transport de l'information. Au cours des années 70, on a assisté à une transformation, une adaptation, une synthèse des concepts propres aux réseaux et aux systèmes pour aboutir aux systèmes répartis.

La définition et la spécification d'un système réparti font appel à des mécanismes intégrant à la fois le traitement et le transport. Une conséquence directe en est que le traitement de l'information et son transport s'y trouvent liés d'une façon qui n'est pas toujours simple à maîtriser, tant du point de vue de la protection que de la fiabilité.
L'emprunt aux systèmes centralisés et aux réseaux est plus ou moins grand dans les différents systèmes réalisés de par le monde. Cependant de nombreux concepts ont été formalisés et servent de support aux études actuelles sur les systèmes répartis.

Le projet Chorus a été créé afin d'étudier les caractéristiques de tels systèmes et d'offrir les moyens de les réaliser. Ses travaux s'étayaient sur des recherches menées jusqu'alors sur les systèmes centralisés (projet ESOPE [FER 76]) et les réseaux (projet CYCLADES, [ZIM 77]. Les premiers résultats de ces travaux ont conduit à la définition de concepts systèmes permettant l'analyse, la description et la réalisation d'applications réparties.

L'architecture Chorus ([BAN 80],[ZIM 81])se caractérise par un type particulier d'organisation des éléments de traitement de l'information et par une définition stricte des méthodes de communication entre eux. La structuration des éléments de traitement et leur intégration dans le langage Pascal font l'objet d'une thèse [GUI 82]. On y trouve largement décrits les mécanismes de traitement et il n'est pas dans

notre propos de les développer plus avant. Parallèlement à cette organisation des traitements, il a fallu définir les mécanismes de communications qui devaient être mis à la disposition du concepteur d'une application. S'il existe un consensus sur les moyens de communication à développer (définition d'interface, de protocoles) dans un système réparti, l'accord est loin d'être fait sur la façon de les organiser et de les utiliser.

Nous n'avons pas la prétention de donner ici "le bon choix", mais de montrer que parmi les choix possibles, il est possible d'en accorder un certain nombre pour obtenir un système réparti dont la fonction de communication est compatible avec les objectifs d'une architecture distribuée.
Nous utiliserons le système CHORUS comme support de démonstration puisque, de part sa conception, il nous a permis de pousser l'étude du service de communication entre les entités de traitement.

1/ Qu'est-ce qu'un système réparti?

L'expression "système informatique réparti" est souvent employée sans qu'une définition unique en ait été fournie. Nous allons en donner les principales caractéristiques physiques et fonctionnelles que l'on s'accorde à leur reconnaître.

1.1/ Caractéristiques physiques

On peut tenter de classer les systèmes réparti par les raisons, universellement reconnues, qui ont poussé à les construire. On peut citer la miniaturisation des équipements, la baisse du coût des composants, l'amélioration des moyens de transmission de données qui ont permis d'apporter une plus-value et un plus grand confort d'utilisation de l'informatique grâce à la modularité, la

privatisation et la puissance des petits systèmes. Il faut y ajouter l'amélioration des techniques de transmission (commutation de circuits, de messages, de paquets) et plus généralement le développement des réseaux.

Deux démarches différentes ont abouti à deux grandes familles de systèmes répartis.

1.1.1/ Les systèmes faiblement couplés

Dans cette famille on classe les systèmes répartis que l'on peut décrire comme des "réseaux d'ordinateurs". Les relations entre les machines sont matérialisées par des lignes de transmission dont l'organisation peut se compliquer jusqu'à la constitution d'un "réseau de transport de messages" [CLIP 76, ELO 77, HEA 73].

1.1.2/ Les systèmes fortement couplés

Dans cette famille on classe les systèmes répartis qui employent un autre moyen de connexion que le "réseau" entre les machines. Un des buts avoués étant d'obtenir du parallélisme dans les traitements. Les systèmes dits multi-processeurs connectés par un bus spécialisé (non un réseau) représente la majeure partie de ces systèmes [ENS 77, SWA 77]. Les premières versions n'étaient pas admises en tant que système réparti parce qu'elles utilisaient exclusivement de la mémoire commune. Mais les générations de machines actuelles avec une mémoire privée pour chaque processeur peuvent être considérées comme des systèmes répartis (cela n'exclut pas l'existence d'une zone de mémoire commune pour le partage du code par exemple).

1.1.3/ Les systèmes hybrides

Cette famille est constituée des systèmes qui ont fait des emprunts aux deux groupes que nous venons de citer. Ce sont des systèmes réparti constitués de machines multi(ou mono)-processeurs à mémoire privée reliées entre elles par un "réseau de transport de messages". On trouve de tels système dans les applications des réseaux locaux [FAR 72]. Le réseau est très souvent un réseau local à haut débit [HOP 80, MET 75]. C'est le genre de système actuellement utilisé dans le projet Chorus.

1.2/ Caractéristiques fonctionnelles

Un système réparti est construit avec des machines dans lesquelles les informations sont mémorisées sous forme de blocs de longueur finie et toutes les structures de données y sont représentées et traitées sous cette forme. Il apparaît donc souhaitable que, lorsqu'on relie ces machines entre elles, le système de communication permette d'échanger de tels blocs d'information.

Dans la mesure où le système réparti utilise souvent un ou des réseaux pour relier ses machines, dans la mesure où ces machines ne partagent pas de mémoire commune, il est important d'utiliser des mécanismes de communication indépendants des configurations et des types de connexions.

D'autre part, les mécanismes de communication ne doivent pas être vus comme différents suivant que la communication est locale à une machine ou distante.

L'échange de messages (constitués de blocs de longueur finie) utilisé dans les réseaux à commutation de paquets, par exemple, amène une uniformisation des échanges de données. Ce mécanisme est donc

généralisé à tous les échanges dans les systèmes répartis actuels [BAL 76], [WAR 80].

Actuellement, on décrit un système réparti de la façon suivante:

- Un système réparti est composé d'un nombre arbitraire de systèmes informatiques et de processus utilisateurs.
- Son architecture est modulaire, ce qui lui offre la possibilité d'avoir un nombre variable d'éléments de traitement (processus et processeurs).
- Les communications se font par messages sur une structure de communication commune (exceptée la mémoire commune).
- Un certain contrôle sera fait à l'échelle du système réparti, afin de permettre la coopération dynamique de processus et la gestion des exécutions.
- Le délai de transit des messages est variable et non borné (problème du réseau). Il existe un temps non nul entre la production d'un évènement par un processus et la matérialisation de sa production chez le processus récepteur.

Remarque

Les caractéristiques du système introduisent quelques problèmes nouveaux d'un point de vue fonctionnel.

Le choix de la communication par message peut sembler une contrainte trop forte eu égard à la fiabilité et la lenteur des communications. L'impossibilité de connaître l'état global du système entraîne une incertitude sur les transferts de l'information, incertitude qu'il est difficile de lever en un point quelconque du système. Ainsi, il n'y a pas une certitude absolu qu'un évènement signalé par un message sera effectivement percu par le destinataire de ce message.

1.3/ Les fonctions dans le système réparti

Un système réparti possède une organisation propre à répondre à la fois aux nécessités d'une gestion de processus "centralisé" (sur un même processeur) et à celles d'une gestion de processus distribuée (sur plusieurs processeurs). Ces fonctions incombent au système d'exploitation réparti qui se charge d'offrir l'environnement nécessaire à l'exécution des processus et à leur communication [ENS 78].

1.3.1/ Le système d'exploitation réparti

Le système d'exploitation réparti doit répondre à deux préoccupations majeures:

1) ses fonctions doivent être accessibles de tous les points du système, ce qui conduit à les répartir plus ou moins, suivant leur importance vis-à-vis de la disponibilité et de l'accessibilité.
2) Il doit gérer l'utilisation des moyens de communications.

La solution adoptée dans la majeure partie des systèmes est la minimisation des fonctions reconnues comme vitales qui vont alors être regroupées dans un bloc appelé noyau (kernel, nucleus, etc.). Ce noyau est installé sur chaque machine ou chaque processeur et réalise partout des fonctions identiques. Le reste des fonctions du système est confié à des processus système qui sont identiques aux processus d'application [DON 79].

1.3.1.1/ Le noyau

Le noyau de système est souvent chargé:

- du cadencement des processus,
- de la commutation locale (sur une machine ou un processeur) des messages,
- du traitement des interruptions,
- de la réalisation des entrées/sorties,
- de la gestion du temps.

1.3.1.2/ Les processus système

Suivant les options des concepteurs, on trouve un certain nombre de systèmes dont les autres fonctions d'exploitation sont partagées entre le noyau et des processus système. On peut citer entre autres fonctions, la création et la destruction des processus, la création et la destruction des structures de communication (portes, liaisons etc...), la gestion des communications distantes, la gestion des fichiers.

1.3.2/ Le service de communication

Dans le système réparti, tous les échanges se font par message. Les communications deviennent alors un élément de base du système d'exploitation. Le service de communication doit assurer le cheminement du message dans le système depuis l'émetteur jusqu'au récepteur. On peut distinguer plusieurs tâches pour ce service:

1) Prise en charge du message de l'émetteur.
2) Détermination du site du récepteur.
3) Acheminement du message
4) Remise au récepteur.

Cette liste est exhaustive au sens des tâches indispensables pour la communication. Mais en fait, elle permet tout juste d'aborder les problèmes de la communication tels que la désignation des entités qui communiquent, le routage des messages, les caractéristiques des voies de communication empruntées. Il est donc nécessaire de réaliser beaucoup de fonctions pour effectuer correctement les opérations de communication.

2/ Définition du service de communication

Il n'existe pas une définition unique du service de communication d'un système réparti. Il s'agit en fait d'un certain nombre de fonctions qui sont regroupées dans la même activité. Nous avons donc pris en exemple quelques services de communication de systèmes répartis variés. Nous avons isolé et décrit dans chacun d'eux les principales caractéristiques de la communication.

Puis nous expliciterons ce qui nous semble essentiel pour définir un service de communication.

2.1/ Quelques services de communication

Nous avons trois exemples:

- un système de communication spécialisé dans le transport des messages : Transpac,
- un service de communication utilisant des liaisons entre les processus: le service de communication dans Trix,
- un service de communication sans liaison entre les processus: le service de communication dans Accent.

Nous devons, avant ces exemples, donner deux définitions importantes.

Les techniques de commutation de messages dans les réseaux ont amené le développement de deux types de service définis au C.C.I.T.T. (avis X2):

a) Le service "datagramme" correspondant à l'acheminement de blocs d'information indépendants les uns des autres, sans garantie de séquentialité et sans établissement au préalable d'une communication. Les fonctions de séquencement et de contrôle de flux, si elles sont nécessaires sont à la charge des abonnés au réseau [ANSI 79].

b) Le service "circuit virtuel" permettant, après établissement d'une voie logique et physique de communication entre deux abonnés du réseau, l'échange bidirectionnel simultané de séquences quelconques de blocs d'information, avec conservation de l'ordre d'émission; outre cette fonction de séquencement, le circuit virtuel inclut une fonction de contrôle de flux sur chaque sens de transmission, s'exercant de facon indépendante sur chaque circuit virtuel [CCI 80, MAC 77]. Il est possible d'offrir un service de circuit virtuel permanent ayant les mêmes fonctions de transmission, sans phase préalable d'établissement (relation fixe entre deux abonnés).

2.1.1/ Transpac

Transpac [DES 76] est le réseau public francais de transport de paquet.

Objectif

Le réseau Transpac a pour objectif de permettre l'interconnexion de terminaux et d'ordinateurs. C'est un réseau de transport de paquets qui permet le partage des ressources de transmission.

Organisation

Le réseau Transpac est un réseau maillé constitué de noeuds de commutation et de lignes rapides (72 kilobits/s). La technique de transmission est la commutation de paquets sur circuit virtuel.

Fonction et caractéristiques du S.C.

Le service de communication assure le transfert de paquet depuis un abonné émetteur jusqu'à un abonné récepteur. L'acheminement du paquet se fait sur circuit virtuel. Le service minimum est:

- l'acheminement correct si le destinataire est correct,
- le séquencement des paquets,
- le contrôle de flux et le contrôle d'erreur sur le circuit.

Désignation

Chaque abonné est désigné par un nom unique sur le réseau. Ce nom sert à identifier les partenaires à l'établisssement du circuit. Lorsque le circuit est établi les abonnés sont identifiés par circuit et les paquets sont adressés par circuit. C'est-à-dire que l'on a un adressage local dans chaque noeud de commutation et qui indique le chemin physique (ligne d'entrée et de sortie du circuit) que doit emprunter le message.

Routage

Le routage est adaptatif à l'établissement du circuit virtuel. Il est fixe lorsque le circuit est établi. Il y possibilité d'établir des circuits permanents entre deux abonnés.

Spécificité du service

C'est un service de transport de données par paquets. Le transport est séparé des traitements qu'il ne connaît pas.

Ce service impose la connection comme préambule à tout échange de message. En contre-partie il offre l'avantage de conserver l'ordre des paquets et de rendre fiable les transferts sur le circuit virtuel ouvert. Il ne permet pas de s'affranchir des pannes de lignes entraînant la rupture du circuit. Il est donc nécessaire de superviser l'utilisation du réseau à un niveau supérieur.

2.1.2/ Trix

Trix [WAR 80] est un système d'exploitation pour système physiquement réparti.

Organisation

Le système d'exploitation est constitué de noyaux et de processus système implantés sur les différents processeurs.

Fonctions et caractéristiques du S.C.

Il permet l'échange de messages entre un processus maître et un processus esclave sur des voies de communication appelées "streams". Le stream est une voie de transfert entre deux processus qui s'y sont connectés. Les transferts sont asynchrones et non bloquants.

A un instant donné sur le stream, un maître envoie les données et un esclave les recoit. Le stream est bidirectionnel et les rôles maître/esclave peuvent s'inverser. Les streams sont partageables entre plusieurs processus.

Protocole de communication

Le transfert sur le stream est en datagramme.

Désignation

Les streams sont répertoriés de façon hiérarchique. L'identification des streams se fait par une consultation de répertoires qui permettent de déterminer les associations (par concaténation) de streams qui donnent l'accès à un processus donné.

L'identification est indépendante de la localisation.

Routage

Il se fait par une succession de chaînage à des streams. Les répertoires sont accessibles par des streams ce qui permet de "booter" le mécanisme de communication. En effet, un processus est créé avec un premier stream vers un premier répertoire qui lui permettra d'acquérir d'autres streams.

Spécificités du service

Pour communiquer il faut acquérir un lien. Ce lien est accessible par un répertoire qui consistue la table de routage du service de communication. Le protocole de base est le datagramme. Le traitement et le transport peuvent être séparés car les transferts de données ne sont pas bloquants.

2.1.3/ Accent

Accent [RAS*81] est un système exploitation orienté vers la communication à travers un réseau.

Organisation

C'est un réseau d'ordinateurs Vax connectés par un réseau à diffusion Ethernet. Sur chaque machine sont implantés un noyau et des processus système.

Fonctions et caractéristiques du S.C.

Le service de communication commute les messages entre des portes. Les portes appartiennent au noyau et sont associées, sur demande, aux processus utilisateurs. La possession d'une porte peut être partagée entre plusieurs processus, mais un seul peut l'utiliser à un moment donné.

Le transport offre plusieurs types de services locaux pour tenir compte des tailles de files d'attente limitées:

- le processus est suspendu jusqu'à ce que le message soit placé sur la file d'attente de la porte réceptrice,
- le processus n'est pas bloqué et reçoit un message d'erreur si il y a un incident à la remise,

- Le processus n'est pas bloqué et reçoit un message d'acquittement lorsque son message est placé dans la file réceptrice. Dans cette situation, il n'est permis d'envoyer qu'un message à la fois.

Les messages sont soumis à des priorités sur les portes réceptrices. Il sont séquencés en fonction du processeur sur lequel ils ont été générés. Le service de communication surveille la durée de vie d'un paquet ce qui garantit un délai de transit maximum dans les communications.

Les communications distantes sont réalisées par des processus serveurs de réseau.

Désignation

Les portes identifient les services ou les structures de données accessibles chez un processus donné. Elles ont un nom local, le seul connu des utilisateurs, et des noms globaux (uniques dans le système réparti) connus des noyaux uniquement.

Routage

Le noyau local fait le routage des messages vers les portes locales en utilisant leur nom local.

Si la porte réceptrice est distante, le noyau local émetteur fait la conversion nom local/ nom global pour le routage entre les sites puis le noyau récepteur fait la conversion nom global/ nom local pour la remise du message sur la porte réceptrice.

Spécificités du service

Le service de communication de Accent prend bien en compte les problèmes de l'univers réparti sur un réseau (délai de transit, séquencement suivant la chronologie du site émetteur, etc.). Le service de base est un datagramme avec diagnostics.
Le transport et le traitement peuvent être assez liés dans le sens qu'un processus peut être bloqué dans l'attente d'une opération réussie de commutation. Mais il offre aussi la possibilité d'un asynchronisme total entre transport et traitement (diagnostic en différé). Le concept de porte constitue une définition nette, concise et modulaire de l'interface de communication entre traitement et transport.

2.1.4/ Conclusions

Les trois services de communications que nous venons de décrire mettent l'accent sur pliusieurs problèmes liés aux communications dans le système réparti (routage, désignation, partage de ressources de communication).

Chacune de ces implantations correspond au choix délibéré d'un modèle de communication et le découpage des fonctions et le niveau de visibilité offert aux processus utilisateurs est différnet suivant l'option choisie.

Nous voyons donc que le modèle Chorus devra être défini non seulement en fonction des communications que l'on voudra réalisé mais encore suivant le type de service rendu par le système aux utilisateurs.

2.2/ Le choix du modèle de communication de Chorus

Le modèle de communication de Chorus permet la définition et la structuration des communications dans l'architecture de système réparti Chorus. Il est important de choisir un modèle de communication sur lequel pourront s'appuyer les spécifications des interfaces de traitement et de transport, les choix des mécanismes de désignation et de routage. Pour cela il s'agit de bien définir les contraintes de notre système et les besoins réels de notre architecture.

2.2.1/ Contraintes générales

Le partage des ressources

Le partage des ressources de communication conduit à utiliser des techniques de transmissions qui assurent une utilisation équitable des supports de transmission. Cela se traduit par le découpage en paquets et la mise en file d'attente de ces paquets en attendant qu'ils soient commutés.

Le choix du message et l'utilisation des réseaux dans le système réparti conduisent à l'utilisation des mécanismes de commutation. La commutation de paquet est la technique la plus répandue dans les réseaux et elle est tout à fait réaliste à l'intérieur d'une machine puisqu'elle fait appel à la gestion de files d'attente.

La_désignation

La désignation permet d'associer une information de désignation et une entité désignée. Le système de communication utilise cette désignation pour effectuer l'accès à cette entité [SHO 78].

Les_interfaces_et_les_protocoles_de_haut_niveau

Pour utiliser les équipements, pour échanger les messages il est nécessaire de définir des interfaces et des protocoles qui améliorent l'accès et la compréhension des mécanismes et équipements de base [POU 79].

2.2.2/ Les_modèles_possibles

Le modèle de communication que nous voulons définir va nous permettre d'abord de mettre en relation les processus de traitement qui sont en principe assez éloignés de la structure matérielle du système.

Le modèle de communication va donc surtout consister en la définition d'interfaces et de protocoles qui vont régir non seulement le dialogue entre les processus mais également le dialogue d'accès au service de communication.

Personne ne nie l'importance des protocoles de communication et chacun s'accorde à reconnaître qu'il existe une hiérarchie entre ces protocoles [SUN 75]. Cependant, il y a plusieurs façons de les implanter et par conséquent la visibilité qu'en auront les utilisateurs peut être très variée.

On peut distinguer deux approches différentes du le modèle de communication mis en place sur les systèmes répartis.

- Une approche que nous dirons "intégrée",
- Une approche que nous dirons "hiérarchique".

2.2.3/ L'approche intégrée

Dans cette approche, le parti est pris d'inclure la plupart des primitives de communication dans le langage de programmation des processus. Dans ce cas, le jeu de primitives permet un contrôle étroit des communications par les traitements. Un exemple significatif de cette approche nous est fourni par la réalisation d'un service de communication basé sur les appels de procédures distantes (Remote Procédure Calls) [NEL 81]. Une application en est faite sur un réseau local Ethernet reliant des calculateurs Alto de Xerox [SPE 82].

Cette application a pour but d'offrir un mécanisme de communication utilisant des primitives, appelées références, accessibles par des "calls" depuis le programme utilisateur.
Les communications se font entre un maître, qui a l'initiative de l'échange, et un esclave distant qui reçoit le message.

A chaque référence est associée une opération distante (mise en file d'attente par exemple) qui est exécutée par l'esclave. Les paramètres d'appel sont passés dans le "call référence" à une procédure de référence. Cette procédure prépare et envoi les paramètres dans un message à destination de la procédure de référence distante. Le message est pris en charge et commuté par des processus de communication présents sur tous les sites. La procédure de référence distante reçoit le message, exécute la requête et émet un message de compte-rendu qui contient les paramètres en retour de la procédure distante. L'arrivée du compte rendu permet au maître de reprendre son éxécution là où elle s'était arrêtée.

Il est possible de définir de nombreuses références en fonction des opérations dont on veut disposer au niveau langage.

Commentaires

Cette approche permet de disposer de toute une panoplie de procédures qui rendent visible la gestion des protocoles au niveau programme. Elle augmente les performances des communications si le service de commutation est rapide.

Mais:

Tout programmeur n'est pas forcément à même de concevoir un protocole de communication, surtout s'il doit être imbriqué avec les traitements de l'application.

Le processus appelant est suspendu jusqu'au retour de la réponse. Un tel contrainte n'est admissible que si l'on dispose d'un mécanisme de comunication très fiable et d'un délai de transit sinon borné, du moins très acceptable.

Le système n'a pas de visibilité sur les communications. L'utilisateur supporte la programmation des protocoles complexes qui peuvent nécessiter beaucoup d'appels de références. Ce modèle ne permet pas d'expérimenter aisément différents protocoles car il est vraisemblable qu'il faille, dans certains cas, reprogrammer les processus qui communiquent.

Une telle approche donne cependant de bons résultats si l'on est dans un système homogène, qui dispose de moyens rapides et fiables de communication (simplification des interfaces et des protocoles).

2.2.4/ L'approche hiérarchique

Dans cette approche le service de communication est appréhendé comme un ensemble hiérarchisé de couches dont les fonctionnalités sont clairement définies et complémentaires les unes des autres. Cette approche des communications fait l'objet des études de normalisation de l'ISO et un modèle d'architecture de systèmes ouverts est élaboré [AFN 82, ISO 81]. On trouve en annexe une description de ce modèle dont nous ne rappelons ici que les principales caractéristiques et conséquences.

Les couches basses (physique, liaison de données, réseau) représentent chacune un niveau de service, désormais bien connu, que l'on peut aisément formaliser dans un système réparti.
Les couches hautes (application, présentation, session) sont moins faciles à formaliser mais elles peuvent, en revanche, être groupées dans une même couche sans que cela modifie fortement la nature des traitements, donc des processus de traitement.

Les deux groupes s'articulent autour de la couche transport qui constitue une véritable passerelle entre les moyens de traitement et les moyens de transport de l'information.

Cette couche transport rend transparentes, aux traitements, les différentes fonctions mises en oeuvre lors des communications et permet ainsi une séparation plus nette du transport et du traitement.

Commentaires

Cette approche permet une décomposition fine du système de communication, avec la possibilité de définir des protocoles et des interfaces standards dans le système réparti. On peut espérer obtenir une plus grande visibilité externe des communications entre processus. L'approche hiérarchique permet d'utiliser différentes politiques au niveau du service de communication sans changer l'interface avec les niveaux supérieurs, en l'occurence: les traitements.

Les inconvénients, en contrepartie, proviennent de la construction de ces couches, des interfaces et de leur gestion qui alourdissent et pénalisent les utilisateurs du service de communication qui n'ont pas toujours besoin de toutes les fonctions mises en oeuvre. La définition d'un service minimum est plus complexe que dans l'approche intégrée parce que celui-ci doit être plus général que dans l'approche intégrée.
Enfin, l'application des principes du modèle en couches amène une surcharge au niveau des structures de l'information transportée (en-tête des messages).

Cette approche semble cependant se justifier dans un système hétérogène.

2.2.5/ Conclusions

On remarque que les deux approches sont fondées. Le choix dépend en partie du domaine d'application visé.

Par conséquent, pour définir le service de communication, il est nécessaire de connaître sa cible matérielle et les applications que l'on voudra réaliser. Le choix du modèle sera influencé par ces

caractéristiques, car il pourra être, suivant le cas, suffisamment général ou très spécifique.

Ce modèle doit faciliter l'implantation d'interfaces et de protocoles adaptés aux applications de Chorus.

2.3/ Présentation générale de l'étude

L'architecture Chorus est le résultat d'une recherche sur les architectures de systèmes répartis. Il n'est pas fait d'hypothèses sur la machine ou le réseau, ce qui nous conduit à envisager la possibilité d'utiliser un système hétérogène et à faire une étude un peu générale sur les mécanismes qui servent de base à la communication.

Notre principal souci est de définir un service de communication possédant des mécanismes de base simples mais suffisamment puissants pour que des mécanismes plus sophistiqués puissent être construits en les utilisant.

2.3.1/ L'approche

Notre approche de la communication est fortement influencée par les réseaux. Notre service de communication sera conçu pour effacer les dissemblances entre les moyens de transmission sans masquer l'aspect "système d'exploitation orienté communication" au concepteur d'une application répartie.

La répartition est abordée par la définition d'un mécanisme de désignation unique qui permet aux processus de traitement de s'abstraire de la localisation géographique de leur correspondant et facilite les fonctions de routage dans le service de communication.

Les contraintes introduites par l'utilisation des transmissions plus ou moins fiables et rapides vont être abordées afin de définir les interfaces et les protocoles à mettre en oeuvre dans le service de communication. Notre projet ici n'est pas tant de trouver des protocoles particuliers que de montrer où et comment peuvent intervenir dans l'architecture du service de communication des protocoles existants.

2.3.2/ La démarche

La démarche est à deux niveaux:

- nous devons définir un modèle de communication qui soit compatible avec le modèle d'organisation des traitements,
- nous devons construire les outils, les interfaces compatibles avec les concepts du modèle de traitement et qui permettent d'expérimenter et implanter tous les protocoles que l'on jugera utile d'exploiter.

Nous ne nous sentons pas obligés d'inventer un modèle de communication "original". Notre but est de proposer une organisation du service de communication qui réponde, le mieux possible, au modèle de communication choisi.

Le chapitre 2 présente l'architecture Chorus et le modèle de communication qui nous semble le plus approprié à cette architecture.

Le chapitre 3 pose le problème des relations entre le transport et les traitements. Dans ce chapitre, nous faisons une étude qui permet de déterminer le minimum requis pour le service de transport.

Le chapitre 4 décrit les outils disponibles et la construction des mécanismes de base du service de communication. Il s'agit surtout de

donner les caractéristiques du noyau et du "service réseau" qui sont utilisés par le service de transport.

Le chapitre 5 décrit le service de transport et les facilités qu'il offre aux acteurs utilisateurs.

Le chapitre 6 donne quelques remarques sur l'analyse et l'expérimentation du service de communication.

Les savants du Massachusett's Institut for Advancement of Science and Technology ont réussi a provoquer artificiel-lement chez le crocodile familier (Crocodilus affectuosus) une mutation telle que ls dos de cet animal au plumage si prisé s'orne spontanément de poignées régulièrement espacées. Il ne reste plus qu'à débiter l'insoucieux reptile en morceaux de longueur adéquate pour obtenir d'élégants sacs à mains avec le minimum de main d'oeuvre et donc le maximum de profit.

CAVANNA

CHAPITRE 2: CHORUS ET SON MODELE DE COMMUNICATION

INTRODUCTION

Le projet CHORUS a pour but de définir une architecture d'exécution pour des applications réparties, de concevoir et de réaliser un système d'exploitation destiné à supporter des applications construites selon cette architecture [BAN 80], [ZIM 81]. L'architecture Chorus est bâtie sur la notion de communication par message, qui est comprise comme mécanisme de base de la synchronisation. Le système d'exploitation repose sur la notion de noyau qui intègre une partie de la fonction de communication.

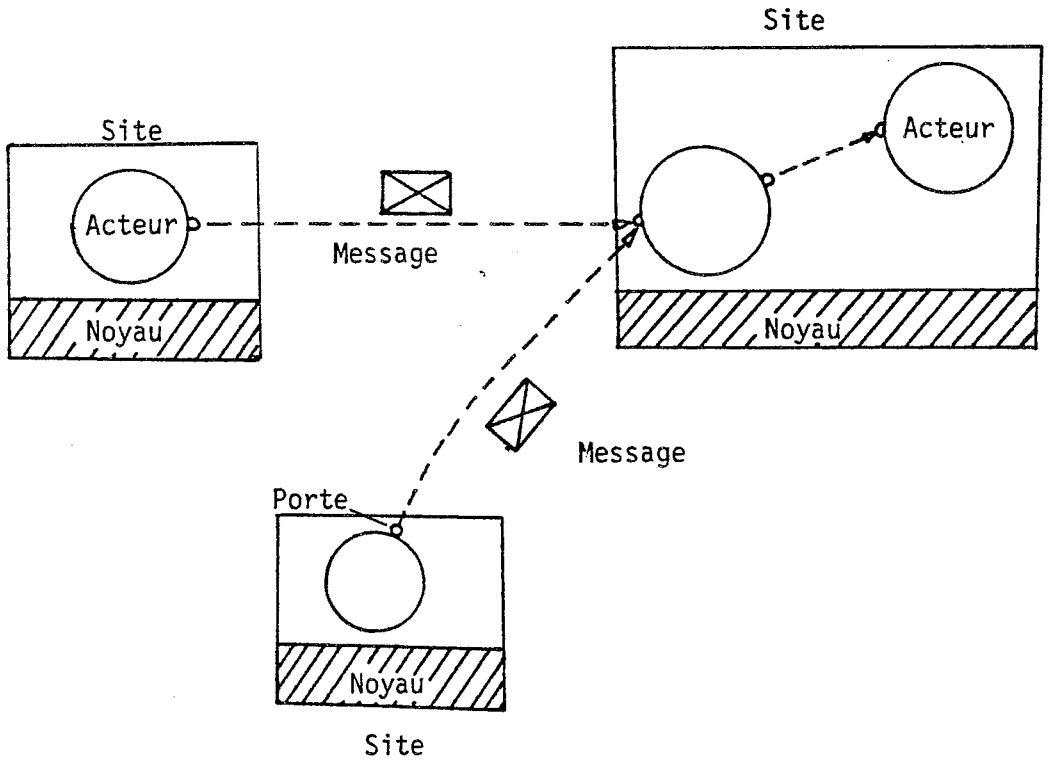
Aperçu général de l'architecture CHORUS

Dans CHORUS une application est conçue comme un ensemble d'acteurs coopérant en vue d'une tâche globale. Les acteurs sont des entités d'exécution locales (non réparties) et sont implantés sur des sites (machines) interconnectés. La notion d'acteur est presque équivalente au concept usuel de processus séquentiel dans la mesure où il ne s'exécute que sur un processeur à la fois.

Les acteurs se communiquent de l'information en échangeant des messages à travers des portes.

L'ordonnancement et l'exécution des acteurs sont supportés par le noyau CHORUS.

Le système d'exploitation réparti est constitué d'un noyau implanté sur chaque site et d'acteurs système [GAI 82].



Architecture Chorus

Dans ce chapitre, nous ne présentons que les principales caractéristiques de l'architecture Chorus afin de fixer des points de repaires pour le lecteur.

Nous analysons les conséquences des choix de Chorus dans l'organisation des traitements, ce qui nous permet de définir notre modèle de communication.

1/ Structure générale de Chorus

1.1/ Les éléments de Chorus

1.1.1/ L'acteur

L'acteur est l'entité active de traitement. Il représente l'unité d'exécution dans CHORUS.
Un acteur est local, c'est-à-dire qu'il se trouve sur un seul site à la fois.
Il est séquentiel, c'est-à-dire qu'il traite séquentiellement les messages qu'il reçoit.
Il ne traite qu'un message à la fois dans ce qui est appelé une étape de traitement.
Chaque message reçu déclenche une étape de traitement et seule la réception d'un message peut déclencher une étape de traitement.

Un acteur est reconnu du monde extérieur au moyen de portes sur lesquelles il reçoit les messages.

1.1.2/ La porte

Au sens du traitement, la porte est l'interface d'accès aux traitements réalisés dans l'acteur. Elle reçoit les messages destinés à l'acteur. La sélection des messages se fait sur les portes et l'aiguillage associe le nom de la porte à un point d'entrée d'une étape de traitement.

Au sens de la communication, elle représente la désignation unique de chaque entité du système que l'on veut identifier. La porte est l'interface universelle de communication. Les messages sont émis et reçus par des portes. Un message ne peut être acheminé qu'entre deux portes par le système de communication.

1.1.3/ Le noyau

C'est un élément de traitement dans la mesure où il est chargé de l'enchaînement des exécutions d'acteurs, de la prise en compte des interruptions, du traitement des entrées/sorties élémentaires. Dans Chorus, les interruptions physiques sont transformées par le noyau en un message qui est déposé sur la porte qui a été déclarée comme pouvant traiter l'interruption en question.

C'est un élément de la communication dans la mesure où il participe à la "commutation locale" des messages produits par les acteurs et les interruptions.

1.1.4/ Les messages

Dans Chorus, le message identifie le format sous lequel l'information est véhiculée d'un élément de traitement à un autre. Il contient l'information qui est échangée entre les acteurs ainsi que les informations nécessaires à la réalisation de la communication (adresses pour le routage et informations de contrôle pour les options de service choisies).

1.1.5/ Le réseau

Il s'agit du médium de communication entre les processeurs. Chorus ne fait pas d'hypothèse à priori sur le type de réseau à employer. Les caractéristiques de ce réseau ont cependant une incidence sur les performances et l'organisation interne du service de communication.

1.2/ La désignation

Le mécanisme de désignation permet une identification simple et globale des acteurs et des portes. Les noms globaux Chorus constituent un espace de noms partitionné dans lequel on peut distinguer deux ensembles de noms:

- les noms d'acteurs,
- les noms de portes.

Ces noms peuvent être répartis dans plusieurs classes (spécifiques, fonctionnels, génériques) mais nous ne nous intéressons ici qu'aux noms spécifiques.

Chacune des classes de noms compte deux catégories qui identifient distinctement:

- les entités sédentaires qui restent liées à leur site de création durant toute leur existence dans le système,
- les entités nomades qui peuvent migrer de site en site à travers le système.

Cette distinction est fondamentale pour le transport puisqu'elle offre beaucoup d'avantages lorsque l'on recherche la localisation d'une porte du système réparti. En effet, pour toutes les portes sédentaires le service de transport n'est pas obligé de faire de recherches. Il lui suffit d'interpréter son nom pour trouver le site de résidence d'une porte réceptrice (d'un message à transporter).

Dans Chorus tous les accès à des fonctions sont identifiés par des portes. Certaines de ces fonctions sont considérées comme des services système qui doivent être désignés de façon standard et publique dans le système. Toutes les portes de Chorus sont donc partagées en deux variétés de noms, liées au mode de désignation à leur création. Il existe deux modes de désignation à la création:

- le nom est choisi par le système (génération automatique de nom unique),
- le nom est imposé par le créateur de la porte.

Ces deux types de noms sont appelés les estampilles des noms globaux Chorus.

On voit l'intérêt d'une estampille imposée dans le cas de la communication entre des acteurs homologues. Un acteur du service de communication, qui a identifié la localisation d'une porte, peut construire le nom de la porte de son homologue de communication sur le site recherché. Il émet alors le message utilisateur dans un "message" adressé à la porte de transport de son homologue distant. L'estampille prédéfinie permet de se passer des tables de routage vers les homologues d'un service donné (le nom peut être entièrement calculé).

Ce mécanisme d'estampille n'est pas réservé aux seuls services système. On peut créer autant d'ensembles d'homologues qu'il y a d'applications dans l'univers Chorus.

Pour conclure nous donnons la structure des noms Chorus:
- nom du site de création (noté /S/),
- classe de l'entité désignée (spécifique/fonctionnelle/générique), (noté Sp/f/g),
- catégorie de l'entité désignée (sédentaire/nomade), (noté Sd/Nd),
- type de l'entité désignée (acteur/porte) (noté A/P/p),
- mode de création de l'estampille (choix système/choix imposé), (noté A/a P/p),
- estampille (système/service 1/service 2/applic. 1/applic. 2/etc..),

Exemple de nom de porte:

S/Sp/Sd/P/SERVICE : identifie la porte de nom Spécifique Imposé (P) SERVICE, Sédentaire sur le site S.

S/Sp/ND/p/BIDON : identifie la porte de nom Spécifique Bidon choisi par le système (p), sédentaire sur le site S.

1.3/ Organisation du traitement

1.3.1/ Exécution de l'acteur

Le traitement d'un message dans un acteur passe par trois phases essentielles à partir du moment où il est en attente dans la file d'une porte.

- la sélection du message qui permet de choisir la porte qui fournit le message à traiter. Les critères de sélection peuvent être fixés, modifiés dynamiquement par l'acteur.

- l'aiguillage du message qui permet de déterminer le traitement qui va être effectué. A cet effet, chaque étape de traitement est désignée dans l'acteur par un point d'entrée. Les paramètres d'aiguillage des messages vers les points d'entrée peuvent être fixés, modifiés dynamiquement par l'acteur.

- le traitement du message qui dépend évidemmment du programme spécifié par l'étape de traitement. Il peut à son tour provoquer la génération d'autres messages qui iront déclencher d'autres étapes de traitement dans d'autres acteurs ou dans le même.

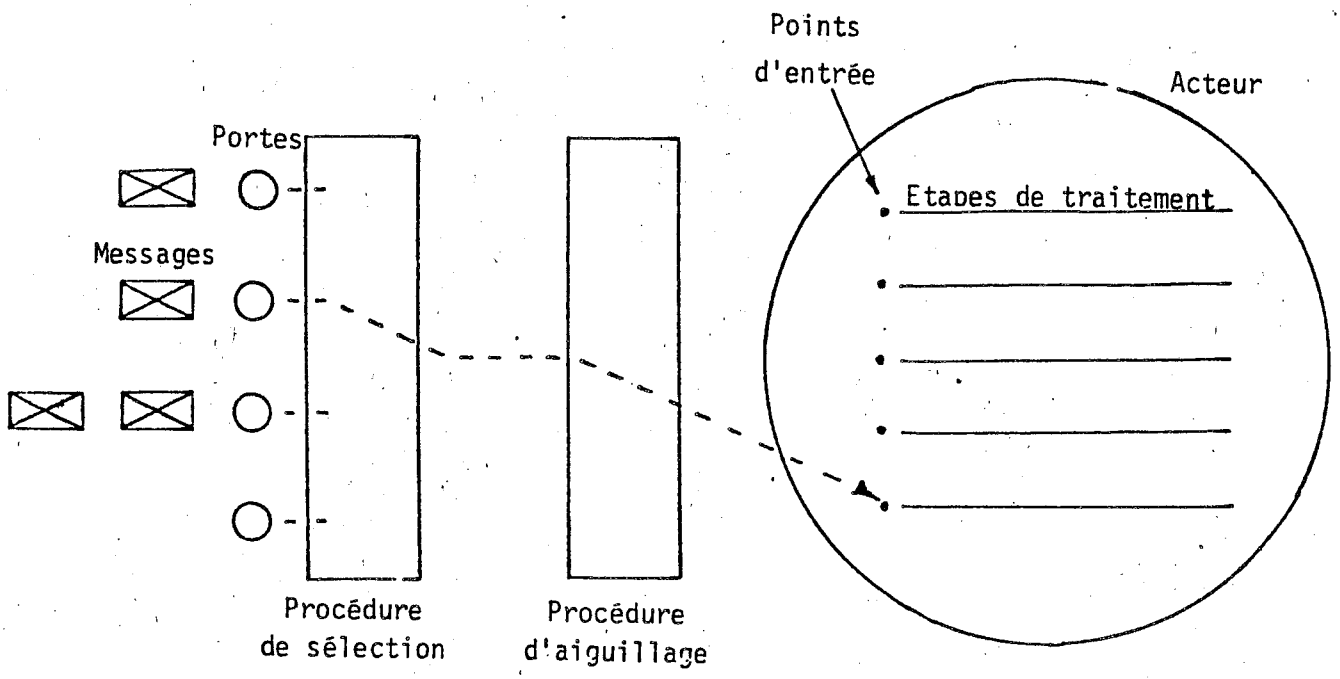


figure 1.1 : Exécution d'un acteur

L'acteur qui ne reçoit aucun message ou qui a traité tous ceux qu'il a reçus est en attente implicite. De ce fait, il peut se retrouver "bloqué" en attente de message (par exemple en cas de panne de son correspondant ou de perte de message durant le transport). Le système CHORUS lui offre la possibilité d'armer une temporisation sur une de ses portes. Les conséquences sont les suivantes:

- la temporisation est désarmée si le message attendu arrive avant l'échéance,
- un message de diagnostic est envoyé par le système à la porte soumise à la temporisation, si celle-ci est arrivée à son terme.

1.3.2/ Enchaînement des exécutions d'acteur

L'enchaînement des exécutions d'acteur est à la charge du noyau. L'activation d'un acteur n'intervient qu'à la fin d'une série de choix faits par le noyau. Un acteur ne sera lancé par le noyau que si:

1) il existe au moins un message sur l'une de ces portes,
2) cette porte est la plus prioritaire du site,
3) il n'y a pas de contre-indication (condition de sélection par exemple) à l'activation de l'acteur.

1.4/ Organisation d'une application répartie

Une application répartie dans Chorus est décrite comme un ensemble de traitements déclenchés par des messages. Il est possible de regrouper plusieurs traitements dans un même acteur.

Construire une application répartie signifie donc que l'on définit des traitements, qu'on les groupe dans des acteurs et que l'on définit des portes d'accès à ces traitements. En dehors de ceux qui sont inclus dans le noyau, tous les services offerts par le système sont réalisés par un ou plusieurs acteurs en coopération.

2/ Le modèle de communication de Chorus

Il convient d'abord de préciser les contraintes liées à l'architecture Chorus et leurs implications sur notre choix.

2.1/ Les contraintes de Chorus

Une étape de traitement est une opération atomique du point de vue de la communication, c'est-à-dire qu'il n'apparaît aucun message d'acteur pendant la durée du traitement.
L'acteur ne peut être activé que par un message. C'est-à-dire que son activation dépend principalement de la fiabilité du service de communication.
Les interruptions sont transformées en message, le service de communication doit donc être assez rapide pour transmettre un "message_événement".
Le message est un support pour véhiculer l'information mais il sert également à signaler un événement.

2.2/ Les implications

Le service de communication doit permettre le respect de l'atomicité de l'étape de traitement en n'offrant qu'un point de visibilité à l'acteur: la fin de l'étape de traitement. Cela n'est possible que si l'acteur est dans l'impossibilité matérielle de communiquer pendant l'étape de traitement.

Le modèle doit faciliter la séparation du traitement et du transport. L'acteur se soucie peu de la façon dont l'information est acheminée, si le transport est correctement fait ou rapidement fait suivant les cas.

Le modèle doit permettre la prise en compte rapide et le transfert des informations sur les phénomènes physiques locaux à une machine.

2.3/ Décision

Dans le chapitre d'introduction, nous avons indiqué que notre choix se ferait entre deux approches différentes: l'approche intégrée et l'approche en couches.

L'approche intégrée offre plus de souplesse à l'utilisateur sur le choix des protocoles qu'il utilise pour la communication ainsi qu'une amélioration des performances par simplification de la gestion des communications. Elle présente néanmoins le désavantage d'impliquer l'acteur de façon non négligeable dans les communications.
De plus, cette approche va à l'encontre du désir de séparer traitement et transport qui doit rester avec l'atomicité de l'étape de traitement l'un des points clé de l'architecture Chorus.

Notre choix se porte donc sur le modèle de l'approche hiérarchique. L'intérêt que nous y voyons est multiple:

- il est possible de séparer les traitements (dans les acteurs) et les communications,
- il est possible de conserver l'atomicité de l'étape de traitement en ne permettant l'activation du service de transport qu'à la fin de celle-ci,
- Il est possible d'adapter le service de communication aux contraintes des applications réalisées avec Chorus et aux contraintes des moyens de communication utilisés,
- Il est possible de simplifier la façon dont les acteurs appréhendent le service de communication.

Avertissement

Nous faisons le choix d'un modèle qui va nous permettre de spécifier le service de communication. Il n'est pas question d'appliquer à la lettre le modèle de l'ISO qui n'est pas un modèle d'implantation. Son intérêt pour notre recherche est de clarifier et de formaliser les concepts associés aux communications. Il nous fournit un langage et une syntaxe de base pour exprimer ce qu'est le service de communication du système réparti Chorus.

3/ Spécification du service de communication

3.1/ Chorus et le modèle ISO

Le choix du modèle ISO nous conduit à définir les entités qui participent au service de communication, les niveaux de l'architecture Chorus, et les points d'accès au service de communication.

3.1.1/ Les entités

Nous avons vu au chapitre 1 que le système d'exploitation est composé du noyau et d'acteurs système. Le service de communication est une fonction du système d'exploitation et ses fonctions sont partagées par le noyau et des acteurs du service de communication.

3.1.2/ Les points d'accès

Pour accéder à un service ou pour identifier la provenance d'une demande de service, les entités de deux niveaux adjacents utilisent la notion de point d'accès au service.

Dans Chorus tous les appels de service se font par des messages. Ce sont donc des portes qui identifient les points d'accès au service de

communicat ion.

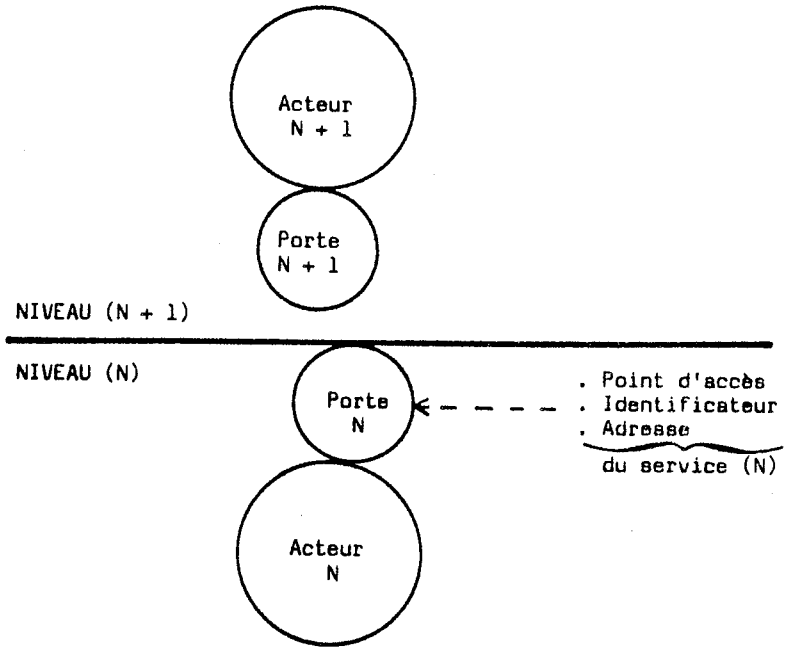


figure 3.1 : Point d'accès et identificateurs

3.1.3/ Les messages

Il est beaucoup question des messages lorsque l'on traite de la communication. Mais ce terme reste souvent ambigu et vague quant à savoir ce dont il s'agit en réalité.

La description qui en est faite dans Chorus est la suivante: lorsqu'un acteur (ou le noyau) veut envoyer un bloc d'information de traitement à une autre entité du système, il communique le contenu de ce bloc à la fonction de transport en même temps que des indications sur la destination et les options de service demandées (s'il y en a). Afin de faciliter leur transport, ces informations sont rassemblées par le service de communication en un message le temps de passer d'une porte à l'autre. Un bloc d'information de traitement peut être acheminé par le service de communication à l'intérieur de plusieurs messages successivement.

Pour reprendre le parallèle avec l'ISO, le message identifie une unité de données.

3.1.4/ Les interfaces

Le service de communication sert d'intermédiaire entre les acteurs qui l'utilisent et les moyens physiques de communication. L'interface est dans tous les cas constituée de portes mais la façon dont elles sont accédées est différente.

Il existe donc une interface de haut niveau qui permet l'échange des données entre les acteurs "utilisateurs" et le service de communication, et une interface de bas niveau entre le service de communication et le support physique (machine et réseau de communication).

3.1.4.1/ Interface de haut niveau

L'interface entre les acteurs utilisateurs et le service de communication reçoit les messages préparés par l'acteur durant l'étape de travail.

Interface acteur/service de communication

Cette interface est composée:

- d'une unité de données qui précise les conditions de transport de l'information utilisateur,
- d'une porte sur laquelle sont stockés, pendant l'étape de traitement, les messages à transporter,

Le service de communication est activé lorsque l'acteur signale la fin de son étape de traitement.

Interface_service/acteur

Cette interface est composée:

- de l'unité de données de l'acteur émetteur,
- de la porte d'acteur qui reçoit l'unité de données.

Lorsque l'acteur est activé, le service de communication lui fournit les informations de traitement qui ont été véhiculées et les paramètres de transport (le nom de la porte émettrice des données de traitement, etc..).

3.1.4.2/ Interface_de_bas_niveau

L'interface de bas niveau permet de prendre en compte les évènements physiques (entrées/sorties, interruptions) et de les transformer en message à destination de portes Chorus.

3.1.4.2.1/ La_gestion_des_interruptions

Une interruption est traitée par le noyau ou par un acteur. Le mode d'exécution de l'acteur impose que celui-ci ne traite que des messages et ne puisse être déclenché que par un message.

En conséquence, une interruption Chorus est un message envoyé d'une porte du noyau identifiant l'interruption et adressé à la porte d'acteur qui identifie le traitement associé à l'interruption. Le noyau effectue lui-même la transformation en un message Chorus de toutes les interruptions physiques qu'il ne traite pas. Ce message d'IT est émis d'une porte du noyau.

La gestion des interruptions est faite par un acteur système gestionnaire des interruptions. Elle consiste à superviser la distribution des traitements des interruptions entre les acteurs du

système qui le désirent. Un acteur qui veut traiter les messages d'une interruption itn s'adresse au gestionnaire d'interruptions. Lorsque le gestionnaire d'IT a accepté les requêtes de l'acteur pour traiter l'IT itn, il envoie un message au noyau pour lui donner le nom de la porte qui recevra les messages correspondant à l'IT itn.

Les données de l'acteur gestionnaire fournies au noyau sont:

- le numéro de l'IT,
- le nom de la porte d'acteur qui reçoit les ITs,
- le nom de la porte qui envoie le message d'IT.

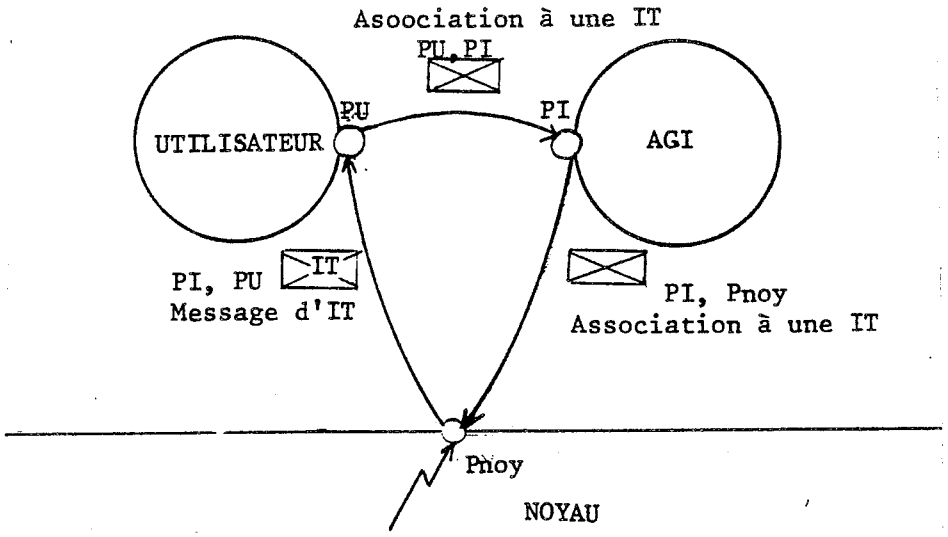


figure 3.2 : Gestion des interruptions

3.1.4.2.2/ La gestion des entrées/sorties

Le principe utilisé pour la gestion des interruptions est appliqué également à la gestion des entrées/sorties; une entrée/sortie est vue comme un échange de messages entre une porte d'acteur et une porte d'entrée/sortie. A chaque type d'entrée/sortie est associée une interruption Chorus. L'acteur qui veut réaliser l'entrée/sortie initialise une zone de données qui sera lue ou écrite par la procédure de transmission (du noyau ou du périphérique).

Lorsque l'entrée/sortie est terminée la procédure de transmission en signale la fin par une interruption destinée au noyau. Ce dernier génère un message d'interruption de fin d'entrée/sortie à destination de l'acteur concerné.

3.1.5/ Les niveaux de Chorus

Le choix de l'architecture en couches nous amène à décrire le service de communication comme un ensemble de fonctions regroupées sur différents niveaux.

Dans l'architecture Chorus, on peut distinguer d'emblée deux groupes qui correspondent pour l'un aux acteurs utilisateurs du système d'exploitation réparti, pour l'autre au support physique du système réparti (machine et réseau). Le service de communication sert de lien entre ces deux niveaux.

Lorsqu'un utilisateur veut transmettre un bloc de données il demande l'envoi de ce bloc au service de communication qui doit en réaliser le transport (au sens de l'ISO) dans une unité de données de transport.
La fonction de transport peut être plus ou moins complexe suivant la localisation du récepteur: nous l'isolons dans un niveau transport.

L'acheminement des unités de données de transport passe par une fonction de commutation à travers différents moyens de transmission qui doivent rester transparents aux entités du niveau transport; nous isolons ainsi un niveau réseau.

Le service de communication possède un certain nombre de portes entre lesquelles sont échangées les unités de données transportant les informations de contrôle des communications ou les données des utilisateurs. Ces échanges correspondent à des transferts physiques

entre les portes du service de communication. Nous isolons donc un niveau physique.

Nous arrivons ainsi à une première décomposition en niveaux de l'architecture Chorus dans laquelle on peut distinguer:

- le niveau physique, qui est constitué des moyens de transfert physique de l'information,
- le niveau réseau, qui est constitué des moyens logiques de commutation locale et à distance (noyau, réseau, acteur réseau)
- le niveau transport, qui est constitué des moyens logiques d'accès au service de communication,
- le niveau utilisateur, qui est constitué de tous les acteurs qui ne participent pas au service de communication.

Le schéma suivant résume la représentation de Chorus en niveau.

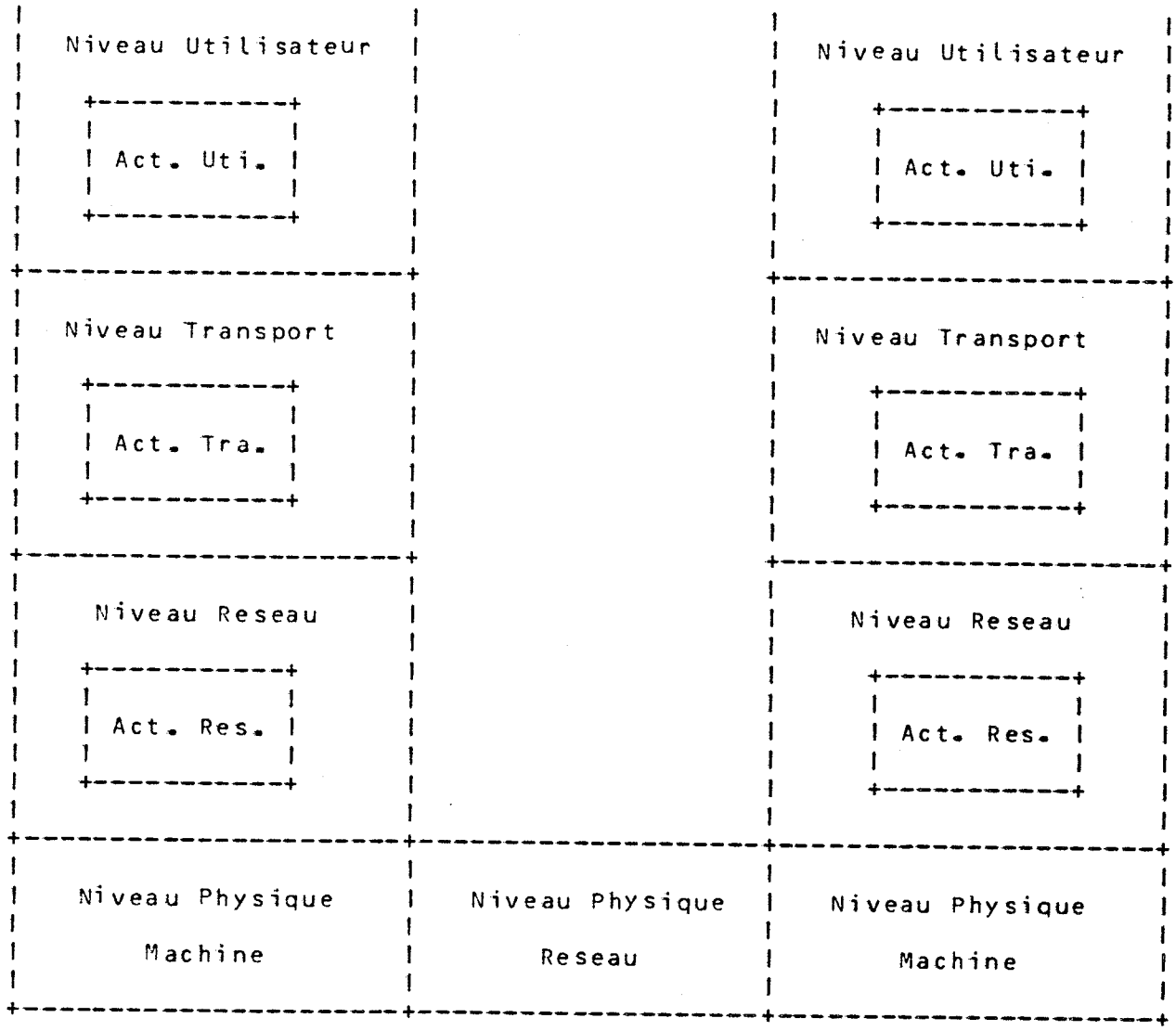


figure 3.3 : Les niveaux de Chorus

3.2/ Discussion

Pour réaliser le service de communication, nous allons utiliser des acteurs et le noyau de Chorus. La construction du service doit donc se plier aux contraintes définies pour les traitements.

3.2.1/ Les implications du modèle

Nos hypothèses de niveau impliquent que deux entités de deux couches adjacentes soient incluses dans deux acteurs distincts. Les conséquences sont nombreuses et peuvent paraître trop restrictives a priori. En effet, cela implique:

- que toutes les communications entre les couches sont des échanges de messages,
- que les points d'accès au service sont des portes,
- que les interfaces sont un ensemble de portes.

Ces hypothèses sont aussi fortes à dessein. Elles permettent de donner au concepteur d'une application répartie des mécanismes de communication stricts, ce qui facilite l'analyse et l'organisation des coopérations.

3.2.2/ Organisation du service de communication

3.2.2.1/ Représentation géométrique

On peut imaginer une représentation spaciale de Chorus, on remarque alors que le système peut se décrire sur un seul plan; les portes des acteurs et du noyau. Toutes les actions visibles passent, en effet, par celles-ci. l'information passe d'une porte à l'autre par un mécanisme de "transfert" des données. Una représentation géométrique de Chorus qui vient immédiatement à l'esprit est la suivante: un plan de transfert de données dans lequel se trouvent les portes de Chorus.

Du point de vue de la communication, les "transferts" entre les portes sont de deux types suivant qu'il s'agisse des transferts (considérés comme des recopies physiques de zones de données) entre

les portes du service de communication ou des transferts (avec création des messages) entre les portes des acteurs utilisateurs.

On peut alors considérer qu'il existe deux espaces de portes; l'un étant l'espace_des_transferts_physiques qui supporte les portes du "service de communication", l'autre l'espace_du_transport supportant les portes d'utilisateurs. L'intersection de ces deux espaces est matérialisé par les portes de "transport" du service de communication.

L'espace des portes "utilisateurs" est défini avec la restriction suivante: pour passer d'une porte à une autre de l'espace de transport, il faut passer par une porte de transport.

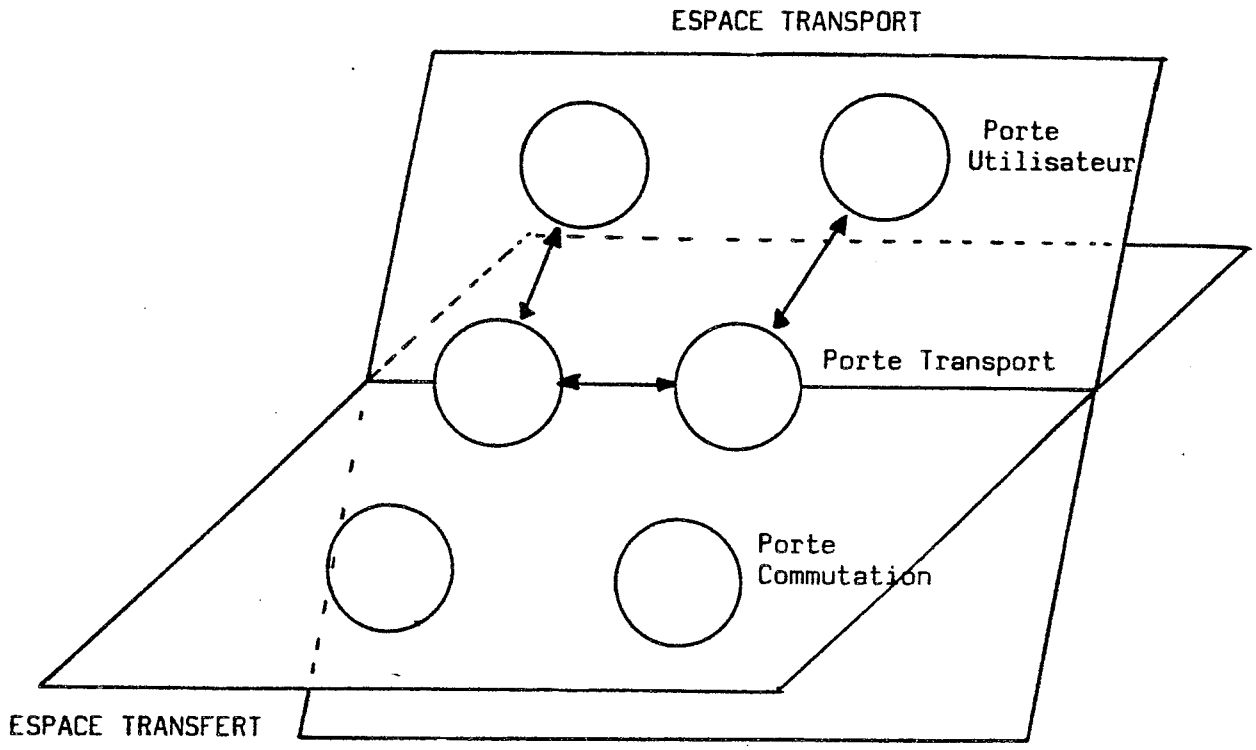


figure 3.4 : Représentation de l'univers Chorus

3.2.2.2/ L'implantation

La contrainte (introduite par le modèle) des portes d'accès à un niveau donné n'est pas imcompatible avec une implantation on l'on ne matérialise pas systématiquent les portes. Le modèle est d'abord créé pour spécifier le système que l'on veut construire et définir les portes. A l'implantation, ces dernières peuvent être remplacées (si les niveaux sont regroupés dans un même acteur) par des appels successifs à des procédures privées. Les transferts se font, par exemple, en utilisant une mémoire commune ou un passage de paramètres. Il n'est pas nécessaire de modifier la structure des informations échangées.

4/ Conclusions

Le service de communication de Chorus est vu comme un ensemble de fonctions accessibles aux acteurs et au noyau pour leur permettre de s'échanger de l'information.

Le modèle de communication choisi implique une séparation de ces fonctions en niveaux disjoints. Chaque niveau est accessible par des portes identifiant un ou plusieurs de ses services spécifiques et l'information s'échange dans des unités de données appelés messages.

Dans cet empilement, le niveau transport apparaît comme une charnière entre les traitements et la transmission. Avant de le décrire, nous allons donner dans le chapitre suivant les principes qui ont prévalus pour la conception de la fonction de transport.

Aussitôt qu'il eut inventé la Roue, l'homme l'utilisa.
Il eut tort.
Il aurait mieux fait d'attendre d'avoir inventé le Frein.

CAVANNA
le saviez-vous ? (2 ème fournée)

CHAPITRE 3: RELATION ENTRE TRAITEMENT ET TRANSPORT

INTRODUCTION

Un système dans lequel tous les échanges se font par des messages pose le problème de la confiance que l'on peut accorder au système de communication pour réaliser des traitements s'appuyant sur des entités distribuées. Cette confiance dépend principalement de la qualité du service de communication.
Mais que signifie la "qualité du service de communication" pour un de ses utilisateurs? La réponse dépend en partie des caractéristiques des applications qui utilisent ce service [SPR 78].

L'architecture Chorus se veut universelle. On peut dire, cependant, que les applications visées, en priorité, sont celles qui utilisent largement la répartition, ce qui impose des contraintes de fiabilité et de disponibilité des communications.

En raison même du principe de fonctionnement des acteurs Chorus, nous pensons que la fiabilité est la première qualité demandée au service de communication [SUN 76]. Cette affirmation est basée sur le fait que toute la synchronisation repose sur les messages, il faut donc pouvoir les acheminer de façon sûre (fiabilité) et si possible à tout moment (diponibilité).

L'évaluation de la fiabilité n'a de sens que si l'acteur est capable de détecter un incident au bout d'un temps fini, voilà pourquoi il faut ajouter la nécessité de définir une borne pour le délai de transit d'un message dans le service de communication.

Dans ce chapitre, nous essayons donc de répondre à la question: quel mode de transport peut-on choisir comme service de base pour la communication entre les acteurs utilisateurs.

Nous commencons par une typologie des applications visées qui nous permet de dégager un protocole courant d'échange. Puis nous

analysons ce protocole pour différents types de service de transport simples. Nous donnons enfin les éléments statistiques qui nous permettent de fixer notre choix.

1/ Typologie des applications visées

Le système d'exploitation Chorus est réalisé sous la forme d'une application répartie. Les contraintes rencontrées à cette occasion nous semblent représentatives de celles qui seraient imposées à une application quelconque qui utiliserait le service de communication de Chorus.

Nous prenons l'exemple d'une fonction du système d'exploitation Chorus: la création d'un acteur.

1.1/ Exemple d'application répartie Chorus

La création d'un acteur est une application répartie dans Chorus car elle est le résultat de la coopération de plusieurs acteurs qui peuvent se trouver sur des sites différents.

1.1.1/ Les acteurs systèmes

Les acteurs système réalisent tous les services qui ne sont pas inclus dans le noyau:

- création et destruction des acteurs (AGA),
- création et destruction, ouverture et fermeture des portes (AGP),
- gestion de la mémoire (AGM), gestion des fichiers (AGF),
- gestion des noms (AGN), gestion des interruptions (AGN)

Nous n'avons cité que les acteurs qui interviennent dans la création d'un acteur.

1.1.2/ Création d'un acteur

La création d'un acteur demande de nombreuses communications entre les différents acteur qui participe à ce service. Nous prendrons les conventions suivantes pour la notation des échanges:

AG* --> Message pour faire action X --> AG*

AG* représente l'acteur qui émet et recoit le message de demande de service dont le sens est donné entre les deux --->.

* est remplacée par l'indicatif de l'acteur

La flèche indique le sens du dialogue.

Nous supposons que la création d'un acteur B est demandée par un acteur A.

L'acteur A envoie à l'acteur de gestion des acteurs AGA un message contenant tous les renseignements sur l'acteur B:

Pa -->

Modèle d'acteur (dont est issu B)

Taille mémoire pour l'acteur

------ Info porte ombilicale ----------

Modèle de porte ombilicale (première porte de l'acteur)

Taille mémoire pour la porte ombilicale

Longueur de la file d'attente

Niveau de priorité de la porte ombilicale

Mot de passe

MESSAGE INITIAL (message de lancement de B)

--> AGA

1) L'AGA recoit le message de demande de création; il envoie à l'acteur "Gestion mémoire" (AGM) un message de demande de réservation d'une zone mémoire pour B.

AGA --> Demande mémoire de taille "Taille" --> AGM

2) AGM détermine l'emplacement mémoire qui va être alloué à B. On suppose que l'opération est possible. AGM répond à AGA en lui fournissant l'adresse de la zone allouée.

AGA <-- Adresse ZONEx allouée <-- AGM

3) AGA demande à l'acteur "Chargeur de Programmes" (ACP) de charger à l'adresse ZONEx le modèle d'acteur de B.

AGA --> Modèle d'acteur, adresse ZONEx --> ACP

4) Lorsque le chargement est terminé ACP envoie le contexte de B à AGA.

AGA <-- Contexte de B <-- ACP

5) AGA demande à l'acteur "Gestion des Noms" (AGN) un nom pour l'acteur B.

AGA --> Un nom pour B SVP ! --> AGN

6) AGN attribue un nom global unique a B et indique ce nom à AGA.

AGA <-- Nom de l'acteur B = XXXX <-- AGN

7) AGA demande au noyau de contrôler et prendre en compte, si les contrôles sont positifs, la création du nouvel acteur.

AGA --> Contrôler et créer B de la part de A --> Noyau

8) Le noyau prend en compte l'existence du nouvel acteur et donne un compte-rendu positif à l'AGA.

AGA <-- Ok Création <-- Noyau

9) L'AGA demande à l'acteur "Gestionnaire des portes" (AGP) de crêer et d'ouvrir la porte ombilicale de B.

AGA --> porte ombilicale pour B --> AGP

10) AGP crée la porte la porte ombilicale Pb et la signale à l'AGA.

AGA <-- Porte ombilicale de B = Pb <-- AGP

11) L'AGA enregistre l'acteur B dans sa table et envoie le message initial de B sur la porte ombilicale Pb. Il envoie également un message de compte-rendu de fin de création à l'acteur A.

AGA --> Message initial --> Pb

AGA --> Acteur B appelé XXXX crée --> Pa

L'analyse de cette activité de création d'acteur nous permet de distinguer trois types de dialogues entre les acteurs système:

- l'appel avec réponse qui correspond, par exemple, à l'envoi d'une demande de service avec message de compte-rendu en retour (demande de place mémoire de la part de l'acteur gestionnaire d'acteur à l'acteur gestionnaire de la mémoire, suivi d'un message de compte-rendu indiquant que la mémoire a été allouée),

- l'appel sans réponse qui correspond à l'absence de message en retour pour l'expéditeur d'un message qui vient d'être traité (message de compte-rendu d'allocation de la mémoire par exemple),

- le transfert qui correspond au cas où l'échange de message est unilatéral et caractérisé par un débit important (le chargement de la copie du modèle d'acteur peut être faite en collaboration avec l'acteur de gestion de fichier qui transmet le binaire par message à l'acteur chargeur).

Remarque

Les deux premiers types d'échanges sont les plus courants. Les transferts rentrent dans un domaine particulier que nous considérons comme plus exceptionnnel. Nous ne faisons cette distinction que dans le souci d'orienter notre choix de transport d'abord vers des mécanismes de base les plus simples possibles et répondant aux demandes les plus courantes.

1.2/ Le protocole Client-serveur

La typologie que nous venons de faire nous permet de définir deux comportement d'acteurs qui sont caractéristiques de deux situations différentes: le client et le serveur.

1.2.1/ Définitions

Le Client effectue des demandes de services en envoyant des messages du même nom sur une porte du serveur.

Le Serveur traite les demandes reçues (ou les fait sous-traiter). A la fin du traitement il envoie une réponse qui est le message de compte-rendu de l'exécution de la demande.
Les règles du dialogue entre le client et le serveur forment le protocole Client-Serveur.

1.2.2/ Caractéristiques

Le protocole Client-Serveur peut être assimilé à l'exécution d'une transaction entre le client et le serveur. Le message de demande marque le début de la transaction et le message de réponse marque la fin de la transaction.

Nous nous proposons d'étudier ce protocole pour différents services de communication. Les paragraphes qui suivent ne donnent que des indications générales sur le comportement des serveurs et des clients.

1.2.2.1/ Le comportement du client

Il suit un dialogue du type appel-réponse. C'est-à-dire qu'il envoie un message [demande de service] et qu'au bout d'un temps fini il recoit:

- soit un message de réponse,
- soit un message de diagnostic du service de communication (si celui-ci en donne),
- soit un message de temporisation échue (s'il a fait la demande d'un contrôle de temporisation échue).

1.2.2.2/ Le comportement du serveur

Il suit un dialogue du type appel sans réponse.

1.2.3/ Relation entre protocole C-S et transport

Le protocole C-S repose uniquement sur les messages. La qualité du service de transport a donc une incidence directe sur lui.
En effet, le service de transport peut être réalisé de manière à offrir des informations plus ou moins précises sur son fonctionnement. De ce fait, les acteurs peuvent adopter des comportements différents pour chaque type de service susceptible d'être fourni par le niveau transport, ce qui va donc donner plusieurs solutions du protocole C-S.

2/ Quel transport pour Chorus?

2.1/ Les choix possibles

Le modèle ISO prévoit deux types de service de transport: Le transport en mode non connecté et le transport en mode connecté.

2.1.1/ Le mode non connecté

Le mode non connecté correspond à un mode où il n'y a pas de connexion entre portes utilisatrices. Dans le mode sans connexion le service de transport assure le transfert d'unités de données utilisateur indépendantes d'une porte émettrice vers une ou plusieurs portes réceptrices (diffusion).

Ce mode ne requiert pas un accord préalable entre les correspondants ni même leur présence simultanée.

2.1.2/ Le mode connecté

Dans le mode connecté, le service de transport permet l'établissement d'une connexion entre deux portes d'acteurs utilisateurs et le transfert sûr (contrôle d'erreur) d'une suite de messages dans chaque sens sur cette connexion, avec asservissement de chacun des émetteurs par le récepteur correspondant (contrôle de flux).

La connexion peut être rompue à l'initiative de l'un ou l'autre des correspondants, ou en cas de panne du service de transport.

Ce mode requiert l'accord préalable des correspondants et leur présence simultanée.

Lorsqu'un message a été accepté par le service de transport, il sera (sauf panne) remis au destinataire lorsque celui-ci sera en mesure de l'accepter. Le défaut d'un des correspondants (désactivation de la porte, panne, etc.) cause la rupture de la connexion.

Dans le mode connecté, le service de transport implicite est toujours l'acheminement des messages, mais il propose en plus deux facilités:

- le contrôle d'erreur,
- le contrôle de flux.

2.2/ L'approche de base

Le mode connecté est le plus utilisé dans les applications actuelles de Chorus. Il semble le plus compatible avec des échanges ponctuels qui ne nécessitent pas d'accord préalable entre les partenaires. Nous proposons d'étudier un type de transport correspondant à un mode sans connexion avec diagnostic éventuel. Les options de transport possibles sont les suivantes:

- ne pas délivrer de diagnostic,
- délivrer un diagnostic négatif, c'est-à-dire que le service de

transport donne à l'émetteur un message de compte-rendu s'il y a eu un incident de transport ayant entraîné la perte du message,

- délivrer un diagnostic positif, c'est-à-dire que le service de transport donne à l'émetteur un message de compte-rendu si le transport s'est bien déroulé.
- délivrer un diagnostic positif ou négatif, c'est-à-dire que le service de transport donne un compte-rendu dans tous les cas (transport bien ou mal réalisé).

Le dernier type de service n'est pas très intéressant dans le mode connecté. Il demande la génération d'un trop grand nombre de messages ce qui réduit sensiblement la performance du transport.

La question à laquelle nous allons répondre dans ce chapitre est donc la suivante: si l'on utilise un service de transport en mode non connecté avec diagnostics, ceux-ci permettent-ils de mettre en place des mécanismes fiables (statistiquement) de coopération entre les acteurs?

Remarques importantes

Dans cette étude nous admettons que le délai de transit est borné. C'est-à-dire que le service de transport élimine les messages dont la durée de vie est supérieure à une certaine valeur en temps. Ce qui permet aux acteurs de régler leur attente sur temporisation.

Nous nous intéressons principalement aux incidents de transport et non de traitement. En conséquence, nous considérons que le client ou le serveur fonctionnent parfaitement, c'est-à-dire sans panne.

2.3/ Etude du protocole C-S

L'étude du protocole C-S va consister à décrire et analyser les solutions réalistes dans chacun des cas suivants:

- transport sans diagnostic,
- transport avec diagnostic négatif,
- transport avec diagnostic positif.

L'analyse se fera comme suit:

- énumération des erreurs possibles,
- les reprises du client,
- les reprises du serveur,
- discussion des conséquences et description du protocole C-S correspondant.

2.3.1/ Transport sans diagnostic

Les erreurs possibles

La perte de la demande ou la perte de la réponse sont les deux incidents de transport qui peuvent perturber le protocole C-S.

Les reprises du client

Lorsque le client a émis sa demande, il est en attente de la réponse ou du message de temporisation échue.

Si la temporisation arrive à échéance, le client doit décider d'une reprise. Mais il est dans l'incertitude la plus totale en ce qui concerne l'état du serveur ou des communications. Une alternative s'offre à lui; reémettre la demande ou non.

Cas 1)

Il ne reémet pas la demande.

- si la demande est perdue, le service ne sera jamais réalisé.
- si la réponse du serveur est perdue, le client n'en est pas avisé. Mais le service a cependant été réalisé. L'importance des conséquences dépend de l'application.

Cas 2)

Il reémet la demande.

- si la demande est perdue, le serveur va voir la seconde demande comme si c'était le première et exécuter le service.
- Si la première demande était arrivée, le serveur va recevoir une demande en double (doublon de demande).

Les reprises du serveur

Il ignore le client tant qu'il n'a pas recu de demande émanant de celui-ci.

Dans le cas d'un doublon d'une demande, il ne doit pas exécuter le service mais il doit impérativement renvoyer la réponse.

Discussion

Les choix offerts par l'alternative du client peuvent se justifier tous les deux, ce qui donne deux versions du protocole C-S.

Dans le premier cas, il y a un risque d'abandonner le protocole avec un service exécuté dont le client n'a pas connaissance.

Dans le second cas, le serveur peut être inondé de demandes, mais il est raisonnable de penser que les incidents répétitifs sur un même protocole sont assez rares. On a donc l'assurance, si le client reémet la demande jusqu'à la réception d'une réponse, de terminer correctement le protocole C-S. Nous retenons donc cette option.

Le protocole C-S avec transport sans diagnostic est donc le suivant:
Pour le client,

1) - Emettre la demande,

2) - Attendre la réponse,

3) - Si temporisation, alors revenir en 1.

Pour le serveur,

1) - Attendre une demande,

2) - SI pas Doublon ALORS exécuter service
SINON détruire demande

3) - envoyer réponse, revenir en 1.

Nous avons représenté ces deux comportements client et serveur sur des diagrammes dont nous donnons le formalisme utilisé:

Les diagrammes représentent les étapes de transition des entités suivant le déroulement du protocole. Les conditions de déclenchement d'une transition sont; soit des demandes internes à l'entité (marquées ===>); soit des arrivées de messages (marquées par un rectangle contenant le type du message et une flèche pointant sur la transition).

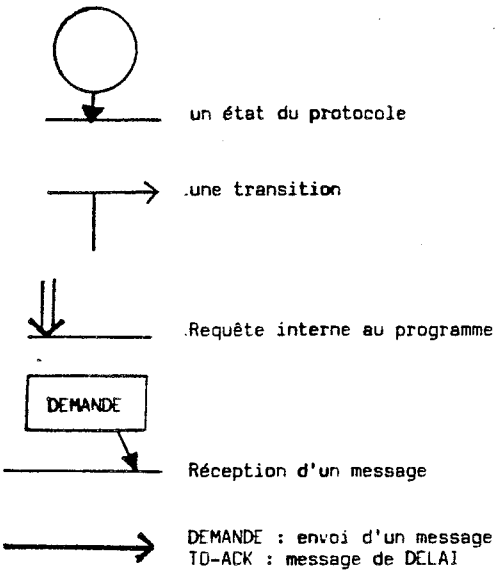


figure 2.1 : Conventions des diagrammes des protocoles

L'état 0 représente l'état de repos.

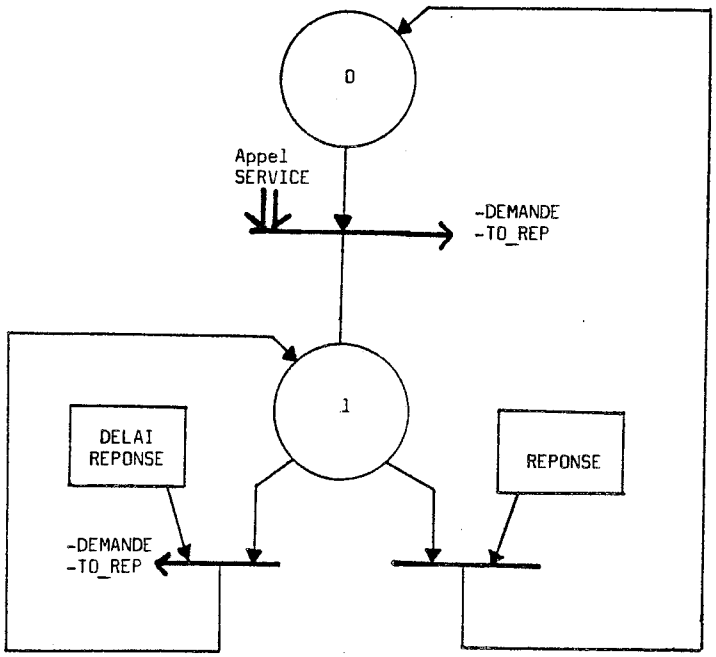


figure 2.2 : Protocole Client: transport sans diagnostic

1 = En attente de réponse

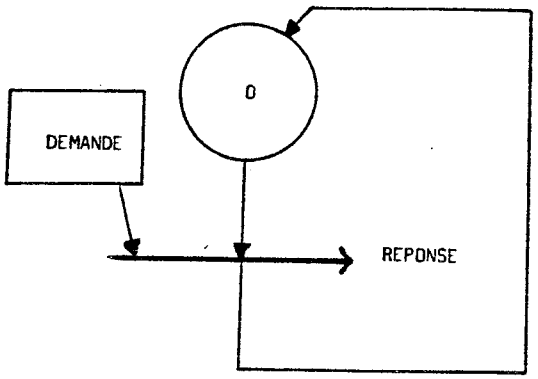


figure 2.3 : Protocle Serveur: transport sans diagnostic

2.3.2/ Transport avec diagnostic négatif

Les erreurs possibles sont les pertes des messages de demande et de réponse auxquelles s'ajoutent les pertes possibles des messages de diagnostic négatif. Nous identifions le message de compte rendu par NACK.

Les enchaînements d'erreurs possibles sont donc les suivants:

1) arrivée de la demande, arrivée de la réponse,
2) arrivée de la demande, perte de la réponse, arrivée du NACK de la réponse,
3) arrivée de la demande, perte de la réponse, perte du NACK de la réponse,
4) perte de la demande, arrivée du NACK de la demande,
5) perte de la demande, perte du NACK de la demande.

Les_reprises_du_client

Nous examinons le comportement du client dans chacune des situations précitées.

1) Le message et la réponse sont correctement parvenus à destination. Le protocole se termine normalement.
2) En principe le client doit supposer que sa demande a été recue par le serveur. Il doit donc attendre un temps suffisant en supposant que la réponse a pu être perdue (réglage de temporisation).
3) Pour le client, ce cas est équivalent au précédent. Mais le serveur n'émettra jamais la réponse.
4) La demande est perdue. Le client reémet une demande.
5) On se trouve dans un cas d'incertitude pour le client car il peut supposer que sa demande est parvenue (pas de NACK) mais il n'obtiendra jamais de réponse. Il peut choisir de réitérer sa demande.

Les_reprises_du_serveur

Nous reprenons l'examen des cas d'erreur.

1) Le protocole se déroule normalement.
2) Le serveur reémet la réponse.
3) Le serveur n'a pas connaissance du début du protocole.
4) Le serveur recoit la demande comme si c'était la première.
5) Le serveur n'a pas connaissance du début du protocole.

Discussion

Avec ce type de service il est possible d'installer un protocole dans lequel le client et le serveur ne reémettent leur message que sur la réception d'un diagnostic.
Cela signifie que lorsqu'un client recoit un message de temporisation dépassée il doit abandonner le protocole en cours ou attendre un nouveau délai pour l'arrivée de la réponse.
On voit aussi qu'il est toujours possible que le service ait été réalisé sans que le client en soit averti. C'est pourquoi cette option de ne pas reémettre ne nous semble pas satisfaisante.

Il nous semble donc préférable de préconiser la reémission de la demande par le client, afin d'avoir le maximum de chances de voir le protocole se dérouler normalement.

Le protocole retenu est donc le suivant:

Pour le client:

1) - Emettre la demande,

2) - Attendre la réponse,

3) - SI réponse ALORS FIN

4) - SI NACK ALORS aller en 1,

5) - SI temporisation échue ALORS aller en 1).

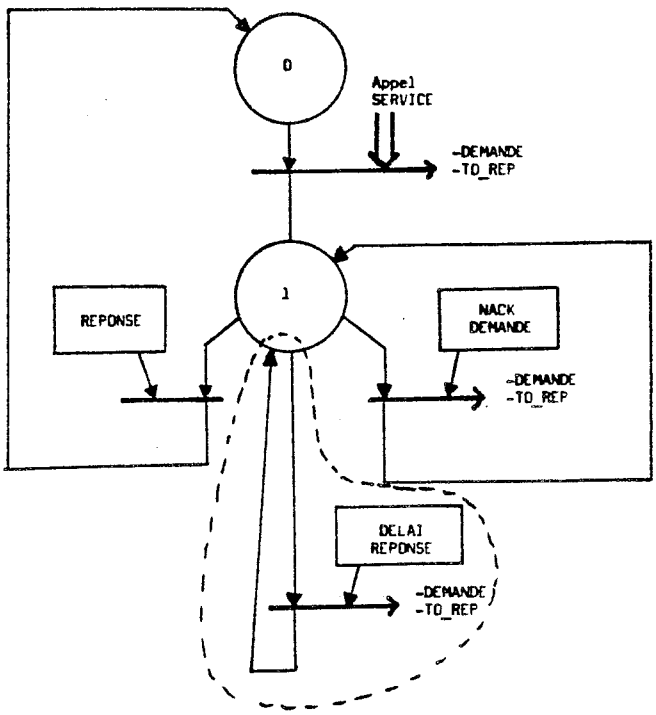


figure 2.4 : Protocole Client: transport avec diagnostic négatif

0 = état de repos,

1 = attente de réponse ou NACK ou délai d'attente de réponse

La partie entourée de pointillés montre ce qui est est ajouté pour permettr de reprendre le protocole à la demande.

Et pour le serveur

1) - Attendre demande,

2) - SI NACK ALORS aller en 4,

3) - SI pas doublon ALORS exécuter service,
SINON détruire demande,

4) - envoyer réponse, aller en 1,

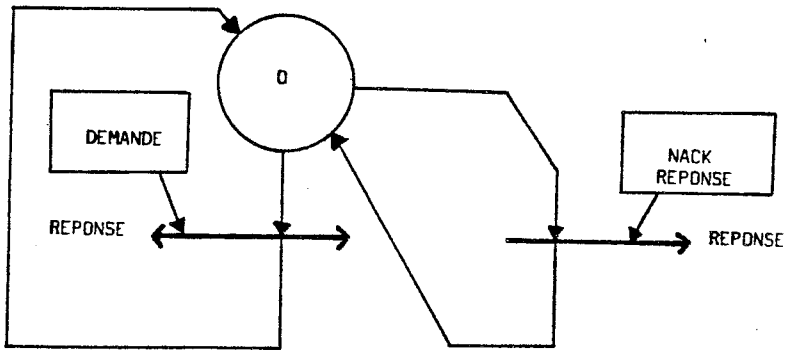


figure 2.5 : Protocole Serveur: transport avec diagnostic négatif

2.3.3/ Transport avec diagnostic positif

Chaque message est acquitté dès son arrivée dans la file d'attente de la porte réceptrice. Nous parlerons donc ici du message et de son diagnostic ACK. Ce type de transport conduit l'utilisateur à décider d'une reémission s'il ne recoit pas l'ACK de son message.

Les_erreurs_possibles

Nous dénombrons comme dans les autres cas toutes les situations probables:

1) la demande et son diagnostic arrivent, la réponse et son diagnostic également,
2) la demande et son diagnostic arrivent, la réponse également mais pas son diagnostic,
3) la demande et son diagnostic arrivent, la réponse n'arrive pas,
4) La demande arrive, le diagnostic pas, la réponse et son diagnostic arrivent,
5) la demande n'arrive pas.

Les_reprises_du_client

Le client doit d'abord faire un choix qui lui est propre en ce qui concerne les temporisations. Doit-il attendre l'ACK avant la réponse ou les deux indifféremment? Cela nous conduit à définir deux temporisations T01 et T02.

La temporisation T01 sert à attendre l'arrivée de l'ACK ou de la réponse.

L'arrivée de l'ACK désarme T01 et le client arme alors T02 pour attendre la réponse.

Si T01 arrive à échéance, le client reémet la demande.

Si T02 arrive à échéance, le client attend la réponse durant un nouveau délai égal à T02.

Les_reprises_du_serveur

Le serveur reémet la réponse s'il ne recoit pas l'ACK au bout d'un temps T03.

Discussion

Dans ce cas le client et le serveur ont la garantie d'être informés que leur interlocuteur a recu leur message. Ce qui permet de réagir plus vite aux incidents de transport. Mais il peut y avoir très fréquemment des doublons chez l'un et l'autre. On peut décrire le déroulement du protocole C-S dans ce cas de transport de la façon suivante:

Pour le client:

1) - Emettre la demande.

2) - Attendre pendant T01 l'ACK ou la réponse.

3) - SI ACK ALORS désarmer T01

4) - attendre pendant T02 la réponse

5) - SI réponse alors aller en 8

6) - SI temporisation T02 alors aller en 4

7) - SI temporisation T01 alors aller en 1

8) - continuer en surveillant les doublons de réponse.

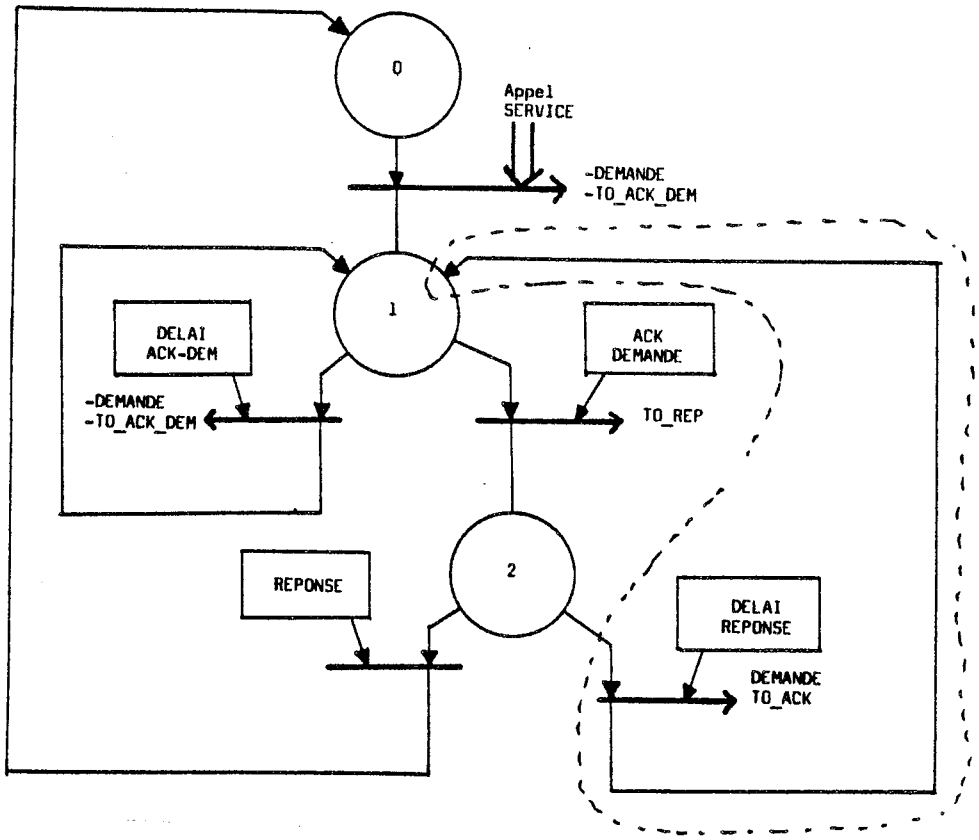


figure 2.6 : Protocole Client: transport avec diagnostic positif

0 = repos,

1 = attente de l'ACK ou du délai de l'ACK,

2 = attente de la réponse (la partie en pointillé a été ajoutée pour rendre le protocole plus résistant aux pannes.

Pour le serveur:

1) - Attendre demande

2) - SI pas doublon ALORS exécuter service
SINON détruire demande

3) - envoyer réponse

4) - attendre ACK pendant T03

5) - SI ACK ALORS aller en 1

6) - SI TO3 ALORS aller en 3

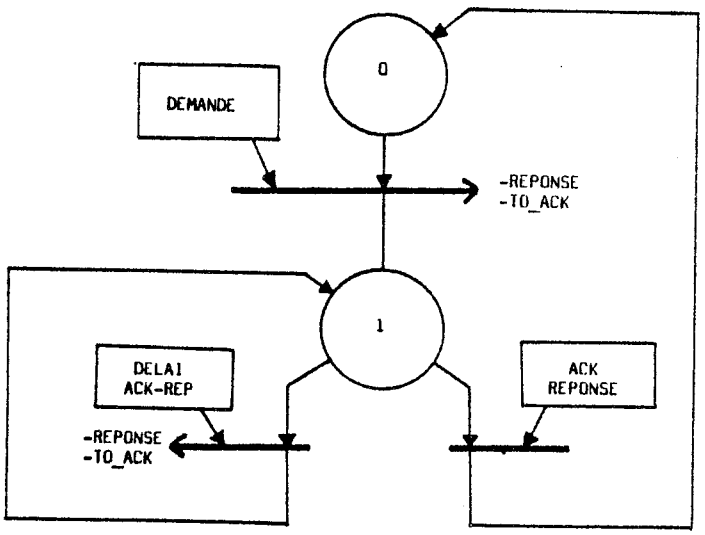


figure 2.7 : Protocole Serveur: transport avec diagnostic positif

2.3.4/ Conclusions partielles

Nous venons de montrer que le protocole C-S pouvait prendre des aspects plus ou moins complexes suivant le type de transport offert. Nous rappelons que ce protocole ne s'applique qu'à des échanges ponctuels.

A ce stade de l'analyse nous pouvons faire trois remarques portant sur:
le nombre de messages générés pour le protocole C-S,
l'utilisation des temporisations par le client et le serveur,
les effets de bord des reprises effectuées par l'un et l'autre.

Le nombre de messages générés

Il est réduit dans le cas du transport sans diagnostic, mais le transport avec diagnostic positif double leur nombre ce qui augmente les possibilités d'incidents de transport et donc de défaillances possibles sur le protocole C-S.

Utilisation des temporisations

Il n'y a qu'une temporisation armée par le client dans le cas du transport sans diagnostic et du transport avec diagnostic négatif.

Il y en a un deux chez le client dans le cas du transport avec diagnostic positif et une chez le serveur.

Dans Chorus, les temporisations sont gérées de telle façon que l'on peut en utiliser un grand nombre mais ce mécanisme reste tout de même délicat à régler.

Les effets de bord

Aucun des types de transport ne garantit à 100% à l'utilisateur que le premier message émis est arrivé ou non. Cela conduit le client à doubler le message de demande, la charge incombant au serveur de détruire les doubles. Dans le cas du transport avec diagnostic positif, il y a des doublons chez le client et le serveur. On peut admettre l'installation du mécanisme d'élimination chez le serveur qui doit gérer l'unicité de l'éxécution du service, mais l'imposer chez tous les clients nous semble peu intéressant du point de vue du fonctionnement de l'acteur client.

Nous serions donc tentés de dire que le transport sans diagnostic est le plus souhaitable parce que:

- la temporisation est nécessaire dans tous les cas,
- aucun des cas ne permet d'éviter les doublons,
- le transport sans diagnostic permet de minimiser le nombre de messages échangés.

Mais dans ce cas le protocole C-S est entièrement soumis à la fiabilité des transmissions. Nous devons donc, pour étayer cette

première intuition, faire une évaluation statistique des conséquences des erreurs sur le déroulement des différents protocoles C-S.

Nous allons évaluer en termes de probabilité de succès ou d'échec, les différents transferts de messages nécessités par le protocole et tenter de donner une indication sur les risques encourus dans chacun des cas de transport.

2.4/ Evaluation du protocole C-S

2.4.1/ Présentation des résultats

L'évaluation du protocole C-S porte sur le calcul des statistiques d'arrivée ou de perte des messages du protocole dans chacun des transport étudiés. Les calculs de ces probabilités est reporté en annexe 2.

Nous en déduisons des statistiques sur l'abandon ou les doublements de messages qui sont récapitulés dans le tableau suivant. Le taux de perte supposé du réseau est de 1 message sur 1000.

Probabilites	sans Diagnostic	Diagnostic negatif	Diagnostic positif
Abandon	10**-3	0.98x10**-3	< 10**-3N
doublon demande	10**-3	0.90x10**-3	10**-3
doublon reponse	0	0	2x10**-3

figure 2.8 : Tableau récapitulatif des probabilités sur C-S.

2.4.2/ Commentaires

Avec l'hypothèse de 1 perte pour 1000 messages, les transports avec diagnostic n'apportent pas une amélioration sensible des résultats sur le protocole C-S. Les probabilités restent du même ordre de grandeur que le taux de panne.

Si l'on veut trouver un intérêt aux diagnostics, il faut regarder du côté du transport avec diagnostic positif car la probabilité d'abandon (avec service réalisé) tend vers 0 avec les répétitions. Cet aspect des choses n'est pas prépondérant et en contrepartie il y a le risque de doublon chez le client.

Les résultats obtenus ne sont pas surprenants car si le client reémet sa demande, il émet, en fait, l'équivalent d'un NACK pour le serveur. Si bien que n'importe quel protocole C-S avec reémission du client permet à chacun des partenaires de dialoguer dans un mode appel réponse. Dans ce cas on a peu de chance d'abandonner le protocole avec un service non réalisé.

En ce qui concerne les doublons, les résultats sont équivalents pour chacun des modes de transport.

3/ Conclusion

En conclusion nous choisissons d'offrir le mode non connecté sans diagnostic comme service de transport implicite.

Le mode connecté n'est pas exclu, mais il s'agit d'un service optionel du transport dont l'activation doit être demandée par les utilisateurs.

Colbert, qui était un ministre prévoyant, fit planter des forêts d'arbres aux fûts bien droits afin que, trois cents ans plus tard, la marine française soit abondamment fournie en mâts excellents pour ses frégates et ses goélettes. Nous devons être reconnaissants envers la mémoire de Colbert.

CAVANNA
le saviez-vous?(2eme fournée)

CHAPITRE 4: LES NIVEAUX PHYSIQUE ET RESEAU DE CHORUS

INTRODUCTION

Ce chapitre décrit les outils et mécanismes mis en oeuvre pour supporter le service de transport.

Le but de ces mécanismes est d'adapter le service de communication aux caractéristiques physiques du support physique d'une part et d'autre part de construire un service réseau sur lequel pourra être élaboré un service de transport tel que nous l'avons prévu au chapitre 3.

Rappel des fonctions par niveau

Le niveau physique représente la fonction de transfert de zone mémoire.
Le niveau réseau représente la fonction de commutation de message.
Le niveau transport représente la fonction de transport de message.

Avertissement

La présentation et l'organisation des différents niveaux qui sont faites ici respectent le principe de structuration avec les portes et les niveaux. Il ne s'agit pas de spécifications d'implantation. L'exemple d'implantation de l'annexe 3 montre les restrictions d'implantation qui sont possibles, sans trahir les principes du modèle de communication que nous nous sommes fixé.

1/ Le niveau physique

1.1/ Le service de base

Le service de base du niveau physique est le transfert physique des données réalisé soit entre deux zones situées sur la même machine, soit entre deux zones situées sur des machines différentes. Cette différence nous amène à distinguer le transfert local du transfert distant.

1.1.1/ Le transfert local

Le transfert local est la copie ou la réaffectation d'une unité de données depuis la file d'attente d'une porte émettrice vers la file d'attente d'une porte réceptrice locale. Ce transfert s'effectue avec ou sans recopie de la zone suivant le type de machine.

Il est réalisé par le noyau.

Discussion

On construit actuellement des machines qui permettent des échanges locaux efficaces et sûrs grâce à leur principe même de conception et de fonctionnement. La fiabilité du mécanisme de transfert local peut être augmentée (pour devenir quasi-parfaite) par la mise en place de solutions fort simples utilisant les mécanismes de base de la machine.

Ainsi sur une machine segmentée, possédant une unité de gestion mémoire (SM90 par exemple [FIN 81]), il est possible:

- d'avoir une protection mémoire (physique et logique) efficace, ce qui permet d'allouer des zones protégées pour chaque acteur et pour les messages.

- d'éviter les recopies et d'accélérer les transferts de message, en allouant la même zone protégée successivement aux différents propriétaires d'un message. Ce qui permet aussi de ne pas limiter la longueur des files d'attente des portes et donc d'éviter les destructions de messages à cause d'une file d'attente saturée (contrôle de flux local).

Si la machine ne permet pas de protéger la mémoire, le service de communication recopie dans ses propres tampons, les unités de données préparées par les acteurs. Il peut aussi, à la remise, chaîner les messages sur les files d'attente réceptrices. Le problème qui demeure concerne le nombre de tampons de messages disponibles dans le service de transport. Il se résoud en fonction de l'application.

1.1.2/ Le transfert distant

Le transfert distant correspond à la transmission d'un bloc de données depuis une adresse physique sur la machine vers une adresse physique extérieure à la machine et inversement.

Remarque

La tendance actuelle en matière de réseau est à la réalisation d'équipements intelligents dans lesquels sont intégrées les fonctions des couches 1 et 2 du modèle ISO. Ces matériels sont branchés sur le bus de la machine et la communication avec (le ou les) processeur(s) de traitement se fait par l'utilisation d'une zone de mémoire commune, d'un mot d'état et d'interruptions [MAR 81], [SCH 80].

Les contraintes dues à l'accès réseau sont donc réglées généralement par l'utilisation d'un coupleur avec lequel les échanges se font par des transferts dans des zones de mémoire commune. Les opérations de communication avec le réseau, du point de vue physique, se ramènent donc au transfert de zone comme sur la machine.

Le transfert distant est réalisé par une collaboration du noyau et d'un acteur spécialisé dans la gestion du coupleur.

L'acteur spécialisé réserve une zone mémoire dans laquelle seront stockées les données distantes émises ou recues. Les émissions se font généralement par des procédures d'écriture du langage (write, print) et échappent ainsi au noyau.

Les réceptions se font sur interruption selon le scénario suivant:

Tant que la machine recoit des données destinées au même bloc il y a stockage par le programme d'interruption dans la zone réservée pour la reception du message. Lorsque la fin de message est détectée, le programme d'interruption déclenche une interruption Chorus de fin d'entrée/sortie. Le noyau détecte cette interruption et la transforme en un message d'interruption destiné à l'acteur qui a initialisé l'entrée/sortie.

Lorsqu'il traite le message d'interruption, l'acteur sait qu'il peut lire les données dans la zone réservée.

Remarque

Ce mode de fonctionnement n'est réservé qu'aux acteurs système spécialisés dans la gestion d'une d'entrée/sortie. Les acteurs utilisateurs courants n'ont pas accès à ce mécanisme.

1.2/ Interface du niveau physique

1.2.1/ Point d'accès au niveau physique

Le service de transfert local est accessible par la porte "TRANSFERER SEGMENT" [S/Sd/Sp/PTRANSFERT]. Cette porte n'est accessible que par des portes du service de communication (mécanisme de transfert entre portes du service de comunication).

1.2.2/ Protocole d'interface physique/réseau

Il n'y a pas de protocole d'interface particulier. Le dépot d'une unité de données sur la porte "TRANSFERER SEGMENT", constitue une demande implicite.

1.2.3/ Unités de données

L'unité de données de l'interface réseau/niveau physique contient:

- l'adresse de début de la zone à transferer,
- l'adresse de début de la zone qui doit recevoir les données transférées
- la taille de la zone de données à transférer.

1.3/ Fonctions du niveau physique

Le routage physique

Nous ne parlons que du routage physique local fait par le noyau sur son site. Le routage physique distant est réalisé par les équipements spécialisés du réseau.

Le noyau dispose d'une table des portes qui est un ensemble de zones de mémoire réservées. Lorsqu'une porte est créée, le noyau lui attribue une entrée dans la table, ce qui permet d'identifier la file d'attente de la porte. Cette attribution détermine donc le routage des unités de données et le transfert vers l'adresse physique correspond à une écriture dans un emplacement vide de la file d'attente de la porte réceptrice.

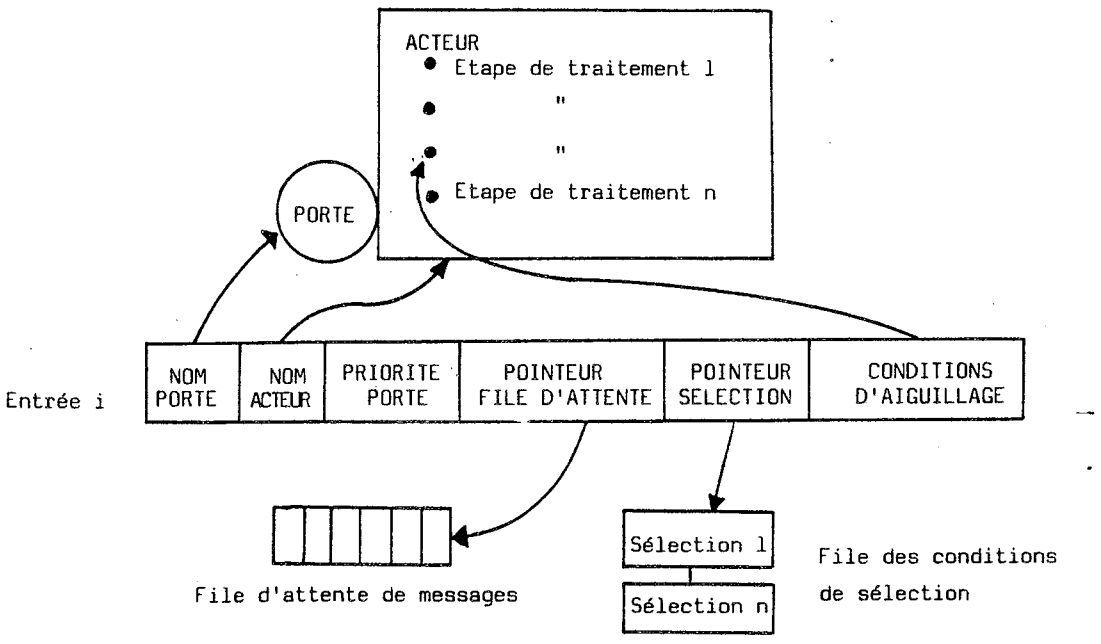


figure 1.1 : Table des portes du noyau

1.4/ Conclusions

Le noyau est la seule entité Chorus du niveau physique.
Le niveau physique fournit un service de transfert de données entre des zones mémoire de la machine, une interface permettant la transformation des évènements physiques en messages pour les niveaux supérieurs, la possibilité d'accéder au mécanisme d'entrée/sortie pour interfacer le réseau et les transferts distants.

Le service de transfert du niveau physique ne nécessite aucun accord préalable entre le noyau et l'entité utilisatrice du service de transfert.

Les transferts locaux peuvent être qualifiés de sûrs en raison de la possibilité d'implanter des mécanismes sûrs de transfert mémoire.

La qualité des transferts distants est fonction des matériels de communication. Ils peuvent être rendus fiables par l'utilisation de procédures de transmission.

2/ Le niveau réseau

2.1/ Rôle du niveau réseau

Le niveau réseau fournit les moyens d'acheminer des unités de données de transport entre les portes du système réparti. Les acteurs, qui communiquent, peuvent se trouver sur la même machine ou sur des machines reliées par un réseau ce qui nous amène à distinguer deux cas:

- la commutation locale de message,
- la commutation distante de message.

2.1.1/ La commutation locale de message

La commutation locale fait passer une unité de données de transport d'une porte émettrice locale à la file d'attente d'une porte réceptrice locale.

2.1.2/ La commutation distante

La commutation distante est tout ce qui concerne l'acheminement à travers un réseau d'une unité de données de transport générée sur un site et destinée à une entité de transport située sur un autre site accessible par le réseau.

2.2/ Les entités du niveau réseau

2.2.1/ Le noyau

Nous avons vu que le noyau dispose d'une table des portes locales qui lui permet de réserver des zones pour la gestion des files d'attente et le stockage des messages. La commutation locale consiste à inscrire dans la table que le message M émis par la porte Pe est maintenant dans la file d'attente de la porte Pr à la place j.

Le noyau est donc tout désigné pour effectuer toutes les opérations locales de commutation de messages.

2.2.2/ L'acteur de commutation distante

Lorsque la porte réceptrice n'est pas locale, la fonction de commutation distante entre en jeu. Cette fonction de commutation distante est réalisée par l'acteur de commutation distante qui est chargé de la gestion du réseau.

Remarque

L'acteur de commutation distante peut faire appel à un acteur réseau, spécialisé dans l'accès à un réseau donné. On peut avoir plusieurs acteurs réseau (connus de l'acteur de commutation distante) qui interfacent plusieurs types de réseaux différents.

L'acteur de commutation distante représente, sur le réseau de communication, la machine sur laquelle il se trouve. Réciproquement il est, sur son site, le représentant de tous les sites avec lesquels il peut communiquer.

Nous ne pouvons pas étudier tous les types de commutation distante possibles, aussi parlerons-nous de l'acteur commutation distante sans

distinguer si c'est lui ou un acteur réseau particulier qui réalise les fonctions décrites.

2.3/ Les services

2.3.1/ Le service de commutation locale

Lorsque la porte réceptrice est locale, les transferts de mémoire à mémoire sûrs (protection mémoire) peuvent être garantis au niveau physique. On peut y ajouter un mécanisme de commutation qui minimise les pertes d'unités de données du service de commutation. En effet, les risques importants de perdre de messages localement déendent de la place mémoire disponible pour créer les messages et de la limitation éventuelle de la taille des files d'attente (en réception) des portes. La saturation de la file d'attente peut entraîner la destruction du message par le noyau.

Il est possible d'utiliser un mécanisme qui contrôle l'allocation des tampons de messages aux acteurs utilisateurs et effectuer un chaînage des messages sur les files d'attente réceptrices. Dans ce cas, le service de communication ne permet la génération que des messages qu'il peut stocker et les files d'attente sont infinies. Il n'y a pas de formule miracle. Tout dépend de la politique de gestion de la mémoire qui varie d'une machine à l'autre en fonction des caractéristiques de l'implantation et de la taille de la mémoire disponible.

Le noyau ne donne aucun diagnostic après une commutation locale. Le service de commutation locale est en datagramme fiable sans diagnostic.

2.3.2/ Le service de commutation distante

La commutation distante pose le problème un peu différemment parce qu'aucune entité du niveau réseau n'a une connaissance et une maîtrise globale des communications. De plus, la qualité du service de commutation distante est différente suivant qu'il est réalisé en datagramme ou sur circuit virtuel.

Sur un réseau à datagramme la qualité sera directement celle du réseau de transmission.
Sur un réseau à circuit virtuel, lorsque le circuit est ouvert la qualité est bonne puisqu'il y a un contrôle d'erreur sur chaque message. On n'évite cependant pas les incidents dus à la rupture du circuit.

On choisit de considérer la commutation distante comme un service réalisé en datagramme. Il y aura simplement moins d'erreurs, statistiquement parlant, si on utilise des circuits virtuels.

Justifications

Dans Chorus, le service fourni aux entités de transport doit être le même quel que soit le type de réseau utilisé. La commutation locale en datagramme est jugée satisfaisante. La commutation distante s'aligne sur ce choix.

Il est toujours possible d'utiliser un protocole d'échanges de paquets qui améliore la commutation distante si elle est trop défectueuse sur un réseau donné. Mais cette situation est extrême car la reprise des erreurs résiduelles de commutation est laissée au niveau transport.

2.4/ Les interfaces

Nous devons distinguer ici:

- les interfaces entre le transport et la commutation,
- les interfaces entre les acteurs de commutation distante à travers le réseau.

2.4.1/ Les points d'accès du service réseau

2.4.1.1/ Commutation locale

Le service de commutation locale est identifié par la porte "COMMUTER MESSAGE" [S/Sd/Sp/PCOMLOC].

2.4.1.2/ Commutation distante

Le service de commutation distante est accessible sur son site par la porte "COMMUTER A DISTANCE" [S/Sd/Sp/PCOMDIST] de l'acteur de commutation distante.
Cette porte n'est accessible que par les entités de transport distant.
Cette porte représente toutes les portes extérieures au site.

2.4.2/ Interface entre acteur de C.D.

Cet interface détermine la vision que les acteurs de commutation distante ont de leurs homologues.

Sur un réseau à datagramme, l'identificateur de l'acteur de commutation distante est le nom global de la porte réseau qui reçoit les messages venant du réseau.

Sur un réseau à circuit virtuel, deux acteurs de commutation distante ne peuvent communiquer que si un circuit a été établi entre eux et

dans ce cas c'est le numéro de circuit qui identifie l'acteur correspondant. A l'établissement, le nom global de la porte réseau de l'acteur commutation distante est utilisé.

2.5/ Les protocoles

2.5.1/ Protocole avec le niveau transport

Le service réseau étant équivalent à la commutation en datagramme sans diagnostic, il n'y a pas de protocole d'accès particulier au service réseau. Il suffit à l'entité de transport d'envoyer un message vers la porte "COMMUTER MESSAGE" ou la porte "COMMUTER A DISTANCE" suivant que la commutation du message doit être locale ou distante.

2.5.2/ Les protocoles du niveau réseau

2.5.2.1/ Commutation locale

Le noyau effectue toutes les opérations de commutation locale, il ne collabore avec aucune entité pour effectuer ce travail, il n'y a donc pas de protocole particulier au cours d'une commutation locale.

2.5.2.2/ Commutation distante

Le seul protocole de niveau réseau actuellement envisagé permet le transfert de données sur un circuit virtuel.

Commut. dist. sur circuit virtuel

Dans le cas d'utilisation d'un réseau à circuits virtuels, deux acteurs de commutation distante ne peuvent communiquer que s'ils ont établi un circuit virtuel entre eux. Il y a donc plusieurs phases pour accéder au transfert sur le réseau.

- L'établissement du circuit qui nécessite que chacun connaisse l'adresse réseau de l'autre.
- Le transfert de données après l'établissement du circuit.

Chacun des acteurs doit respecter le protocole de transfert sur le circuit.

Les unités de données du service de transport distant sont acheminées (du point de vue de l'acteur de commutation distante) comme sur un réseau à datagrammes. Le circuit virtuel joue en quelque sorte le rôle d'une procédure de transmission à travers le réseau. Cela signifie qu'il n'y a aucun contrôle (de flux ou d'erreur) et aucun diagnostic en retour pour le niveau transport.

2.6/ Données de l'interface transport/réseau

2.6.1/ Commutation locale

Dans la commutation locale, la porte "COMMUTER MESSAGE" recoit du niveau transport une unité de données composée comme suit:

SerCL,/Pr,Pe,SERVICE,TEXT/

SerCL = Donnée de l'interface transport/Commut. locale

/../ = Unité de données du transport local

Pr : porte utilisateur réceptrice de TEXT,

Pe : porte utilisateur émettrice de TEXT,

SERVICE : service demandé au niveau transport par l'utilisateur.

Dans la mesure où le noyau ne rend pas de service plus élaboré que la commutation locale sans possibilité de choix sur la qualité le service demandé sur la porte "COMMUTER MESSAGE" est implicite.

2.6.2/ Commutation distante

L'unité de données de l'interface transport/réseau remise sur la porte "COMMUTER A DISTANCE" est composée comme suit:

Sr,Se,/Ptdr,Ptde,SerTD,Pr,Pe,SerT,TEXT/

Sr = Site récepteur de l'unité de données de transport,

Se = Site émetteur de l'unité de données de transport,

/../ = Unité de données de transport,

Ptdr = Porte de transport distant réceptrice,

Ptde = Porte de transport distant émettrice,

SerTD = Données du service de transport distant,

Pr = Porte utilisateur réceptrice de TEXT,

Pe = Porte utilisateur émettrice de TEXT,

SerT = Données de service du transport.

2.7/ Les fonctions du niveau réseau

2.7.1/ Le routage local

Le routage local est fait par le noyau qui dépose le message soit sur la porte réceptrice locale (trouvée grâce à la table des portes locales), soit sur la porte par défaut du service de transport distant.

2.7.2/ Le routage distant

L'acteur de commutation distante réalise la fonction de routage réseau distant. Il s'agit donc, pour le service de commutation distante, d'établir une correspondance entre le nom du site récepteur de l'unité de données de transport qui est à acheminer et l'adresse réseau de l'acteur de commutation distante de ce site.

Plusieurs techniques sont envisageables.

a) Pour localiser les différents acteurs de commutation distante du système, l'acteur de commutation distante possède une table de correspondance entre les noms des sites Chorus et l'adresse sur le réseau de leur acteur de commutation distante.

b) Dans le cas d'un réseau à diffusion et à canal unique, il est possible de ne pas conserver une table des correspondances site/adresse réseau des acteurs de commutation distante. En effet, il est possible en un seul message, grâce à une adresse de diffusion commune à tous les acteurs de commutation connectés au réseau, de distribuer le message à tous les sites. Les acteurs de commutation effectuent un filtrage et conservent les unités de données qui sont destinées à des portes de leur site et détruisent les autres.

Remarques

Dans les concepts de Chorus, il n'y a pas de choix fait en faveur d'une technique plutôt qu'une autre. Cela dépend du type de réseau choisi et des performances souhaitées.

Au moment du routage, le message Chorus est transformé en un message réseau dont l'adresse de destination est différente suivant que l'on utilise le datagramme ou le circuit virtuel.

A titre d'exemple on peut citer les deux cas suivants:
- dans un réseau à datagrammes, l'adresse réseau est l'adresse réseau de l'acteur de transport distants du site de la porte réceptrice,
- dans un réseau à circuits virtuels, l'adresse est le numéro du circuit virtuel ouvert entre le site émetteur et le site de la porte réceptrice (on suppose le circuit ouvert).

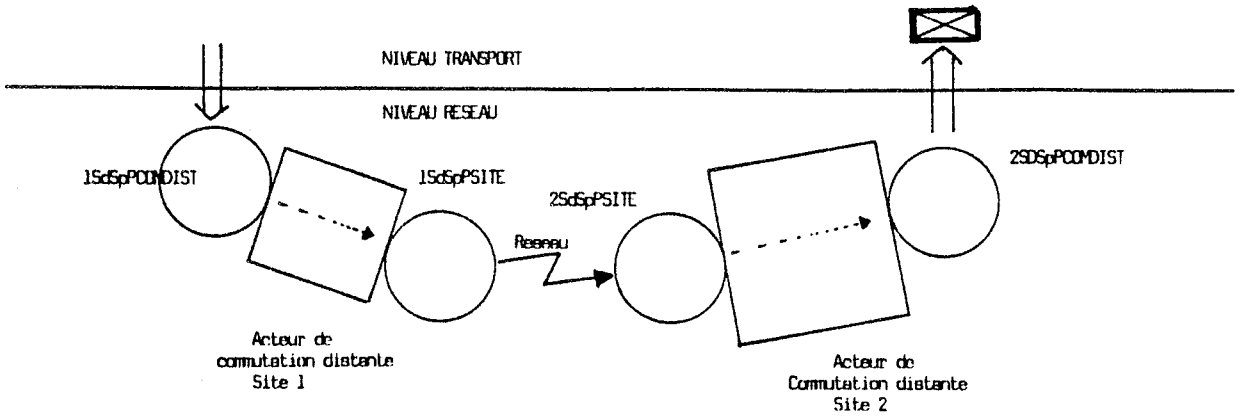


figure 2.1 : La commutation distante

2.2.2.1/ Relais

Le relais est une particularité du routage. Une machine à la possibilité de relayer vers une autre un message qui n'est pas destiné à une de ses portes locales. Le mécanisme de commutation locale permet d'offrir cette propriété puisque le noyau remet un message sur la porte du service de transport lorsque la porte réceptrice de ce message n'est pas locale. Dans ces conditions, un message reçu sur une machine qui n'aura pas pu être remis localement repartira vers la porte de transport distant. Il sera éliminé ou redonné à l'acteur de commutation distante. Lors de l'implantation on veillera cependant à éviter les bouclages des unités de données du service réseau.

2.7.3/ Segmentation et groupage

2.7.3.1/ dans la commutation locale

La commutation locale ne passe pas par un réseau, il n'est donc pas nécessaire de faire de la segmentation des messages qui restent sur le site.

2.7.3.2/ Dans la commutation distante

Pour le moment, la segmentation n'est faite qu'à un seul niveau dans Chorus. Le niveau transport semble le meilleur choix dans notre approche:

- parce qu'il détermine la localisation des récepteurs distant et peut donc négocier avec eux la taille des données échangées,
- on peut définir une taille standard des unités de données de transport distant compatible avec tous les types de réseaux utilisés, le service de transport adapte les unités de données utilisateur à ce format (segmentation) et effectue toutes les opérations et contrôles afférants (réassemblage) à cette opération.

Il n'y a donc pas de segmentation au niveau réseau de CHorus.

2.7.4/ Le séquencement

Commutation locale

Le noyau Chorus commute séquentiellement les messages en respectant l'ordre dans lequel ils ont été générés par l'acteur émetteur. Il procède de même pour les messages d'interruption.

Les files d'attente des portes sont traitées en PEPS (premier entré/premier sorti) lors de l'activation (sauf indication contraire des acteurs par les selections).

Commutation distante

Le maintien en séquence des unités de données de transport ne peut être garanti que si le réseau maintien les messages en séquence [BOG 80].
Dans le cas d'un réseau à circuits virtuels, il y a maintien de la séquence ce qui permet à l'acteur de commutation de remettre les messages dans l'ordre d'émission sur le site distant.

Dans le cas d'un réseau à datagrammes les paquets peuvent arriver en désordre. Le service de commutation distante n'assure pas leur remise en reséquencement.

Il n'y a pas actuellement de fonction de séquencement prévue pour l'acteur de commutation. Le niveau réseau conserve donc l'ordre de tous les messages locaux qui sont restés locaux. Pour les messages provenant d'autre sites la commutation locale ne tient compte que de leur ordre d'arrivée sur le site.

Il existe des algorithmes de maintien d'un ordre total, mais il n'a pas paru essentiel, jusqu'à maintenant, de les utiliser dans Chorus (Estampilles, horloges logiques, séquenceur) [BAN 79,HER 78,HOL 78,LAM 78]

2.7.5/ Délai de transit

Le problème du maintient d'un délai de transit maximum ne se pose pas dans les mêmes termes suivant que l'on utilise un réseau maillé ou non.

Dans les réseaux non maillés, on peut considérer qu'un message qui n'est pas arrivé au bout d'un temps fini est perdu car il n'y a pas de risque de bouclage dans le réseau. On ne mettra pas en place de mécanisme de détection et d'élimination de paquets trop vieux (une

exception dans le cas où l'on permet le relais par l'acteur de commutation; dans ce cas il devra repérer les paquets qui bouclent).

Dans les réseaux maillés, le service réseau se charge en général du l'élimination des paquets trop vieux (Réseau Cigale, par exemple) [GRA 79, SLO 79, WAT 80]. Dans ce cas il n'est pas nécessaire d'ajouter cette fonction à l'acteur de commutation.

Dans la commutation locale on peut penser qu'il n'y a pas de problème de délai de transit.

2.8/ Conlusions sur le niveau réseau

Evaluation de la qualité du service réseau

Cette évaluation est nécessaire pour définir les caractéristiques du service de transport.

Interviennent:

a) le taux d'erreur résiduel qui influence le choix des services de transport à définir dans Chorus. Dans le cas de l'utilisation d'un réseau à datagrammes, ce taux peut être non négligeable. Dans le cas de l'utilisation d'un réseau à circuits virtuels, il peut être ramené à des valeurs très faibles.

   Dans les deux cas, le taux d'erreur ne pourra pas être nul. On ne pourra donc pas toujours se passer de protocole de transport plus complexe que le datagramme sans diagnostic.

b) le débit, critère important dans le cas de la commutation distante. Un débit de commutation faible sur le réseau peut entraîner des perturbations pouvant aller jusqu'à la saturation de la "mémoire de messages" sur les sites. Lorsque les débits seront importants, le service de transport devra offrir un

mécanisme de régulation du flot d'émission de messages pour garantir un transport fiable (en évitant les pertes dues à la congestion du service de communication).

c) le délai de transit est borné mais il varie entre des bornes éloignées car la commutation locale est très rapide et la commuation distante peut être très lente.
Cela est de plus en plus souvent dû au routage en cascade et de moins en moins aux transmissions qui atteignent des vitesses de plusieurs millions de bits/seconde sur les réseaux locaux actuels. (le routage local est rapide mais le routage distant peut demander l'utilisation de plusieurs mécanismes de routage généralement coûteux en temps).

f) la longueur des messages (qui peut être beaucoup plus grande sur une machine que celle admise sur le réseau) peut entraîner des besoins de segmentation et regroupage au niveau transport.

Nous pensons avoir décrit les éléments de base du service de communication. La présentation qui en a été faite tend surtout à montrer comment on peut prendre en compte les différentes contraintes liées au choix des interfaces. Nous n'avons pas décrit de protocoles particuliers parce que nous n'estimons pas en avoir de particulièrement originaux à proposer. Le lecteur aura vu, nous l'espérons, les endroits où se situent les problèmes du choix d'un protocole et quelles justifications on peut apporter à un choix donné pour une situation donnée.

(*) Un exemple d'implantation du noyau et de l'acteur de commutation sur une machine à base d'INTEL 8086 sont donnés en annexe 3.

CHAPITRE 5: LE NIVEAU TRANSPORT DE CHORUS

INTRODUCTION

Ce chapitre décrit le niveau transport du service de communication.

Nous avons conclu au chapitre 3 que la plupart des applications pouvaient se contenter d'un service de transport en mode datagramme si la fiabilité de celui-ci était satisfaisante et si le délai de transit était borné.

Nous avons vu aussi que certains types d'échanges demandent une qualité de service qui ne peut être obtenue qu'au prix d'un service de transport plus élaboré.

Dans ce chapitre nous spécifions un service de transport en mode non connecté, puis nous élaborons un service de transport en mode connecté qui est construit sur le transport en datagramme.

Pour rendre le service de transport, il faut réaliser une application répartie qui permet de transporter les messages à travers le système. Nous donnons les entités et les mécanismes mis en place dans Chorus pour exécuter la fonction de transport à la fin de ce chapitre.

1/ Les services de transport

1.1/ Le transport en mode non-connecté

Le mode de transport sans connexion est défini pour répondre aux besoins d'échanges simplifiés entre acteurs utilisateurs. Cela ne correspond pas à un service dégradé. Simplement la redondance d'informations, au niveau utilisateur, apportée par certaines coopérations (appel avec réponse) ou encore par la répétition d'un évènement identique (messages échangés périodiquement et souvent identiques dans le temps) ne nécessitent pas les complications introduitent par un protocole de transport élaboré.

1.1.1/ Le service

Le service de transport en mode non connecté est _en datagramme sans diagnostic.

Il permet l'envoi de lettre d'une taille maximale qui dépend des choix d'implantation.

Il assure un délai de transit borné pour les unités de données utilisateur qui lui sont confiées (nous ne développerons pas d'avantage ce principe car l'utilisation ou non d'un mécanisme de datation et d'élimination des paquets hors délai dépend du système physique choisi).

1.1.2/ Interface entre transport et utilisateur

1.1.2.1/ Point d'accès

Le point d'accès au service de transport en mode non connecté est identifé par la porte "TRANSPORTER MESSAGE" [S/Sd/Sp/PTRANSLOC]. Cette porte recoit tous les messages préparés par un acteur utilisateur au cours de son étape de traitement.

1.1.2.2/ Unité de données de l'interface U./I.

L'unité de données déposée par l'acteur utilisateur sur la porte "TRANSPORTER MESSAGE" est composée comme suit:

Pr,Pe,SERVICE,/TEXT/

Pr = Porte réceptrice de l'unité de données utilisateur,

Pe = Porte émettrice de l'unité de données utilisateur,

SERVICE = Paramètres du service de transport demandé,

/../ = Unité de données de l'utilisateur
TEXT données de l'utilisateur.

SERVICE représente le type de transport et les paramètres qui lui sont associés. Les paramètres de service peuvent indiquer la taille des données utilisateur, ou des options choisies du service de transport, si celui-ci en propose. Dans le cas du transport en mode non connecté le paramètre de service peut se limiter à la définition du type du message ("DTG" = datagramme sans diagnostic) et à la taille du message si celui-ci peut être de longueur variable.

Remarque

Toutes les unités de données de l'acteur utilisateur ont le format précité. Dans le cas d'un service de transport plus élaboré, les paramètres de service deviennent plus nombreux.

1.1.2.3/ Protocole d'accès au mode non connecté

Le transport en mode non connecté est le service implicite du transport. Il ne retourne aucun diagnostic, l'acteur utilisateur n'a donc aucun protocole particulier à suivre.

1.1.3/ Fonctions du mode non connecté

Le mode non connecté permet l'échange de lettres de longueur fixée par le service de communication. Il garantie la remise si le destinataire est correct. Il réalise un routage des unités de données utilisateur transparent aux acteurs utilisateurs. Nous en verrons les caractéristiques dans l'étude des mécanismes de base du transport à la fin de ce chapitre.

Le mode non connecté n'assure pas le séquencement des messages des acteurs utilisateurs.

1.2/ Conclusions sur le mode non connecté

Les spécifications que nous venons de donner ne renseignent que sur la vision qu'ont les acteurs utilisateurs du service de transport en mode non connecté (nous les verrons plus loin, grâce à l'étude du transport local et du transport distant).

Le service de transport en mode non connecté est considéré comme fiable et il ne demande aucun protocole particulier. C'est le service implicite rendu par le transport aux acteurs utilisateurs.

1.3/ Le transport en mode connecté

Le service de transport en mode non connecté ne permet pas de contrôler des échanges de bout en bout entre portes utilisatrices et surtout de faire du contrôle de flux entre portes utilisatrices.

Le contrôle de flux devient nécessaire lorsqu'il y a un important débit de données entre des acteurs situés sur des machines différentes. Ceci afin surtout que le transport puisse toujours fonctionner dans des conditions acceptables qui garantissent la qualité de service annoncée. Une congestion au niveau transport peut entraîner des pertes de messages plus importantes qui dégraderaient significativement le mode non connecté.

Nous avons décidé d'offrir un service de "transport sur connexion" entre acteurs utilisateurs afin de conserver (au niveau exploitation) le contrôle des communications. Rien n'interdit à des utilisateurs de préférer un protocole d'échange sur connexion privée.

Le service de transport en mode connecté est considéré comme une facilité optionnelle du niveau transport dont l'accès doit être explicitement demandé par les acteurs utilisateurs qui veulent en bénéficier.

1.3.1/ Définition de la connection U(tilisateur)

La connexion U de Chorus est une voie logique de communication établie entre deux portes d'acteurs Utilisateurs.
Elle est bidirectionnelle. Les facilités qu'elle offre sont le contrôle d'erreur, le contrôle de flux et le séquencement des messages utilisateur.

1.3.2/ Le service de transport sur connexion U

La connexion U est le service du mode connecté. Elle est accessible par tous les acteurs utilisateurs quelque soit leur localisation.

Le service de transport sur connexion U offre le moyen d'échanger des données de façon organisée et synchronisée entre acteurs utilisateurs.
Pour cela:

- il assure la transparence des moyens de transport pour les acteurs utilisateurs.
- il permet l'optimisation des moyens de traitement en simplifiant les contrôles nécessaires, dans les acteurs utilisateurs, pour les échanges.
- il optimise l'utilisation du service de transport (en evitant les surcharges inutiles du transport par des messages qui ne sont pas prêts à être pris en compte à l'arrivée).

Les services fournis pour la connexion U sont:
- l'ouverture de connexion,
- le transfert de données sur connexion,
- la fermeture de connexion.

1.3.3/ Interface de transport sur connexion U

1.3.3.1/ Point d'accès

Nous avons pris le parti, pour l'instant, de considérer que l'utilisateur doit avoir la visibilité de la connexion U. En conséquence, il n'envoie pas directement ses messages à son partenaire utilisateur mais à une porte du service de transport en mode connecté.

L'interface de transport sur connexion U est constituée par:
- les portes du service de transport qui permettent l'ouverture, la fermeture de la connexion, ainsi que les portes du service qui recoivent les données de l'utilisateur à transmettre.
- les messages échangés entre le service de transport et les acteurs utilisateurs.

Les portes d'accès au service de transport

Le service de transport sur connexion U est accessible par la porte [S/Sd/Sp/PTCXU] que nous noterons PTCXU (porte transport sur connexion U) connue de tous les acteurs. Cette porte reçoit tous les messages des acteurs utilisateurs lors de leur premier accès au service de transport (par exemple, les demandes d'ouverture).

Lorsque la connexion U est établie, le service de transport ouvre une porte qui lui est dédiée: la "porte privée de connexion U" [S/Sd/Sp/pPCXU] que nous noterons pPCXU. L'acteur utilisateur, en connexion U, devra envoyer tous ses messages sur la porte pPCXU de sa connexion U, à l'exclusion de toute autre porte.

Remarque:
La porte privée de connexion U est l'identificateur unique de l'extrèmité d'une connexion U. Elle permet au service de transport

d'effectuer un contrôle sélectif sur les différents acteurs en connexion U. Elle évite les interférences entre les acteurs utilisateurs qui sont soumis à des contrôles de flux différents.

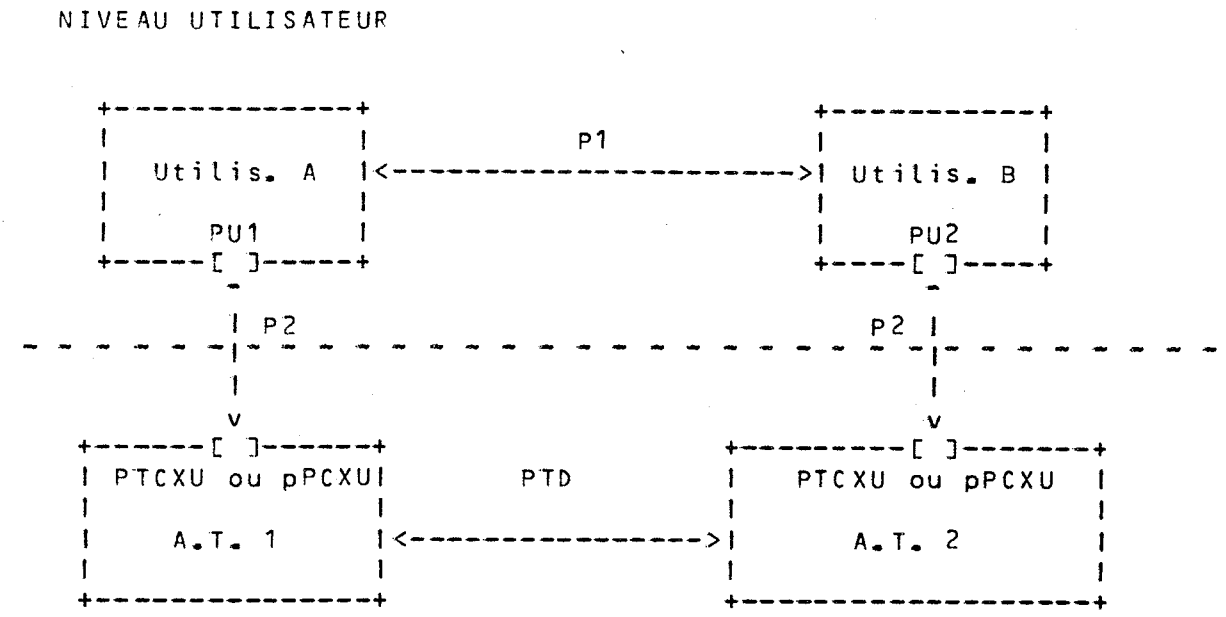
En effet, dans le cas où on utiliserait une seule porte pour plusieurs connexions U, en raison du principe des files d'attente et du déclenchement des traitements par les messages, un acteur qui ne respecterait pas le contrôle de flux imposé par le service de transport pourrait saturer une file d'attente et perturber les échanges des autres connexions utilisant la même porte.

On peut comparer la porte privée à la socket utilisée dans ARPA.

1.3.4/ Le protocole d'accès au mode connecté

1.3.4.1/ Situation du protocole

Le protocole d'accès au service de transport pour les connexions U définit les règles du dialogue entre les acteurs utilisateurs et le service de transport. La figure suivante montre la façon dont ce protocole s'intègre dans l'ensemble des protocoles des niveaux utilisateur et transport.



- P1 = protocole d'application entre Utilisateurs,
- P2 = protocole d'accès au transport sur connexion U,
- PTD = protocole de transport distant entre acteurs de transport
- AT1 et AT2 sont les deux acteurs de transport de deux sites différents.

figure 1.1 : Les protocoles mis en jeu sur connexion U

1.3.4.2/ Principes généraux

Dans ce protocole l'acteur de transport "informe" et "pilote" les acteurs utilisateurs. Ses messages sont donc vus par l'acteur utilisateur comme des compte-rendus ou des indications sur les opérations du service.

Un acteur utilisateur, quant à lui, est un client et à ce titre ses messages sont considérés par le service de transport comme des commandes ou des demandes de services.

1.3.4.3/ Conventions

Nous allons décrire la façon dont se déroule une connexion U entre deux acteurs utilisateurs (de l'ouverture à la fermeture). Pour clarifier l'exposé on utilise:

- une dénomination des portes qui permet de repérer les portes du service de transport,
- une codification des types d'unités de données de l'interface utilisateur/transport échangées entre les acteurs utilisateurs et le service de transport.

1.3.4.3.1/ Les portes Utilisateur et transport

Les portes des acteurs Utilisateurs sont PA et PB. Les portes du service de transport sont:

- PTCXU = porte d'accès au service de transport,
- pPCXU = porte privée réservée à la connexion U.

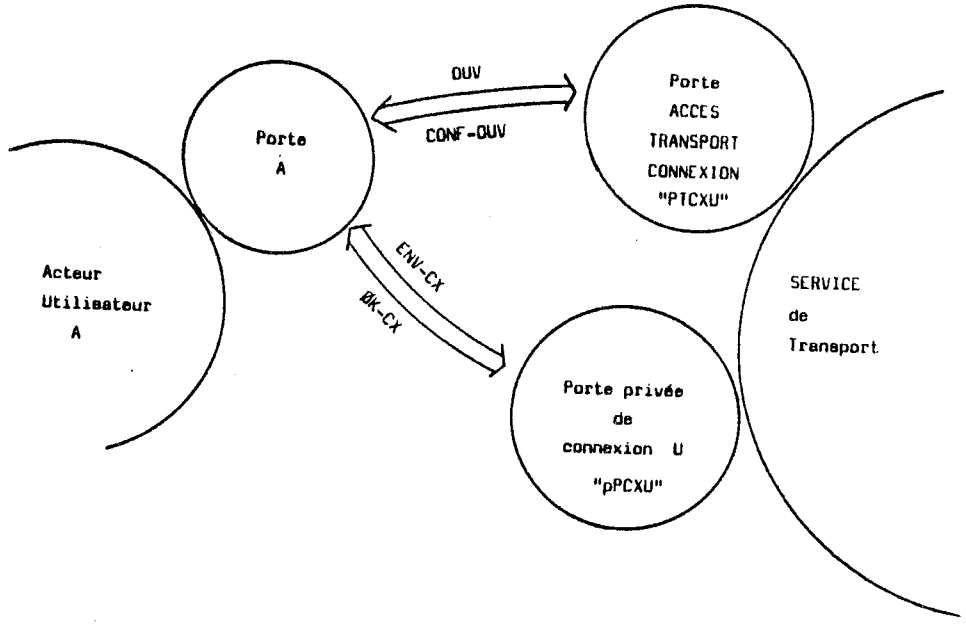


figure 1.2 : Les portes Utilisateur et transport

1.3.4.3.2/ Les codes de messages du protocole

Les conventions utilisées pour identifier les types des unités de données de l'interface utilisateur/transport sur connexion sont les suivantes:

MESSAGES UTILISES PAR L'UTILISATEUR EXCLUSIVEMENT

MC_OUV_CU = message de demande "d'ouverture de connexion U",
MC_COU_CU = message de "confirmation d'ouverture de connexion U",
MC_FER_CU = message de demande de "fermeture de connexion U",
MC_CFE_CU = message de "confirmation de fermeture de connexion U"
MC_ENV_CU = message de demande "d'envoi sur connexion U",

MESSAGES_UTILISES_PAR_LE_SERVICE_DE_TRANSPORT_EXCLUSIVEMENT

MC_IN_OU_CU = message d'indication de "demande d'ouverture de connexion U",

MC_IN_CO_CU = message d'indication de "confirmation d'ouverture de connexion U",

MC_IN_FE_CU = message d'indication de "demande de fermeture de connexion U",

MC_IN_CF_CU = message d'indication de "confirmation de fermeture de connexion U",

MC_IN_EV_CU = message d'indication de "envoi distant sur connexion U",

MESSAGES_UTILISES_PAR_L'UTILISATEUR_ET_LE_SERVICE_DE_TRANSPORT

MC_IN_OK_CU = message d'indication de "prêt à recevoir"

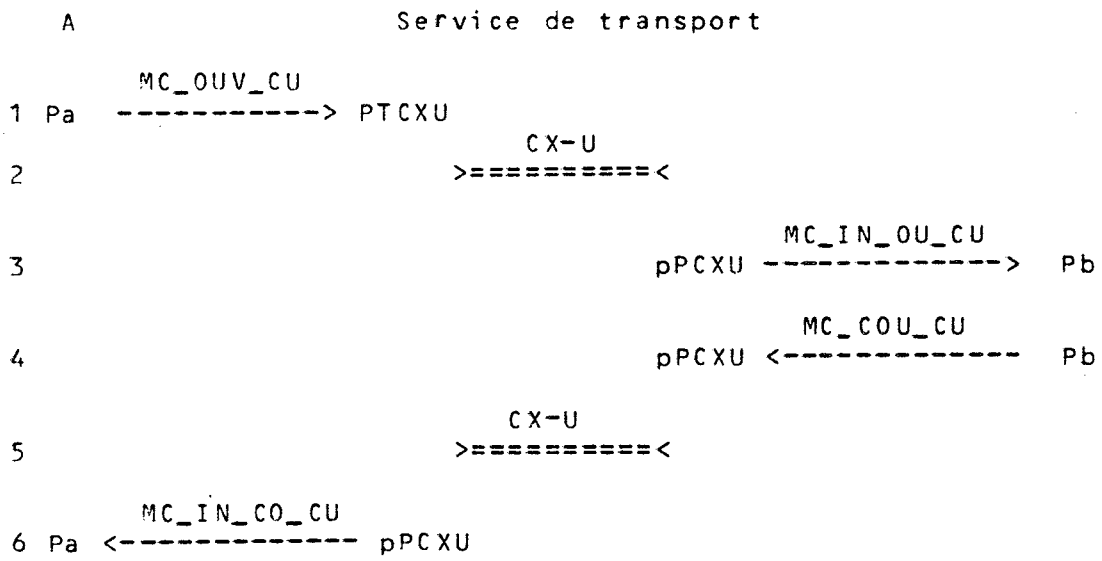
Tous les messages dont ils est question ici seront justifiés dans la présentation du protocole de connexion U.

1.3.4.4/ Description du protocole

Pour décrire le protocole nous considérons le cas de deux acteurs utilisateurs, A et B, qui demandent l'établissement d'une connexion U entre leur portes PA et PB. C'est A qui a l'initiative de l'échange. Le service de transport est représenté comme un intermédiaire entre les deux acteurs utilisateurs.

1.3.4.4.1/ Ouverture

On se place dans le cas où l'ouverture se déroule normalement.



La connexion CX-U représente les opérations internes au service de transport pour acheminer le message de demande d'ouverture vers la porte distante.

figure 1.3 : Ouverture de connexion U

Description des étapes

1) L'acteur A envoie au service de transport AS une demande de connexion U avec l'acteur B.
2) Le service de transport AS transmet le message de demande de façon interne sur CX-U. Il ouvre une porte privée vers B pour la connexion en cours d'ouverture.
3) Le service de transport indique à l'acteur B que A a demandé l'ouverture d'une connexion U.
4) L'acteur B accepte l'ouverture par une confirmation à

l'adresse du service de transport sur la porte privée qui lui a été indiquée.

5) Le service de transport transmet de façon interne, sur CX_U, la réponse de B à la demande d'ouverture de A. L'ouverture est acceptée, il ouvre donc une porte privée de connexion pour A. de A.

6) Le service de transport indique à A que B a confirmé la demande d'ouverture de connexion U.

Caractéristiques de l'établissement

a) Il n'est pas possible de faire transporter des données utilisateur avec le message d'établissement de la connexion U (choix pour le moment).

b) Dès qu'ils sont avisés de l'ouverture de la connexion U, tous les acteurs peuvent émettre au moins un message de données.

1.3.4.4.2/ Fermeture

Dans ce cas on considère que la fermeture se déroule normalement.

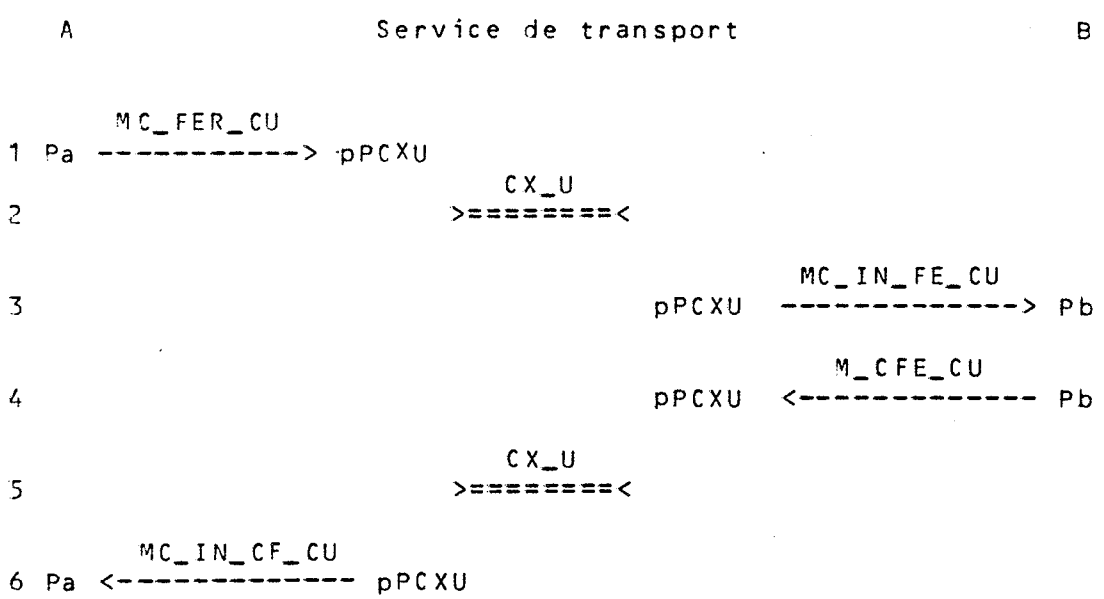


figure 1.4 : Fermeture de connexion U

Caractéristiques de la fermeture de connexion U

a) Il n'est pas possible de faire transporter des données U avec les messages de fermeture de connexion U.

b) Tous les messages en transit sur la connexion U sont délivrés à l'acteur utilisateur final par le service de transport (si cela est possible) avant le message de fermeture.

c) L'utilisateur ne peut plus émettre de message sur la connexion U après avoir reçu l'indication de fermeture ou envoyé la confirmation de fermeture de connexion U.

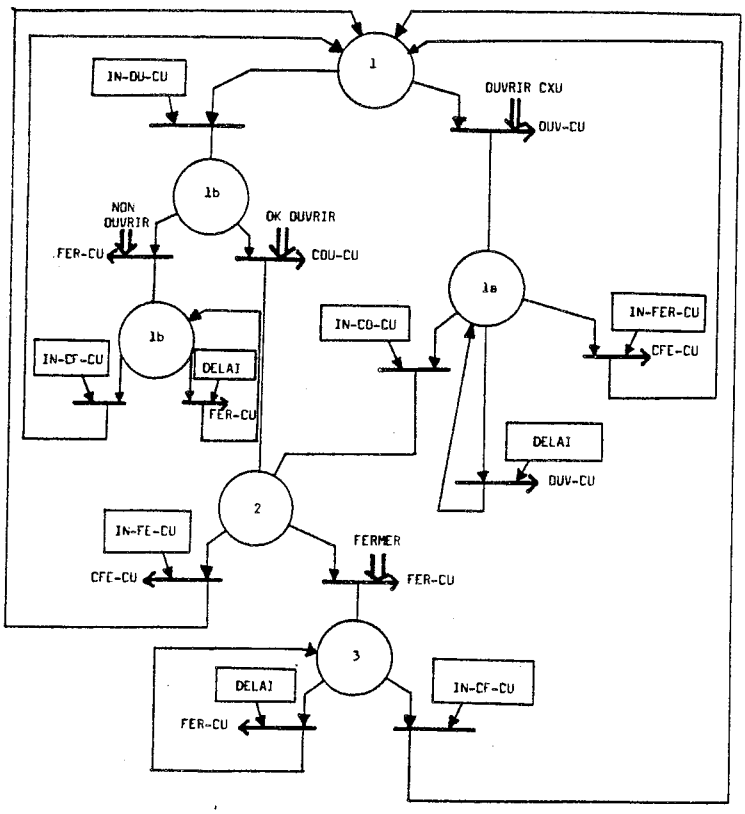


figure 1.5 : Diagramme d'état de l'acteur

(1) = mode non connecté,

(1a) = (1b) = en cours d'établissement,

(2) = mode connecté,

(3) = En attente de confirmation de fermeture de connexion,

1.3.4.4.3/ Transfert sur connexion U

Le transfert sur connexion U permet l'acheminement d'unités de données utilisateurs avec contrôle de flux et contrôle d'erreur.

Contrôle de flux

Dans Chorus l'acteur est activé par un message. Il éxécute alors une étape de traitement qui peut le conduire à émettre des messages.
Le contrôle de flux que l'on définit va utiliser cette propriété. On utilise un "message de contrôle de flux" (noté MC_IN_OK_CU) pour déclencher une activation de l'acteur uniquement si le transport lui permet d'émettre des messages de données sur la connexion U.

Ce message de contrôle de flux est appelé une "autorisation à émettre" et est utilisé par le service de transport pour réguler l'acteur utilisateur et réciproquement. Ce message est repéré d'après nos conventions par le code "MC_IN_OK_CU"

Remarque 1:

L'interprétation à un pour un (un message de contrôle de flux permet l'envoi d'un message de données) du message de contrôle de flux n'est pas obligatoire. Cette technique du "message de contrôle de flux" n'empêche pas de réaliser des implantations où il est possible d'envoyer plusieurs messages en même temps. Dans ce cas, l'acteur utilisateur a une anticipation fixe de N messages à chaque OK du service de transport.

Contrôle d'erreur

Les échanges de messages entre l'acteur utilisateur et le service de transport se font en datagramme sans diagnostic. De fait, cela signifie que l'acteur utilisateur n'a aucune confirmation explicite par le service de transport de la remise de son message.

Mais, lorsque l'acteur utilisateur recoit un message de contrôle de flux, cela signifie que le service de transport sur connexion fera tout pour remettre, dans la file de l'acteur utilisateur final, la (ou les) prochaine(s) unité(s) de données qui va(vont) lui être confiée(s).

Remarque 2:

Le contrôle de flux crée un dialogue de type appel-réponse. Si l'information de contrôle de flux n'est pas reçue au bout d'un temps fini, l'acteur utilisateur (ou le service de transport) peuvent reémettre le dernier message envoyé (données ou OK).

Transfert de données

On suppose la connexion déja ouverte. Pour clarifier on ne donne un transfert de données que de A vers B. La connexion U est bidirectionnelle, les échanges de données peuvent donc être simultanés dans les deux sens.

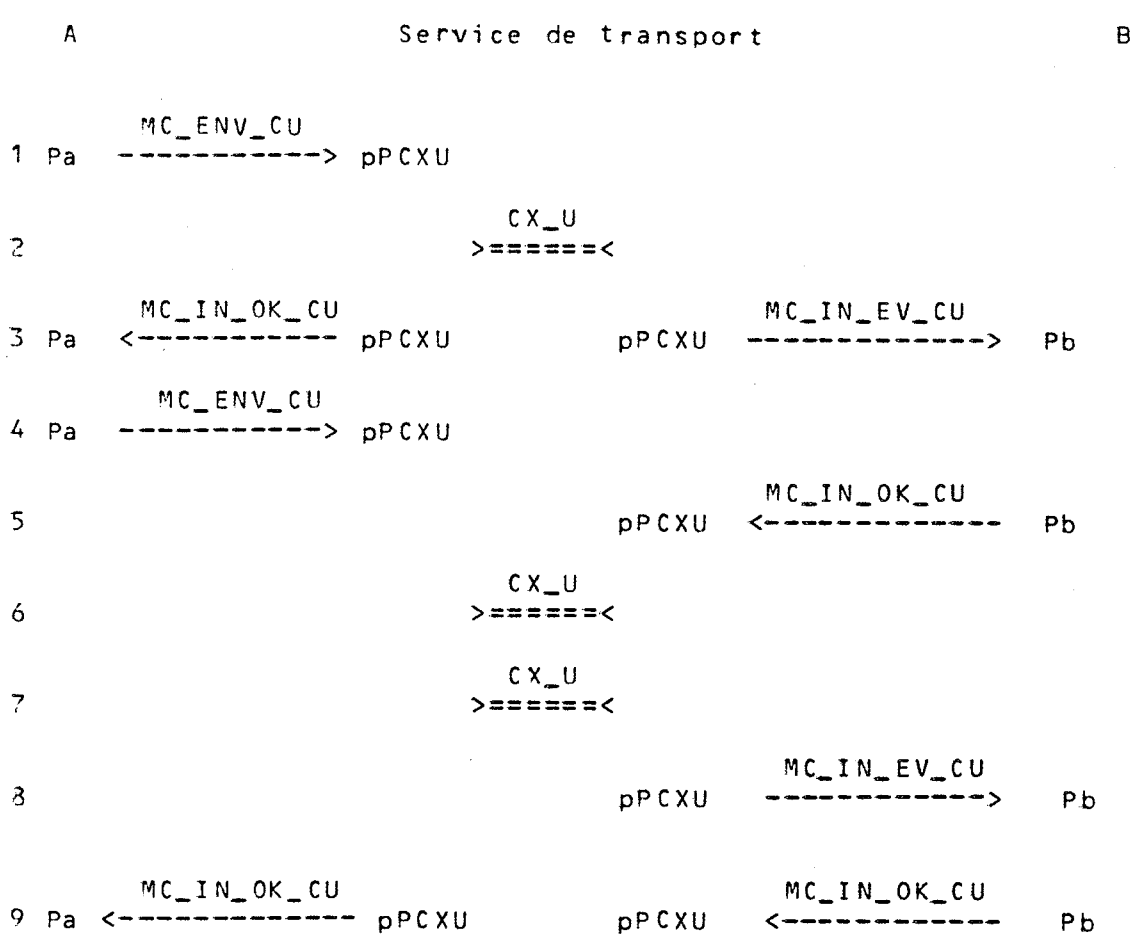


figure 1.6 : Transferts sur connexion U

Description des étapes

1) L'acteur A envoie une unité de données sur la connexion U. Il avait reçu auparavent soit un message de confirmation d'ouverture, soit un message d'autorisation à émettre.

2) Le service de transport transmet l'unité de données vers B.

3) Le service de transport remet l'unité de données à B. (Le service de transport peut se déclarer prêt (en anticipant localement) à recevoir une nouvelle unité de données venant de A sur la connexion U).

4) A envoie une nouvelle unité de données sur la connexion. Le service de transport conserve l'unité de données jusqu'à ce qu'il puisse la remettre à B.

5) L'acteur B se déclare prêt à accepter un nouveau message sur la connexion U.

6) et 7) Le service de transport transmet les indications de contrôle de flux de B et le message de A.

8) Le service de transport délivre l'unité de données de A à B.

9) Le service de transport se déclare prêt à accepter un nouveau message venant de A.
B se déclare prêt à accepter un nouveau message venant du service de transport.

Remarques

Le contrôle de flux est réalisé entre le service de transport et l'acteur utilisateur. Le service de transport peut anticiper sur la remise d'une unité de données et accepter, par un OK, des unités de données de l'émetteur s'il lui est possible de les stocker.

Diagramme d'état du mode connecté

Un acteur qui est en connexion U peut se trouver dans plusieurs états distincts suivant qu'il a reçu ou non l'autorisation d'émettre et suivant qu'il a envoyé ou non l'autorisation d'émettre.

(2 E+R) identifie l'état où l'acteur (ou le service) peut émettre et recevoir des données sur la connexion U.

(2 E) identifie l'état où l'acteur (ou le service) peut émettre et

n'est pas prêt à recevoir des données sur la connexion U.

(2 R) identifie l'état où l'acteur (ou le service) ne peut pas émettre mais est prêt à recevoir des données sur la connexion U.

(2 E+R) identifie l'état où l'acteur (ou le service) ne peut ni émettre ni recevoir des données.E sur la connexion U.

Dans la figure qui suit, les codes de message ont été amputés du MC (MESSAGE CHORUS) pour alléger la notation graphique. Les deux états (2 E+R) représentés sont équivalents.

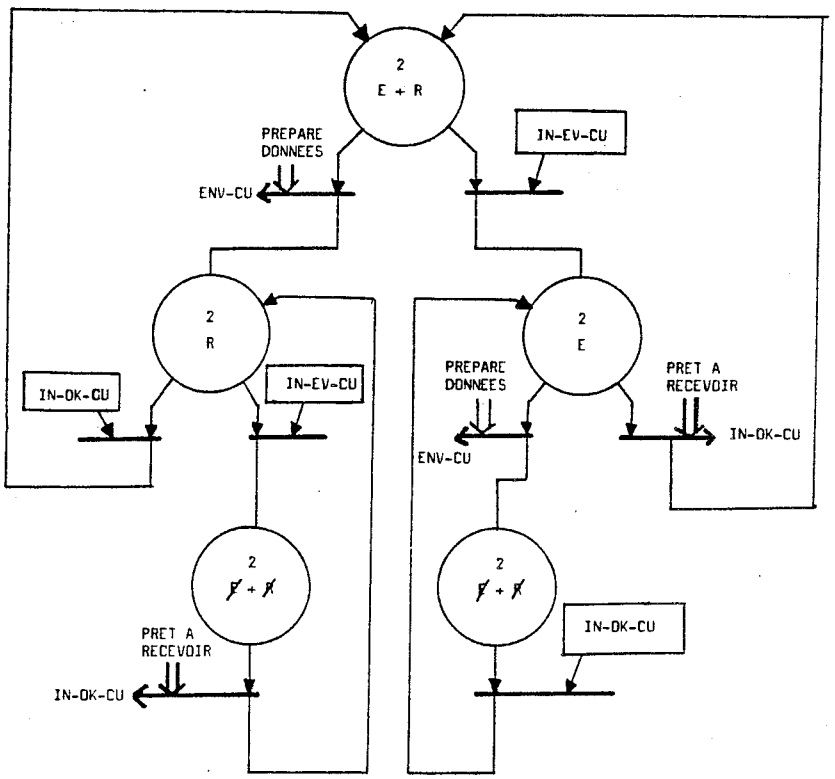


figure 1.7 : Diagramme d'état du mode connecté

1.3.4.4.4/ Commentaires

Les messages de l'un des sens ne peuvent pas interagir avec les messages de l'autre. Ce qui permet de séparer les comportements de l'utilisateur suivant qu'il émet ou qu'il reçoit sur la connexion U.

Cela s'avère intéressant dans le cas de Chorus où, seul, le message peut déclencher un traitement. On peut alors décrire des étapes de traitement totalement indépendantes et non perturbées par le fait qu'il s'agisse d'une réception de données à traiter ou d'une autorisation à émettre.

L'acteur est peu perturbé par l'utilisation du protocole de connexion U dans la mesure où il n'a pas beaucoup d'information à gérer et qu'il est piloté par le service de transport au moyen des messages de contrôle de flux.

1.3.4.5/ Les messages du mode connecté

Nous rappelons pour mémoire le format des unités de données utilisateurs remises au service de transport.

Pr,Pe,SERVICE,/TEXT/

Dans le transport en mode connecté, les paramètres de SERVICE et le texte vont dépendre du type du message du protocole de connexion U. Les portes vont être celles utilisées pour les échanges entre le service de transport et l'acteur utilisateur. Le format du message sur connexion est donc le suivant:

Pr,Pe,{Type,Pabo,Pdist,opt},/TEXT/

{...} = Paramètres du service en mode connecté,

Type : MC_OUV_CU

Pabo : porte du demandeur pour la connexion U,

Pdist : porte vers laquelle est demandée la connexion U,

opt : Paramètres optionnels (taille des messages acceptés, référence utilisateur du message, etc..).

Nous allons donner la structure de chacun des messages du mode connecté en reprenant les conventions sur leur type définies plus haut.

Ouverture

Pour le message MC_OUV_CU:

PTCXU,Pa,{MC_OUV_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_COU_CU:

PTCXU,Pa,{MC_COU_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_IN_OU_CU:

Pa,pPCXU,{MC_IN_OU_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_IN_CO_CU:

Pa,pPCXU,{MC_IN_CO_CU,Pa,Pdist,opt},/NIL/

Fermeture

Pour le message MC_FER_CU:

pPCXU,Pa,{MC_FER_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_CFE_CU:

pPCXU,Pa,{MC_CFE_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_IN_FE_CU:

Pa,pPCXU,{MC_IN_FE_CU,Pa,Pdist,opt},/NIL/

Pour le message MC_IN_CF_CU:

Pa,pPCXU,{MC_IN_CF_CU,Pa,Pdist,opt},/NIL/

Transfert

Pour le message MC_ENV_CU:

pPCXU,Pa,{MC_ENV_CU,Pa,Pdist,opt},/Données utilisateur/

Pour le message MC_IN_EV_CU:

Pa,pPCXU,{MC_IN_EV_CU,Pa,Pdist,opt},
/Données utilisateur distant/

Pour le message MC_IN_OK_CU:

Pa,pPCXU,{MC_IN_OK_CU,Pa,Pdist,opt},/NIL/

ou pPCXU,Pa,{MC_IN_OK_CU,Pa,Pdist,opt},/NIL/

1.3.5/ Fonctions du transport en mode connecté

1.3.5.1/ Etablissement des connexions U

Cette phase a pour but de créer le lien entre deux acteurs utilisateurs. Les opérations principales sont:

- déterminer la localisation des portes à connecter; le routage du mode non connecté est utilisé pour déterminer la localisation de la porte réceptrice de l'autre acteur utilisateur et donc de l'autre homologue de transport,
- créer un contexte propre à la connexion U; ce contexte contient le nom des portes en connexion U, les paramètres de gestion de la connexion U [crédit, référence des messages sur la connexion U, etc..]

1.3.5.2/ Terminaison des connexions U

Le service de transport éxécute toutes les opérations nécessaires à la terminaison de la connexion U. Il permet aux acteurs utilisateurs de décider de la fermeture et s'arrange pour que celui qui n'a pas l'initiative de la fermeture soit prévenu.

Il décide de la fermeture, si la qualité du service de connexion U ne peut pas être maintenue.

1.3.5.3/ Transfert de données

Le transfert de données est la phase essentielle, puisqu'elle permet le transfert des données entre les utilisateurs connectés avec la qualité de service qui est recherchée. C'est celle qui nécessite le plus de fonctions particulières telles que:

- le routage des messages utilisateurs; si les deux portes sont sur le même site, il lui suffit de reémettre le message utilisateur vers la porte réceptrice locale.
- le maintien en séquence,
- la fragmentation/réassemblage,
- le contrôle de flux,
- la détection et le reprise d'erreurs,
- l'identification des connexions Utilisateurs en leur créant un contexte particulier.

Anticipation

Il n'y a pas d'anticipation véritable de la part des acteurs utilisateurs. Suivant les conventions d'ouverture, ils peuvent être autorisés à émettre plusieurs unités de données pour un seul message de contrôle de flux. Cette convention est fixe. Nous avons pris ce parti dans le seul but de ne pas laisser aux acteurs utilisateurs la nécessité d'interpréter et de gérer un crédit qui pourrait être indiqué dans le message de contrôle de flux.
Il est possible de revenir par la suite sur cette décision; dans ce cas le message de contrôle de flux indiquera le nombre de messages que l'acteur est autorisé à émettre pour l'étape de traitement qui commence et uniquement celle-ci.

1.3.6/ Gestion des connexions U

La gestion des connexions U est entièrement faite par le service de transport.

A l'ouverture, il lui faut vérifier:
- l'unicité de la connexion U demandée,
- que la porte avec laquelle l'ouverture est demandée peut accepter

une connexion (on peut imaginer des portes protégées contre des demandes d'ouvertures venant d'acteurs utilisateurs).
Il peut éventuellement refuser une connexion U si les ressources dont il dispose pour gérer les connexions U sont saturées.

En cours de transfert de données, il lui faut vérifier:
- que la porte utilisateur émettrice est correcte (c'est-à-dire qu'elle émet vers la porte privée de connexion qui lui est attribuée),
- que les acteurs utilisateurs respectent le protocole de connexion U.

A la fermeture, il lui faut vérifier:
- que la fermeture se déroule normalement,
- libérer les ressources occupées pour la gestion de la connexion U.

1.3.7/ Conclusions sur le mode connecté

Les techniques employées sur la connexion U, le service offert (porte privée, contrôle de flux et d'erreur, séquencement), en font un outil intéressant toutes les fois que des échanges à fort courant de données entre des machines différentes ont lieu.
Outre la gestion des transmissions, la connexion U peut offrir un mécanisme de protection sûr et simple pour l'acteur parce que:

- on peut être amené à construire des applications pour lesquelles, il est nécessaire d'établir une connexion U avant de communiquer,
- la porte privée de connexion, connue seulement du transport et de l'utilisateur, garantit contre les intrusions dans le protocole,
- on peut inclure dans le service de transport des mécanismes de protection (codage, clés, etc..) garantissant la sécurité des informations échangées sur la connexion U.

1.4/ Conclusions sur le service de transport

Le transport en mode non connecté est un service qu'il est facile de rendre de façon implicite dans Chorus. Il satisfait tous les types d'échanges du genre "appel/réponse". Les acteurs utilisateurs n'ont aucune difficulté ni aucune contrainte de comportement dans ce type de transport.

Sa fiabilité est directement fonction de la qualité des transmissions locales et distantes et cela peut poser des problèmes sur certaines machines ou certains réseaux (leur choix doit donc être judicieux pour limiter les fonctions à réaliser).

Le transport en mode connecté offre une meilleure garantie de qualité du transport, mais sa mise en oeuvre est plus délicate puisque l'acteur utilisateur doit être programmé pour respecter un contrôle de flux qui lui est imposé par des messages.

2/ Réalisation des services de transport de Chorus

Dans cette partie nous décrivons la réalisation du service de transport qui met en oeuvre le noyau et des acteurs de transport. Nous justifions d'abord la décomposition des fonctions que nous allons décrire.

Le service de transport est activé lorsqu'un acteur utilisateur termine son étape de traitement. Le noyau entre alors en action. Si la porte réceptrice est locale, le noyau fait le transfert du message utilisateur sur la file d'attente correspondante. Dans ce cas le transport correspond à un transport local.

Si la porte réceptrice n'est pas locale le noyau transfère le message sur une porte par défaut: la porte "TRANSPORTER A DISTANCE" [S/Sd/Sp/PTRANSADIST] Cette porte appartient au "service de transport

distant" qui est réalisé par la collaboration d'acteurs de transport des différents sites du système réparti. Le service de transport distant achemine le message utilisateur vers le site qui abrite la porte réceptrice.
Sur le site récepteur le message est émis vers la porte réceptrice finale par l'acteur de transport distant local.
Dans ce cas, le transport a été réalisé par la coopération du transport local de chacun des sites concernés par l'échange et du transport distant.

On voit donc apparaître, pour le transport des messages, deux mécanismes de base pour le transport: le transport local et le transport distant.

Cette décomposition reste transparente à l'utilisateur.

2.1/ Le transport local

Le transport local est identique au service de commutation locale, avec une fonction supplémentaire qui est le routage des unités de données de l'utilisateur. Ce routage correspond à la recherche de la localisation de la porte réceptrice de l'unité de données de l'acteur utilisateur.

Ce service est implicite à la fin de l'éxécution de l'acteur utilisateur.

2.1.1/ Entité de transport local

Le transport local est réalisé par le noyau qui dispose des tables des portes locales et a donc accès à tous les mécanismes de communication locaux.

2.1.2/ Point d'accès

Le point d'accès transport local est la porte "TRANSPORTER MESSAGE" du noyau.

2.1.3/ Protocole de transport local

Toutes les opérations sont centralisées dans le noyau. Il n'y a pas de protocole particulier lié à l'acheminement des messages localement.

2.1.4/ Fonctions du transport local

Le noyau assure le séquencement des unités de données qui restent locales, c'est-à-dire qu'il dépose les messages en respectant leur ordre d'émission par l'acteur utilisateur local.

Le noyau effectue le routage en cherchant la porte réceptrice dans la table des portes locales et en traitant le défaut de porte réceptrice comme une demande de dépôt sur la porte par défaut de transport distant.

Le noyau réalise un transport local fiable dans la mesure où il est conçu pour cela (mécanisme de commutation locale et de transfert fiable)

2.1.5/ Conclusions sur le transport local

Le transport local est la fonction implicite du service de transport. Il est réalisé par le noyau. C'est un service de datagramme fiable.

2.2/ Le transport distant

Le service de transport distant permet l'acheminement d'unités de données du transport local utilisateur entre deux sites de Chorus.

2.2.1/ Les entités de transport distant

Le transport distant est une fonction répartie. Sur chaque site se trouve un *acteur de transport*. Les acteurs de transport coopèrent pour rendre le service de transport distant.

2.2.2/ Le service de transport distant

Le service de transport distant utilise le service de commutation distante qui est réalisé en datagramme sans diagnostic.

Un réseau à circuit virtuel, offre une bonne qualité de service grâce au contrôle de flux, d'erreur et il conserve l'ordre des messages.

Par contre, dans le cas d'un réseau à datagramme les erreurs ne sont pas détectées et sont statistiquement non négligeables.

Il est nécessaire d'offrir un service de transport d'une qualité indépendante du service réseau et surtout qui soit comparable avec la qualité du transport local. Dans certains cas, les acteurs de transport doivent donc suivre des protocoles de "transport distant" qui leur permettent d'améliorer les échanges distants.

2.2.3/ Accès au transport distant

Le transport distant est un sous service du niveau transport. Il n'est accessible que par les entités du niveau transport, c'est-à-dire les acteurs de transport et le noyau.

La porte "TRANSPORTER A DISTANCE" est la porte par défaut sur le site. Les acteurs de transport distant communiquent entre eux par la porte "CONNEXION DISTANTE" [S/Sd/Sp/PCONDIST] sur laquelle s'échangent les unités de données de transport distant.

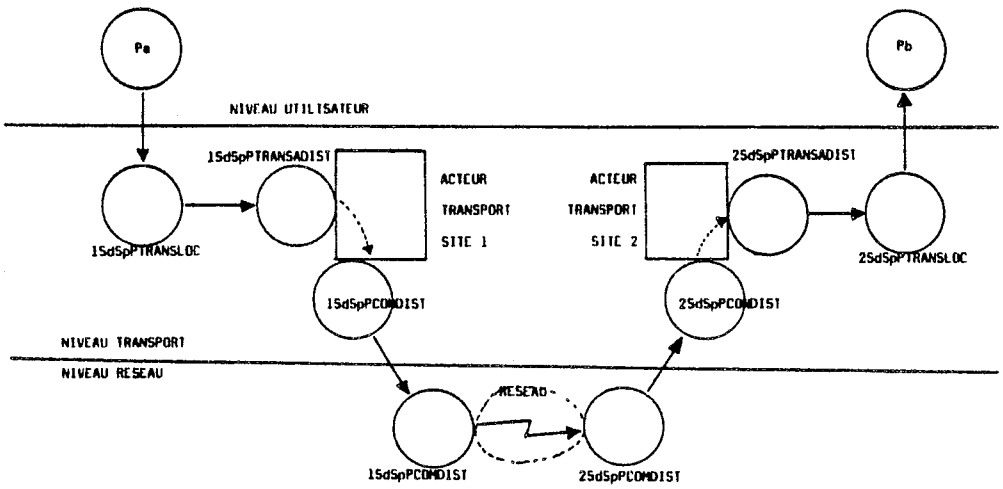


figure 2.1 : Transport distant

2.2.3.1/ Unités de données du transport distant

On identifie ainsi les unités de données échangées entre les acteurs de transport. Elles sont composées comme suit:

PTDr,PTDe,SerTD,/Pe,Pr,SERVICE,TEXT/

PTDr = Porte de transport distant réceptrice,

PTDe = Porte de transport distant émettrice,

SerTD = Paramètres du transport distant

Type du message (dépend du protocole),

Référence de lettre,

Référence de fragment,

etc..

/../ = Unité de données du transport local.

2.2.4/ Les protocoles de transport distant

Les protocoles de transport distant permettent d'améliorer la qualité des acheminements entre sites à travers le réseau.

On distingue deux classes de protocoles: les protocoles sans connexion et les protocoles avec connexion distante.

Les protocoles sans connexion ressemblent aux protocoles que l'on peut trouver pour les procédures de transmission. Ils permettent surtout de réaliser du contrôle d'erreur entre deux sites.

Les protocoles avec connexion permettent de réaliser du contrôle d'erreur, du contrôle de flux et du séquencement.

2.2.4.1/ Protocole sans connexion

Les acteurs de transport s'échangent des "unités de données de transport distant" numérotées et confirment leur réception par un acquittement positif.

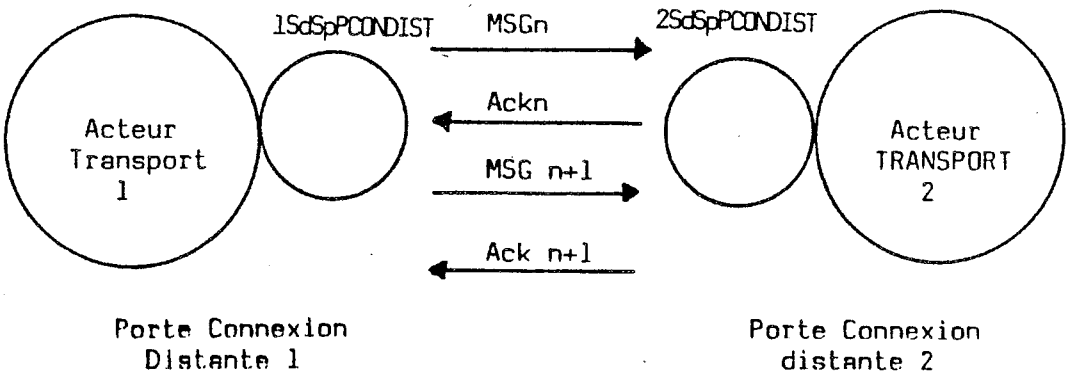


figure 2.2 : Protocole de transport distant sans connexion

Si au bout d'un temps fini un acteur de transport émetteur n'a pas recu d'acquittement, il reémet l'unité de données de service de transport distant.

L'acteur de transport récepteur détruit les doublons éventuels.

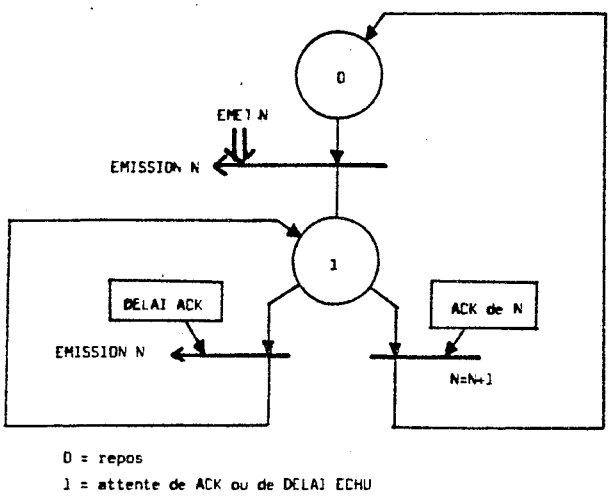


figure 2.3 : Diagramme mode non connecté émetteur

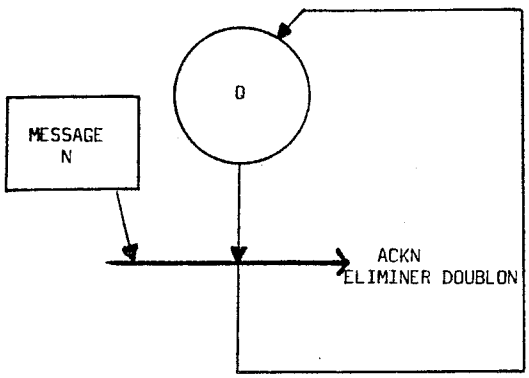


figure 2.4 : Diagramme mode non connecté récepteur

Commentaires

Ce type de protocole permet d'abord de résister aux pertes de messages.

Il permet le séquencement par le site récepteur des messages provenant d'une source donnée. En effet, si les messages sont numérotés à leur création, ils peuvent être ordonnés par l'acteur de transport récepteur qui respecte à la remise l'ordre de création sur le site émetteur.

2.2.4.2/ Protocole avec connexion de transport

La connexion de transport distant (connexion TD) correspond à un protocole qui permet l'établissement d'une voie logique de communication entre deux acteurs de transport, avec des phases d'établissement, de transfert de données, de fermeture. La connexion offre le contrôle de flux et le contrôle d'erreur.

Il convient de rappeler que nous étudions d'abord un modèle qui nous permette de définir les différentes stratégies et options à mettre en oeuvre pour réaliser le service de communication de Chorus. Dans cette démarche, l'étude d'un protocole de transport distant ne présente pas, à nos yeux une originalité suffisante pour être longuement développée ici. Dans les réseaux connus à ce jour, de nombreux protocoles (implantés dans les stations de transport des réseaux [ELI 75]) existent pour ce genre d'application.

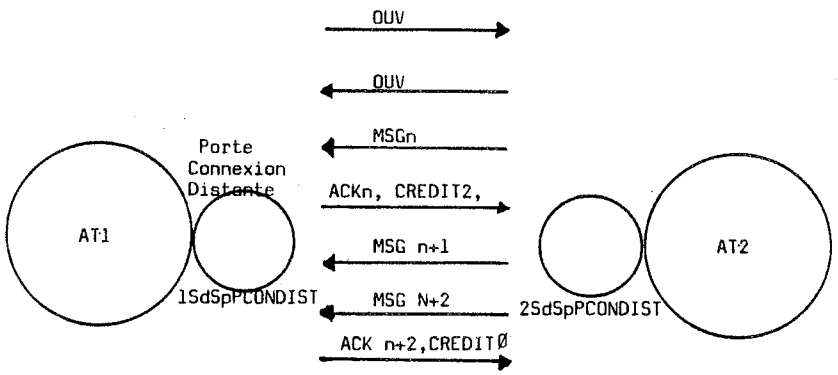


figure 2.5 : Exemple simplifié d'échange sur connexion distante

ACK n = Acquittement du message n,

CREDIT x= droit démettre x message.

Nous désignons dans la suite de cette étude les connexions de transport distant "Connexions TD".

2.2.5/ Fonctions de l'acteur de transport

L'acteur de transport gère le transport distant des messages qui lui sont remis par le noyau. Il établit, si nécessaire, des connexions de transport distant avec ses homologues des autres sites.

Il dépose (et recoit) les unités de données du service de transport distant sur (et de) la porte COMMUTATION DISTANTE de l'acteur de commutation distante.

2.2.5.1/ Le routage distant

Pour router une unité de données de transport local, l'acteur de transport doit localiser le site qui abrite la porte réceptrice. Il peut utiliser la technique suivante:

1) il communique soit en diffusion soit en point à point (cela dépend du service réseau disponible) avec les autres acteurs de transport distant pour leur demander si la porte réceptrice recherchée est sur leur site. Il recoit en retour une réponse positive ou négative.

   Cette technique est particulièrement simple à mettre en oeuvre dans le cas où le service réseau permet la diffusion (grâce à un réseau à diffusion, par exemple).

   Le protocole peut être le suivant:

   L'acteur de transport distant A qui recherche la porte TOTO [TOT XX], demande à l'acteur commutation distante d'émettre sur l'adresse de diffusion, le message suivant:

   "Avez-vous sur votre site la porte TOTO ?"

   Chaque acteur de transport distant qui recoit le message s'enquiert sur son site de la présence de la porte et un seul

répond; l'acteur B qui est celui du site sur lequel se trouve la porte TOTO, avec le message suivant:

"La porte TOTO est sur le site N !!"

L'acteur A recoit la réponse au bout d'un temps fini sinon, il peut: soit abandonner momentanément la recherche, soit retenter sa chance. Il peut conserver un certain temps (cela dépend des implantations) la localisation de la porte réceptrice.

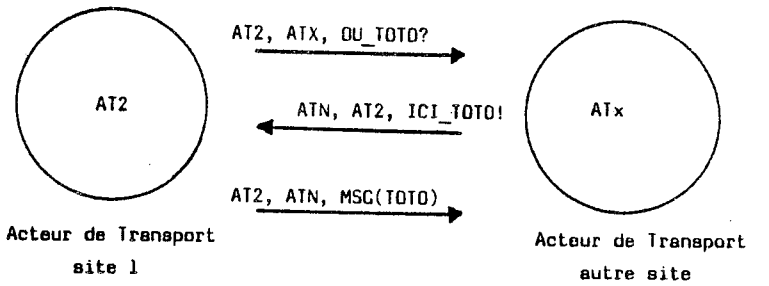


figure 2.6 : Localisation de porte par l'acteur de transport

Dans Chorus le mécanisme de désignation fournit la possibilité de simplifier les problèmes de routage dans le cas des portes sédentaires. En effet, les portes sédentaires demeurent sur leur site de création qui est indiqué dans leur nom. L'acteur de transport peut donc, par interprétation du nom de porte, déterminer le routage vers le site qui possède une porte réceptrice recherchée.

Il peut même "déduire" le nom de la porte de "TRANSPORT DISTANT DU SITE N" par concaténation d'une estampille standard (Porte Transport entre Sites) et du site de la porte sédentaire réceptrice.

D'autres techniques sont envisageables telles que le routage par inondation. Si le service de commutation offre la possibilité de diffuser vers N adresses ou vers une adresse de diffusion, l'acteur de transport qui veut émettre un message, sans localiser la porte

réceptrice, demande sa diffusion à tous les autres acteurs de transport qui font le tri des messages qui sont destinés à des portes de leur site. Cette méthode peut surcharger les acteurs de commutation.

2.2.5.2/ Fragmentation/réassemblage

Le service de transport distant fait de la fragmentation/réassemblage sur les unités de données de transport local qui ne sont pas compatibles avec la taille des paquets du service de commutation. La technique est supposée suffisamment connue pour ne pas être développée ici [CER 74, BOG 80, SHO 79].

2.2.5.3/ Le séquencement

Il dépend du protocole de transport distant utilisé. Il ne peut être fait qu'entre des messages issus du même site.

Nous n'envisageons pas, pour l'instant, d'utiliser de mécanismes d'ordre total sur le système réparti. Plusieurs techniques (estampilles, horloge virtuelle) sont implantables dans l'acteur de transport pour certaines applications qui le nécessiteraient.

2.2.5.4/ Conclusions sur le transport distant

Le mécanisme de transport distant est réalisé par un acteur de transport distant qui peut établir des connexions de transport distant avec les autres acteurs de transport distant des autres sites de Chorus.

Les acteurs de transport distant échangent des unités de données du service de transport distant qui permettent de transporter les unités de données du service de transport local.

Le service de transport distant est une fonction interne au service de transport du niveau transport de Chorus. Les acteurs utilisateurs n'ont pas la visibilité de ce transport distant. Grâce aux fonctions qui y sont incluses, ce service est équivalent au service de transport local c'est-à-dire à un datagramme fiable sans diagnostic. Cela permet d'offrir, aux acteurs utilisateurs, le service de transport en mode datagramme sans diagnostic suffisamment fiable que nous estimions suffisant à la fin du chapitre 3.

3/ Le transport distant et le mode connecté

Ce dernier paragraphe montre l'utilisation de l'acteur de transport pour réaliser le service de transport en mode connecté.

Nous avons vu que le mode connecté est un service explicitement demandé par les acteurs utilisateurs sur des portes du service de transport. Ces portes appartiennent à l'acteur de transport qui a été choisi parce qu'il réalise déja les fonctions de transport distant qui sont utilisées dans le mode connecté.

En effet, l'acteur de transport établit des connexions TD avec les autres acteurs de transport pour bénéficier du contrôle de flux et d'erreur entre les sites. S'il met en correspondance les connexions TD et les connexions U dans une même entité, il peut réaliser un contrôle de flux en cascade entre deux portes utilisateurs situées sur des sites différents.

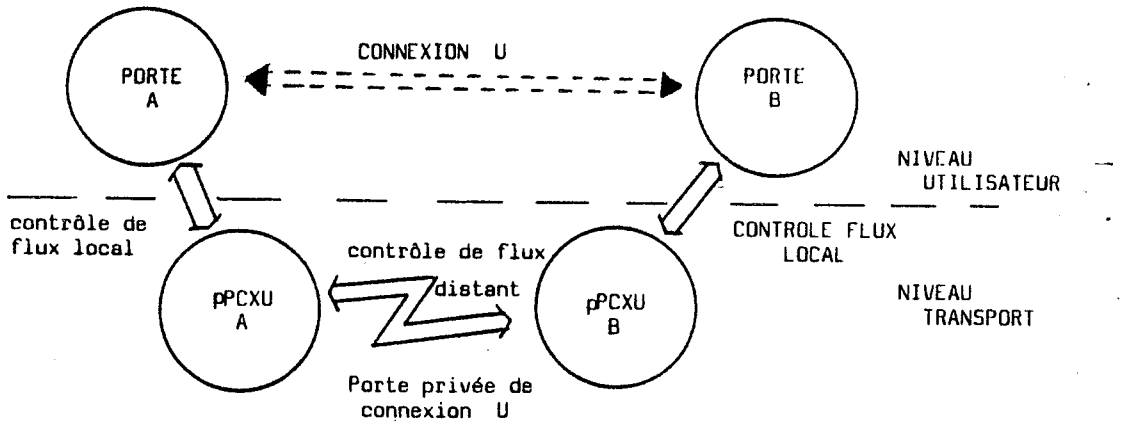


figure 3.1 : Contrôle de flux entre portes utilisateurs

3.1/ Fonctions de l'acteur de transport

L'acteur de transport gère toutes les connexions U de son site. Il réalise le contrôle de flux local avec les acteurs en connexion U de son site. Il établit, ferme, transfère les données des connexions U. Les fonctions de l'acteur de transport sont identiques à celles données dans la description du service de transport en mode connecté. Nous n'ajouterons qu'une particularité qui est propre à la gestion des connexions U. Il s'agit du multiplexage des connexions U sur les connexions TD.

Correspondance Connex. U <==> Connex. TD

L'acteur de transport utilise des connexions TD pour le transport entre site. Il n'est pas nécessaire d'établir une connexion TD par connexion U. L'acteur de transport doit donc établir des correspondances entre connexions TD et connexions U.

Multiplexage NCon. U => 1 Con. TD

Dans le multiplexage de N connexions U sur une connexion TD, les deux acteurs de transport utilisent la même connexion TD pour acheminer les messages de plusieurs connexions U simultanées.

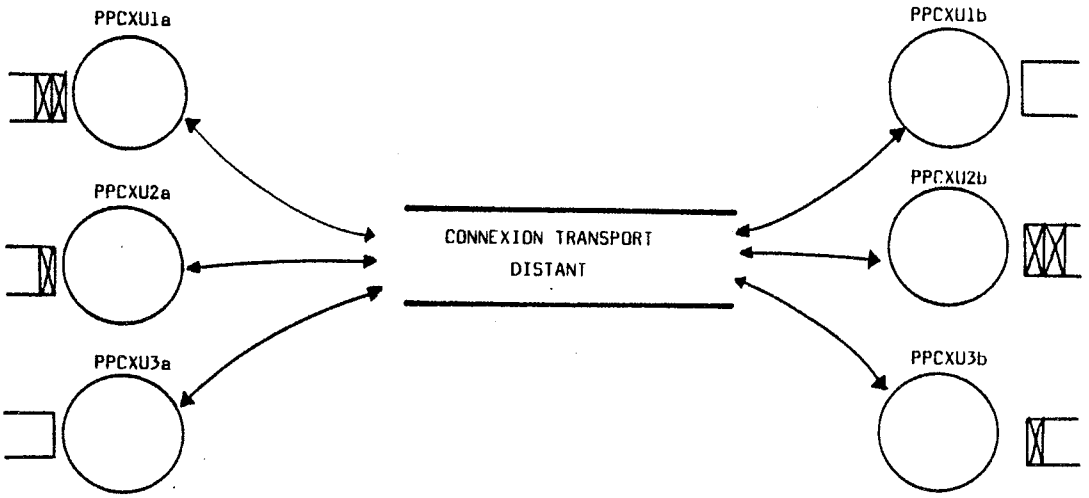


figure 3.2 : Multiplexage N U -> 1 TD

4/ Conclusions sur le niveau transport

Avec la description du niveau transport, nous terminons la définition et les spécifications du modèle (et du service) de communication de Chorus que nous nous étions proposé d'établir. Le niveau transport, tel qu'il a été défini correspond aux préoccupations qui étaient les notres aux début de cette étude à savoir:

- définir un service de communication qui permet l'indépendance des traitements et du transport,
- définir un service qui offre le choix de différentes stratégies de communication de façon transparente aux utilisateurs,
- construire des mécanismes de communication adaptés aux contraintes des machines et des applications visées,

- construire un service de transport qui tienne compte des options prises au niveau de l'organisation des traitements.

Nous n'avons pas développé les facilités offertes aux acteurs utilisateurs pour accéder au service de transport. Les implantations actuelles utilisent des procédures d'interfaces qui préparent les unités de données pour les portes du service de transport. Des exemples, pour une implantation donnée, sont fournis dans les annexes.

CHAPITRE 6: EVALUATION DE L'EXPERIENCE ACQUISE

INTRODUCTION

Cette étude d'un service de communication pour l'architecture Chorus n'est pas achevée aujourd'hui, mais le travail présenté ici nous a permis de nous confronter avec la façon relativement nouvelle d'étudier les mécanismes de communication introduits par Chorus.

Il faut distinguer deux domaines dans lesquels notre étude a posé des problèmes nouveaux pour nous.

- L'analyse et la spécification de l'application répartie de transport pour la traduire en termes d'acteurs.

- La mise au point des programmes de communication en univers réparti nous a apporté des indications sur:
  - les problèmes de mise au point des protocole de communications,
  - les difficultés de synchronisation des acteurs entre eux,
  - les problèmes dus à l'asynchronisme des messages, au temps de réponse et au délai de transit à travers le système.

Les réflexions que nous sommes conduits à faire se divisent donc en deux parties:

- les remarques sur les problèmes ou les avantages de l'analyse d'une application avec Chorus,
- les remarques sur les problèmes et les facilités rencontrés au cours de l'expérimentation sur Chorus.

1/ Expérience tirée de l'analyse

1.1/ Les problèmes d'analyse

Les problèmes d'analyse portaient sur la définition du service de communication à installera sur Chorus et sur la façon de le décrire avec les outils de Chorus.

1.1.1/ Spécification du service de communication

Analyse de l'existant

Notre expérience antérieure, nous a mis au fait des problèmes que peut poser l'utilisation d'un réseau, tant du point de vue de la mise en oeuvre que des caractéristiques de fonctionnement (pannes, pertes de messages, délai de transit non borné). Nous avons aussi une bonne connaissance des mécanismes de transfert de données à l'intérieur d'une machine ainsi que de la fiabilité que l'on peut en attendre.

Notre objectif dans Chorus est de faciliter les traitements répartis, sans masquer la répartition au concepteur. Cela nous amène à construire un service de communication qui facilite l'accès aux mécanisme d'échange, mais qui n'élimine pas les contraintes qui y sont liées (asynchronisme des arrivées de messages, pertes de messages, problèmes de protection, etc.).

Le message est le support le plus approprié pour l'échange d'information.

Identification du problème

En analysant des systèmes existants, nous avons vu que le service de communication est une fonction du système parmi les autres et qu'à ce titre, il peut être décrit avec les même outils que les traitements des applications, et qu'il utiliser des messages pour son propre fonctionnement.

Cette constation nous a conduit à considérer deux axes principaux de travail; définir les mécanismes de base (très près du matériel) qui permettent de faire le lien entre les phénomènes électriques et le logiciel qui traite les évènements physiques (transformation en messages); définir les mécanismes de base qui permettent de traiter les messages.

Le premier axe représente le travail d'adaptation spécifique du service de communication au support choisi.
Le deuxième axe représente la définition et la réalisation d'une organisation des traitements compatible avec une communication par messages.

Le parti raisonnable consistait donc à élaborer d'une part un noyau minimum qui contienne tout ce qui est dépendant de la machine, et d'autre part une organisation des traitements qui s'appuie sur le noyau et le service de communication.

L'organisation du service de communication est donc basée sur la définition du noyau et des acteurs de communications.

Il restait cependant à organiser la répartition et les fonctions de chaque acteur. C'est à ce stade de l'étude que le choix et la définition du modèle s'imposaient. Nous ne reviendrons pas sur les raisons du choix.

Le problème qui se posait alors était de faire un tri judicieux parmi toutes les formules qui s'offraient à nous pour organiser le service de transport.

Nous nous ne voulions pas faire d'hypothèses sur le réseau. Cela signifie dire que le service réseau devait être assez général et assez souple pour s'adapter à différents choix de systèmes de communication. Il nous fallait aussi aller dans le sens d'une simplication des mécanismes de base, dans un souci de performance et de simplicité de programmation.

Voilà pourquoi le principe d'un transport en datagramme a été envisagé. Ce type de service est très courant sur les réseaux locaux, qui sont quand même susceptibles d'être les plus utilisés pour Chorus. Cette orientation vers le mode datagramme est renforcée par le fait que les transmissions sur une machine peuvent être considérées comme très fiables.

1.2/ Evaluation des outils d'analyse

1.2.1/ Les apports du modèle ISO

Les apports du modèle ISO peuvent se résumer ainsi:

- une analyse détaillée des éléments qui entrent en jeu dans les communications,
- découpage du service de communication en fonctions élémentaires,
- définition des interfaces offrant des points de visibilité facilitant l'insertion de protocoles adaptés,

1.2.2/ Les apports de Chorus

Les apports de Chorus concernent surtout l'organisation des traitements associés aux communications. Cette organisation est d'autant plus simple qu'elle applique les règles de base de Chorus dont les atouts les plus importants sont:

- la définition des traitements atomiques du point de vue des communications,
- la séquentialité de l'acteur,
- la définition d'un point unique de visibilité entre les communications et les traitements; la fin d'une étape de traitement,
- une interface unique de communication: la porte.

1.2.2.1/ Ident. des fonctions de communication

Le modèle de communication nous permet, à l'analyse, de séparer les fonctions de communication. L'organisation Chorus nous permet de les considérer comme des étapes de traitement que l'on pourra accéder derrière des portes. L'activation par un message unique assure la précision sur la sélection d'une opération particulière, si elle a pu être isolée dans une étape de traitement.

1.2.2.2/ Simplification des messages

Il est possible d'ouvrir autant de portes que de fonctions particulières que l'on veut accéder. Cela simplifie la description des messages car le service accédé par une porte devient implicite (ce qui permet souvent de limiter le nombre de paramètres de service).

Les messages peuvent alors être très spécifiques et on peut avoir une assistance du système pour génération automatique de ces messages en fonction du service auquel on veut accéder (procédure d'interface).

1.2.2.3/ Visibilité des communications

Le service de communication a la visibilité des communications entre les portes. Cela lui permet de définir et d'utiliser les protocoles appropriés aux différents types de communications qu'il réalise (locales, distantes)

1.2.2.4/ Transport indépendant de la localisation

Il a été assez aisé de concevoir le noyau comme un commutateur local de message car le transport sur la machine réduisait son rôle à trouver la porte réceptrice dans la table des portes locales et à y déposer le message.

Le seul problème qui se posait au niveau du transport local concernait la récupération de l'erreur provoquée par le défaut de porte locale lors de la commutation par le noyau. La solution est venue en introduisant la porte par défaut qui représente toutes les portes qui ne sont pas présentent sur le site. Elle permet d'accéder au transport distant qui se charge de la localisation des portes réceptrices distantes et de l'acheminement des messages vers elles.

Il n'est pas concevable de laisser le noyau piloter l'accès au réseau, surtout s'il y en a plusieurs. On en est donc venu à admettre la nécessité d'un "acteur de commutation distante" qui est chargé du service réseau en collaboration avec les acteurs de commutation des autres sites (il peut sous-traiter l'accès à un réseau particulier à un acteur spécialisé du niveau réseau). La notion d'acteur de transport dérive de la station de transport des réseaux.

1.3/ Les problèmes de spécification d'acteur

1.3.1/ Les avantages du modèle Chorus

La spécification des acteurs de communication est facilitée par les services de base de l'architecture Chorus.

Utilisation des portes

Les portes peuvent être publiques (connues de tous avec une estampille standard) ou privées (connues de quelques acteurs). Les portes privées peuvent être utilisées comme nous l'avons vu dans le mode connecté pour protéger un accès à un service et améliorer la qualité du service rendu.

Les portes publiques peuvent être utilisées pour un accès général à un ensemble de services.

Accès aux traitements

La possibilité de définir des aiguillages, pris en compte dynamiquement par le système, permet de réaliser, à l'intérieur de l'acteur, la convergence, vers un même traitement, des messages arrivant sur différentes portes.

Exemple d'utilisation:

Plusieurs portes privées peuvent être aiguillées vers la même étape de traitement correspondant à un traitement standard (avec contexte défini par la porte et le message reçu).

On peut également faire varier le traitement associé à une porte en fonction des échanges pour lesquels elle est utilisée.

Exemple d'utilisation:

La porte ombilicale est utilisée pour traiter le message initial; elle est aiguillée sur l'étape de traitement d'initialisation de

l'acteur. Lorsque l'initialisation est terminée, cette porte peut être aiguillée vers une autre étape de traitement.

Gestion des files d'attente

Le système Chorus offre la gestion de la file d'attente associée à la porte puisqu'il met ou enlève les messages de cette file et maintient les messages dans cette file jusqu'à leur consommation par l'acteur. En utilisant les règles de sélection un acteur peut donc conserver sur ses portes tous les messages qu'il ne peut pas encore traiter.

Exemple d'utilisation:

Les messages qui ne peuvent être envoyés sur le réseau, sont maintenus à l'extérieur de l'acteur de commutation. Ce dernier n'accepte d'en traiter un (selection de la porte) que lorsqu'il a reçu l'autorisation d'émettre vers le réseau.

Les temporisations

L'incertitude sur les communications due aux erreurs possibles, la nécessité pour l'acteur d'être activé par un message au bout d'un temps fini, rendent la temporisation essentielle. Dans le cas du service de communication elle devient indispensable pour réaliser les protocoles. Le mécanisme d'armement et de réveil par message simplifie la programmation d'un protocole de communication.

Les étapes de traitement permettent d'isoler les étapes du protocole. A chaque étape de traitement l'acteur peut être activé: soit par un message du protocole, soit par un message de temporisation échue.

1.3.2/ Adaptation à la mécanique Chorus

Il subsiste des difficultés à construire un acteur avec Chorus, ou plus précisément à le programmer de façon optimale dans tous les cas de figure. En effet, chaque traitement de message nécessite une nouvelle activation de l'acteur. On peut donc être tenté, afindE de réduire le nombre de commutations, de diminuer le nombre de messages envoyés à l'acteur. Cela implique qu'un même message déclenche plusieurs actions internes à l'acteur. On perd alors la possibilité de spécifier des étapes claires, aux fonctions bien précises et gérées simplement.

Mais il est aussi bien pratique d'isoler le traitement d'un message un peu particulier du contexte général et, en particulier, de le distinguer des autres messages susceptibles d'être pris en compte par l'acteur.

Le paradoxe auquel le concepteur du service de communication se trouve alors confronté est présenté dans les deux paragraphes suivants.

1.3.2.1/ L'avantage lié à Chorus

Le fait que seul un message déclenche le traitement et que seul ce message puisse être traité dans l'étape de travail, constitue un avantage indéniable. Il devient évident lorsque l'on prend l'exemple du traitement des interruptions par un acteur.
Les messages d'IT arrivent sur ses portes, mais ils lui sont masqués tant qu'il est en train de traiter un message (ou s'il a posé une condition de selection sur la porte qui recoit les ITs). Par conséquent, ils ne viennent pas perturber son fonctionnement.

1.3.2.2/ L'inconvénient lié à Chorus

La même caractéristique qui fait l'avantage de Chorus (pour le masquage des ITs, par exemple), en fait un inconvénient car si l'acteur traite un message il lui est impossible d'en traiter un second simultanément.

L'exemple que nous prenons est le suivant:
Supposons qu'un acteur dispose de deux portes Pa et Pb. Sur Pb il recoit des messages qu'il ne peut traiter que si un certain message particulier a été reçu et traité sur Pa (droit d'émettre sur le réseau, par exemple). Il faudra deux étapes de traitement pour que chaque message de Pb soit traité: une pour le traitement du message sur Pa (qui permet la selection de Pb) et une autre pour le traitement du message sur Pb (après traitement sur Pa).

Nous faisons ces remarques pour mettre l'accent sur les nouvelles contraintes de spécification apportées au concepteur par la mécanique de Chorus.

1.4/ Les deux aspects de l'analyse

Dans les spécifications d'un acteur ou d'une application de Chorus, il est nécessaire de distinguer deux aspects de la programmation.

En effet, la programmation dans Chorus on définit deux choses:

- les programmes chargés du traitement des données,
- les enchaînements des exécutions des différentes étapes de traitement en fonction de l'évolution dynamique de l'environnement.

Le premier aspect est favorisé par l'emploi d'un langage de haut niveau et la définition d'un modèle d'acteur qui fixe le cadre d'écriture du programme acteur.

Le deuxième aspect est pris en compte par les services du noyau SELECTION, AIGUILLAGE et TEMPORISATION.

1.4.1/ La programmation des traitements

Les dispositions prises donnent toutes satisfaction pour la programmation du modèle d'acteur, même si elles peuvent être améliorées dans le sens des outils d'aide à la production du logiciel Chorus. C'est ainsi que le modèle de programme acteur est défini et catalogué. Pour écrire un acteur, il suffit d'insérer aux bons emplacements les traitements que le concepteur a prévu.

1.4.2/ Programmation de l'exécution

Nous ne cacherons pas les problèmes rencontrés pour la spécification des enchaînements des traitements des acteurs de communication en fonction des événements qui leurs sont signalés. A notre avis, ces problèmes ne remettent pas en cause les choix fondamentaux. Ils révèlent simplement des "lacunes" qui restent à combler dans la définition de l'architecture Chorus au niveau des outils d'expression de l'exécution d'une application répartie.

Il n'existe actuellement que trois services, que l'acteur doit utiliser de façon très particulière dans certains cas (cf annexe 3).

Les TEMPORISATIONs sont les plus faciles à utiliser dans la mesure où elles doivent être aiguillées vers des procédures de reprises.

Par contre, les SELECTIONs posent le problème du choix de la prochaine étape de traitement qui sera activée. Que la SELECTION

soit rémanante ou non (valable pour plusieurs ou une seule étape de traitement), nous avons remarqué qu'elle doit être traitée à part dans le programme d'acteur. En effet, une SELECTION permet d'indiquer les évènements (messages) qui vont pouvoir déclencher l'acteur.

S'il n'y en a toujours qu'un seul, le problème ne se pose pas; la SELECTION peut être redéfinie a chaque étape ou être défini jusqu'au prochain changement décidé par l'acteur.

S'il y en a plusieurs, l'activation suivante peut provoquer le changement de certaines des conditions et pas des autres. Elle peut aussi nécessiter l'ajout d'autres conditions. Dans ce cas, rémanente ou pas les SELECTIONS doivent être gérées d'une façon globale dans l'acteur, car elles expriment directement l'automate d'exécution de l'acteur.

La solution actuelle est la suivante:

On construit un graphe d'état (tableau des SELECTIONs possibles pour la prochaine étape). Il constitue la mémoire du schéma d'activation de l'acteur. Ce graphe d'état est mis à jour à chaque étape de traitement, en fonction des conditions nouvelles révélées par le traitement du message. Ce graphe d'état est transmis au noyau sous forme d'une série de messages de SELECTION identifiant les conditions d'activation de l'acteur pour la prochaine étape de traitement.

Un modèle d'exécution de l'acteur assez précis et détaillé peut alors être défini en combinant cette gestion des SELECTIONs avec une définition fine des services rendus par l'acteur (rendu possible par l'emploi de plusieurs portes).

Nous l'avons utilisée dans l'acteur de transport, pour la gestion des portes privées du mode connecté. Les portes du mode connecté sont activables si:

- le contrôle de flux est positif pour la connexion qu'elles représentent,
- l'acteur de transport peut prendre un nouveau message de connexion pour l'envoyer vers le réseau, par exemple.

Nous avons donc défini une procédure qui est appelée à la fin de chaque étape de traitement et qui prépare les SELECTIONs en fonction de l'"état" de la porte.

1.5/ Conlusions sur l'analyse

Chorus apporte une façon nouvelle de concevoir l'analyse et l'organisation des traitements. Nous ne dirons pas que cette approche est triviale pour un analyste qui découvre cette architecture. Mais la phase d'adaptation est de très courte durée et les résultats sont très encourageants parce que nets et rapides.

Nous n'avons pas caché les limites actuelles de Chorus au niveau des outils de description de l'exécution dynamique d'une application.

Le modèle de communication n'est pas un modèle d'implantation mais un excellent outil d'analyse qui permet de sérier les problèmes et de faire le tri parmi les mécanismes disponibles pour définir le service de communication.

2/ Expérience tirée de la réalisation

L'expérience que nous avons acquise lors de la réalisation su service de communication de Chorus sur la machine INTEL 8086 concerne deux domaines. Celui des mécanismes physiques de transmission et l'architecture des micro-ordinateurs. (Nous avons du, par exemple, modifier l'interpréteur de P-code afin de réaliser une procédure de transmission sur la ligne asynchrone qui relie l'ordinateur à la

boite de connexion au réseau Danube).

La construction du service de communication s'est faite par couches:

- le transfert physique local, distant,
- la commutation locale, la commutation distante des messages,
- le transfert distant.

Les transferts physiques et locaux ne posent pas de problèmes nouveaux. Ils ont été relativement rapides à réaliser.

La commutation locale demandait que le noyau soit capable de gérer les files d'attente. La commutation distante demandait que le noyau puisse récupérer les interruptions Chorus, les transformer en messages et les remettre à l'acteur de commutation distante. L'environnement nécessaire à l'exécution correcte de l'acteur de commutation nécessite donc que l'acteur de gestion des interruptions existe.

La commutation distante n'a pas posé de problème particulier de mise au point. Les sorties se font par écriture directe depuis le langage Pascal et dans le contexte de l'acteur. Les entrées se font au bas niveau qui déclenche une "interruption Chorus" et une zone de mémoire commune aux mécanismes de bas niveau et à l'acteur de commutation.

Nous n'avons pas rencontré de problème particulier pour la mise au point de la communication locale puisqu'elle est réalisée par le noyau.

La mise en oeuvre et la mise au point des acteurs et des applications sur Chorus sont facilitées par les interfaces de portes et de messages, qui permettent de tester les différentes entités indépendamment les unes des autres, dans leur phase de mise au point spécifique. Lorsque l'on passe au test d'une l'application répartie, les problèmes sont plus vite cernés et sont en général dus à des

synchronisations par message mal évaluées (erreur de temporisation). Les problèmes de ce genre se découvrent très rapidement avec les outils de trace fournis par le noyau (état des files d'attente, nom des services noyau invoqués dans chaque étape de traitement, nom des portes activées, etc..)

Dans la première implantation, exceptée la procédure de transmission sur la ligne asynchrone, qui relie l'ordinateur et le communicateur d'accès au réseau, tout le service de communication est écrit en Pascal. Si les performances en souffrent un peu, la facilité de programmation, de mise au point, la portabilité nous permettent de tester très rapidement les différentes solutions possibles au niveau des protocoles de communications.

2.1/ Expérimentation des connexions U

Cette expérimentation est en cours. Nous avons réalisé une maquette constituée de deux acteurs de transport installé sur deux sites différents. Ils utilisent une connexion TD sur le réseau et gèrent une connexion U entre deux acteurs utilisateurs. Il n'y a pas d'anticipation sur la connexion U entre utilisateur. Par contre, il y a de l'anticipation sur la connexion TD.

Les résultats obtenus sont intéressants dans la mesure où:

- nous n'avons pas remarqué de blocage des acteurs utilisateurs dû à un mauvais fonctionnement du dialogue entre eux et l'acteur de transport dont ils dépendent,
- il n'y a pas de perte de message au niveau de la connexion U,
- le séquencement des messages des acteurs utilisateurs est respecté,

- du fait de l'anticipation sur la connexion distante, on ne remarque pas de gêne particulière liée à l'absence d'anticipation entre l'acteur de transport et l'utilisateur, il ne nous semble pas très important dans cette implantation de faire de l'anticipation (possible) localement, car le temps pris par l'acteur de transport (pour le traitement d'un message à transmettre) et la transmission, en elle-même, sont très supérieurs au temps de commutation locale des messages entre l'acteur utilisateur et l'acteur de transport. Nous ne ferions donc qu'augmenter le nombres de message en attente sur la connexion U, et par là même, nous risquerions de provoquer la congestion du service de communication.
  Nous envisageons de poursuivre cette expérimentation sur la SM90, où nous devrons réaliser de la fragmentation et du réassemblage entre les machines puisque les messages utilisateurs pourront avoir la taille maximale du segment machine (64 Koctets). Les temps de commutation de contexte et de traitement d'un message y sont bien plus faibles. L'accès au réseau doit aussi être plus rapide.

En conclusion sur l'expérimentation nous dirons que le véritable problème est la performance des mécanismes de communication que nous mettons en place et non leur mise au point. Ces performances sont influencées par deux choses: la lenteur du code utilisé pour la programmation, la complexité de certains protocoles de communication.

A cet égard l'expérimentation du mode non connecté et du protocole de connexion U nous a convaincu; l'utilité de ce dernier reste à démontrer sur des réseaux de bonne fiabilité. L'expérimentation sur des gros réseaux maillés et des applications utilisant de gros transferts de données apporterons peut-être une réponse.

" Rien de ce qui est commencé
n'est complètement achevé
tant que tout ce qui est entrepris
n'est pas totalement terminé "

P. DAC
(Pensées)

CONCLUSIONS

Au terme de cette étude sur le service de communication de l'architecture Chorus, nous pensons que nos principaux objectifs ont été atteints.

Nous avons défini un modèle de communication, nous avons construit un service de communication qui répond à l'ensemble des appplications qu'il nous est possible d'imaginer réalisables avec Chorus.

On peut tirer un bilan en deux parties:
- l'une sur l'adéquation du modèle aux problèmes de relations entre traitement et transport que nous voulions résoudre,
- l'autre sur l'adéquation de nos choix de spécification aux problèmes de construction et d'exécution du service de comunication.

Modèle Chorus et modèle ISO

Prendre le parti de définir un modèle de communication n'est réaliste que dans un contexte de traitement bien structuré. En ce sens, le fait de vouloir des traitements atomiques, une séparation nette du transport et du traitement ont grandement orienté notre choix vers le modèle ISO.

Mais il convient de nuancer notre propos, en raison de la comparaison que le lecteur ne peut manquer pas de faire entre le modèle ISO et le modèle de communication de Chorus qui, il faut le souligner, est d'abord défini pour être appliqué à une architecture particulière.

L'architecture Chorus n'est pas liée à un produit ou une philisophie de constructeur. La cohérence, la simplicité, l'homogénéité ont été recherchées dés le départ. Notre tâche, en ce qui concerne l'évaluation des difficultés à résoudre pour construire un service de communication, en a été allégée d'autant.

Le modèle de communication de Chorus est conforme aux principes de l'ISO sur les points suivant:

- la décomposition en niveaux de services,
- les protocoles et interfaces d'accès au services d'une couche,
- les protocoles de service entre les homologues d'une couche.

Les seuls éléments que nous venons de citer suffisent, en général, à entretenir la polémique quant à la complexité de mise en oeuvre de tels principes, surtout si l'on utilise des matériels hétérogènes. Mais, dans le cas de Chorus, le modèle a pu être appliqué, parce que l'architecture Chorus comporte des choix qui simplifient la réalisation du service de communication. Il est honnête de reconnaître qu'ils ont été possibles parce que:

- nous avons la maîtrise de toute la chaîne de communication, depuis les éléments physiques jusqu'aux éléments logiques complexes de transport. Il nous a été possible de concevoir un ensemble homogène, du point de vue de ses spécifications,

- il n'y a qu'un seul type d'interface (les portes et les messages envoyés sur ces portes), ce qui permet de limiter le nombre des mécanismes d'échanges de données et le nombre de formats de données,

- nous avons défini un mécanisme de désignation qui facilite le routage des unités de données vers les portes, car, dans la majeure partie des cas, l'interprétation directe du nom permet de déterminer le site récepteur. Ce mécanisme évite le maintien (et le parcourt) de tables de routage (souvent coûteux et parfois complexe à réaliser),

- le noyau est entièrement spécifié et conçu pour Chorus. On peut y regrouper toutes les fonctions locales de communication afin d'améliorer les performances et la programmation (interface,

protocoles d'échanges de données) sans aller à l'encontre des principes du modèle.

Des protocoles de base simples

Nous avons opté pour une simplification des protocoles d'échanges de données, puisque le mode non connecté est le service de transport de base. Nous expliquons cela par le fait que la plupart des applications visées par Chorus se font sur des matériels dont la fiabilité est bonne (réseaux locaux, machines à protection mémoire cablée, etc.). Il faut y ajouter une façon particulière de communiquer (appel/réponse) qui améliore la qualité des échanges, grâce à la redondance qu'ils apportent sur les informations de transport.

Un modèle strict

Notre analyse du service de communication peut paraître parfois un peu "tirée par les cheveux", puisque faisons apparaître des niveaux jusque dans le noyau. Cette impression est légitime, nous semble-t-il, si l'on considère que les mécanismes de communication, sur une machine, ne sont en général pas aussi finement décomposés que dans notre approche. Mais n'est-ce pas pour la bonne et simple raison que la plupart des approches ne considèrent pas réellement le système local comme un outil de communication.

Pour les communications distantes, les habitudes en matières de réseau montrent une tendance à définir pour chaque site raccordé au réseau une "Station de Transport" qui est, en général, bien distincte des mécanismes locaux de communication et accédée directement par les processus utilisateurs.

Dans Chorus, la station de transport existe, mais en tant qu'acteur

de "l'application répartie de transport distant". Elle est, de ce fait, masquée aux acteurs utilisateurs.

Réalisation du service

Nous avons pu construire un service de communication complet, en partant du support physique jusqu'aux traitements des applications dans les acteurs utilisateurs. Ce service de communication prend en compte de façon optimale les problèmes d'exécution et de contrôle des échanges de données à chaque niveau en permettant la mise en oeuvre des protocoles appropriés à chaque fois.

Il existe donc beaucoup d'implantations possibles du service de communication, mais la réalisation que nous avons fait et celle qui est en cours sur SM90, démontrent la souplesse de configuration de ce service.

Il reste cependant à faire progresser le modèle et l'architecture Chorus. La réalisation du service de transport a permit de faire ressortir quelques aspects encore mal stabilisés dans Chorus.

a) La structuration des données.

On parle beaucoup d'unités de données utilisateurs échangées entre les portes des acteurs utilisateurs, mais leur structuration est plutôt libre. Seules les conventions prévues par les progammeurs des applications définissent le format des données. Ceci est valable aussi bien pour les services système que pour les applications utilisatrices. Pour le transport et la communication dans les couches basses on arrive aisément à définir un format standard. La technique utilisée est l'emballage des unités de données avec des entêtes qui sont ajoutées et enlevées par les différents services rencontrés lors du transport, mais à aucun

moment il n'est tenu compte de la structure des données de traitement.

On peut se demander s'il est possible de définir une structure d'unité de données qui soit compatible avec les traitements et le transport des données. L'idée serait de définir des blocs d'informations qui contiendraient les données nécessaires à leur traitement et à leur transport.

b) le service de transport.

Le mode non connecté est facilement accessible des acteurs puisqu'il est implicite et activé par la déclaration de fin d'étape de traitement. Le mode connecté doit, pour l'instant, être explicitement demandé par l'acteur qui doit savoir ouvrir, suivre et fermer une connexion utilisateur. On peut envisager de rendre le mode connecté implicite à condition de simplifier sa mise en oeuvre par l'acteur utilisateur.

Pour cela, il faut:

* vérifier automatiquement que l'acteur ne peut envoyer que le nombre de messages autorisé par le crédit (vérifié par le noyau à l'allocation du tampon de message),
* que l'acteur puisse être programmé de telle façon que la bonne étape de traitement soit activée suivant le type de "message" recu. (credit positif = vous pouvez émettre, données recues = vous pouvez traiter mais peut-être pas émettre, etc...)

c) les connexions.

Le service de transport en mode connecté pourraient être développées et offrir des facilitées tels que:

* la possibilité d'établir des connexions utilisateur permanentes, à charge du service de transport de les rétablir

si elles sont interrompues par des incidents dus au transport.

* installation d'un mécanisme de passage automatique en mode connecté si deux portes coopérantes sont sur des sites différents. Cela pourrait être demandé, comme une "option de transport", par les acteurs utilisateurs.

* le transfert de données express. Le mode connecté actuel ne réalise pas le transfert de données express. Cette facilité n'est pas offerte, parce que nous ne voyons pas d'application qui en aurait bseoin (du moins pour le genre d'utilisation des connexions U que nous envisageons). Cependant si certaines applications utilisaient le mode connecté de facon permanente, il pourrait être utile d'offrir le transport de données express.

e) les acteurs.

Nous avons limité la décomposition en niveaux de Chorus aux acteurs utilisateurs. Il faudrait aller plus loin c'est-à-dire étudier l'organisation des traitements à l'intérieur de l'acteur. Il est sans doute possible d'isoler des fonctions qui sont propres à des services de couches session ISO (protocole de synchronisation entre acteur d'une application, numérotation des séquences d'unités de données), et même de couche présentation (conventions de présentation des données des traitements distribués).

L'intérêt de cette décomposition plus fine résiderait surtout dans le fait qu'il serait alors possible d'identifier des modules ou des procédures standards pour chaque niveaux. Le concepteur d'une application n'aurait plus qu'à définir les étapes de traitements et les échanges du niveau application proprement dit.

Les étapes traitements des autres niveaux étant insérés

automatiquement à la compilation du modèle d'acteur ou liés dynamiquement au code de l'acteur à l'édition de lien.
Une autre solution étant, bien sûr, de définir des acteurs de niveau session et présentation.

Nous pensons que tous ces développements doivent se faire avec une évolution de notre conception des communications, qui devrait nous amener à reconsidérer notre modèle afin d'y inclure certains avantages de l'approche intégrée que nous n'avons pas retenue dans notre premier choix.

Chorus_et_l'approche_intégrée

Nous avons opté pour une décomposition en couches qui permettait de mieux respecter les principes d'atomicité. Si l'on considère le principe de fonctionnement de l'acteur, le seul obstacle à l'approche intégrée résidait dans le fait qu'il soit nécessaire de suspendre l'exécution de l'étape de traitement et de la reprendre à l'instruction qui suit l'appel de procédure distante, après avoir sauvegardé le contexte de l'acteur.
Mais, dans ce cas, qu'est-ce qui nous empêche de faire que l'étape de traitement ne se termine à l'appel de la procédure distante et qu'une nouvelle étape commence à l'instruction suivante? Surtout le fait que le langage de programmation Pascal ne permette pas la déclaration de points d'entrée dynamiques visibles de l'extérieur.
Cette constatation n'est pas nouvelle [GUI 82], mais nous pouvons la reprendre aujourd'hui car cet exemple nous permet de constater que les raisons invoquées pour l'un ou l'autre des choix tiennent plus à des problèmes de programmation et compilation des traitements qu'à des problèmes de communication.

Sans ce problème (et bien qu'il ne soit pas négligeable), il n'est pas interdit de penser que l'on possédera demain un langage qui

favorisera la construction de systèmes réalisés avec une interface d'accès au service de communication du type appel de procédure distante. Le service de communication quant à lui pourra être organisé suivant le modèle que nous avons décrit dans cette étude.

On imagine l'intérêt d'un modèle qui ferait l'union de ses deux approches. Il suffit pour s'en convaincre de voir comment il a été possible de respecter les principes du modèle que nous étions fixé du point de vue de l'organisation générale des communications. Les difficultés sont surtout apparues au niveau des interfaces entre acteur et noyau. Il a fallu les résoudre par les procédures d'interfaces qui, si elles facilitent la programmation de l'acteur, ne constituent pas véritablement le lien dynamique entre le transport et les traitements.

Voici quelques caractéristiques du modèle que nous aimerions réaliser:

- une organisation en couches des opérations associées aux communications,
- une transparence des traitements aux problèmes de communication; cela serait rendu possible par l'utilisation des appels de procédure distante dans l'interface entre les acteurs et le service de communication et à la définition (dynamique ou statique) de points de synchronisation au niveau langage (point d'entrée) pour le transport dans le programme acteur; tout le contrôle des communications pouvant se faire dans le contexte de l'acteur sans perturber les traitements spécifiés par le concepteur,
- respecter l'atomicité des étapes de traitement; on peut écrire un programme séquentiel avec des points d'entrée connus de l'extérieur et définis automatiquement à l'appel de procédure distante;

Nous ne sommes pas en mesure de dire si ce modèle est pour un avenir proche puisqu'il dépend beaucoup des progrès dans les langages et la compilation (une simulation en est faite dans Chorus par un appel de procédure externe). Pour notre part, nous espérons avoir l'occasion de le simuler en construisant des interfaces de procédures entre les acteurs utilisateurs et le service de communication qui nous rapprocheront de cette association "appel de procédure distante et communication en couches".

Cette démarche de fédération de ces deux approches serait conforme aux principes qui ont prévalu jusqu'à présent dans les travaux de Chorus, à savoir que notre but est moins de réformer que d'harmoniser des concepts existants.

Nous laisserons le mot de la fin à notre maître H. Zimmermann qui a fait la remarque suivante qui résume, de façon fort imag notre démarche; "Comme pour toutes les recettes de cuisine qui utilisent les mêmes ingrédients, certains résultats ont meilleur goût que d'autres". Puisse-t-il en être de même avec le modèle de communication de demain.

Dans ce cas, pourquoi ce nouveau modèle ne s'appellerait-il pas: le modèle CHORISO?

[AFN 82] AFNOR
Système de traitement de l'information,
Modèle de référence de base pour l'interconnexion de systèmes ouverts.
Z 70-001 AFNOR, norme expérimentale (juin 82)

[ANS 79] ANSI
Further Refinements to the Propsoed Datagramme Interface [for X25].
Computer Communication Review 9,1 (janvier 79)

[BAN 79] J.S. Banino, C. Kaiser, H. Zimmermann
Synchronisation for distributed systems using a single broadcast channel.
Fisrt Int. Conference on Distributed Computing Systems, Hunstville, Alabama (October 1979)

[BAN 80] J.S. Banino, A. Caristan, M. Guillemont, G. Morisset, H. Zimmermann
CHORUS : an architecture for distributed systems
Rapport INRIA 42, (Novembre 1980)

[BOG 80] D.R. Boggs, J.F. Shoch, E.A. Taft, R.M. Metcalfe
Pup: an internetwork architecture
IEEE Transactions on communications, vol Com28,4,(Avril 80)

[CCI 80] CCITT Rapporteur Report,
Characteristics of Concern ta Transport Services Working Paper,
ANSI Document Number X 3S37-80-32R, (Avril 80)

[CER 74] V. Cerf,R. Kahn,
A protocol for Packet Network Intercommunication,
IEEE Transactions Communications COM-20,5 (74)

[CER 78] V. Cerf, A. Mac Kenzie, R. Scantlebury, H. Zimmermann
Proposal for a Internetwork end-to-end protocol.
INWG General Note # 96.1 (Janvier 1978)

[CHA 72] J.F. Chambon, M. Elie, J. Le Bihan, H. Zimmermann
Spécifications fonctionnelles des stations de transport du réseau Cyclades. Protocole ST-ST.
Publication INRIA, Réseau Cyclades, SCH502.2 (Novembre 1972).

[CLI 76] W.W. Clipsham, F.E. Glave, M.L. Narraway
Datapack Network Overview
Proceedings of the 3rd International Conference on Computer Communication, Toronto, (Aout 76)

[COR 81] CORNAFION (Groupe de travail)
Les systèmes répartis, concepts et techniques
Dunod, Paris, 1981, 367 p.

[DAV 82] J.M. Davidson
Connection-oriented Protocols of Net/One
International Symposium on Local Computer Networks
Florence, Italie (Avril 1982)

[DES 76] A. Desprès, G. Pichon, A. Danet, A. Le Rest, S. Ritzenthaler
The French public packet switching service: the Transpac Network,
Proc. of the Int. Computer Communications Conférence, Toronto (Août 76)

[DON 79] J.E. Donnelley,
Components of a Network operating System,
Proc. 4th CLCN, (Octobre 79), IEEE 79CH 14464C,1-12

[ELI 75] M. Elie, H. Zimmermann
Transport protocol. Standart end-to-end protocol for heterogenous computer networks,
IFIP WG6.1, INWG 61, (Mai 1975)

[ENS 77] P.H. Enslow
Multiprocessor organisation: a survey,
Compcon Fall 77

[ENS 78] P.H. Enslow
What is a "Distributed" Data Processing System?
Computer 11,1, (Janvier 78)

[ELO 77] H.S. Elovitz, Heitmeyer C.L.
"What is a computer network?"
Compcon Fall, (Septembre 77)

[FAR 72] FARBER D.J., LARSON K.C.
The Structure of a Distributed Computing System: the Communications System
Proc. Symposium on Computer-Communications Networks and Teletraffic
Brooklyn, New-York, April 1972, pp. 21-27

[FAR 72b] FARBER D.J., LARSON K.C.
The structure of a distributed computing system-software
Symposium on Computer-Communications and Teletraffic
Brooklyn, New-York, April 1972, pp. 539-545

[FIN 81] U. Finger, G. Medigue
Architectures multi-micro-processeurs et disponibilié: la SM90
L'écho des recherches, No 105, (Juillet 1981)

[GAI 82] C. Gailliard, C. Senay
L'architecture répartie CHORUS : Le système d'exploitation
SEIR 2, Universidad de Santigo de Compostela (Septembre 1982)

[GAR 74] C. Garcia, R. Gardien, A. Marchand, M. Martin, H. Zimmermann
Spécifications de réalisation de la Station de Transport ST2 portable.
Publication INRIA, Cyclades SCH536 (Octobre 1974)

[GAU 82] S. Gaucher-Cazalis
Conception d'une machine de transport dans un environnement de réseau local.
Thèse de Docteur-Ingénieur, Univeristé Rennes 1 (Juillet 1982)

[GRA 77] J.L. Grange, H. Zimmermann
Les réseaux à commutation de paquets : principes et exemples,
Symposium Granit, Rennes, (Juin 1977)

[GUI 82] M. Guillemont
Intégration d'un système réparti, CHORUS, dans un langage de haut niveau, PASCAL
Thèse de Docteur-Ingénieur, Grenoble, Mars 1982

[HEA 73] F.E. Heart
The Arpa Network.
Computer communication Networks, edited by R.L. Grimsdale and F.F. Kuo, Nato Advanced study Institutes Series, (Septembre 73).

[HER 78] D. Herman, J.P. Verjus
An Algoritme For maintaining the Consistency of Multiple Copies
Proc. 1st ICDPS, (octobre 78)

[HOL 78] A.W. Holt, J.M. Myers
An approach to the Analysis of Clock Networks,
Boston University Report, (Juillet 78)

[HOP 80] A. Hopper
The Cambridge Ring - A local network.
Techniques de communication et réseaux locaux d'ordinateurs, Edinburgh, (Septembre 80).

[ISO 81] Data Processing - Open system Interconnection - Basic reference model.
ISO/TCG7/SC16 N537 - Rev.DP7498 (Août 1981)

[LAM 78] L. Lamport
Time, Clocks and ordering of Events in a distributed System,
Communication ACM 21,7 (Juillet 78)

[LEV 79] LEVIN Roy, SCHROEDER Michael D.
Transport of Electronic Messages Through a Network
Teleinformatics 79, 1979, pp. 29-33

[MAC 77] C. Macchi, J.F. Guilbert
Transport et traitement de l'information dans les réseaux et systèmes télé-informatiques.
Dunod Informatique

[MAR 81] M. Martin, C. Mercier-laurent, N. Naffah, B. Scheurer
Les réseaux locaux : Définitions et exemple de Réalisation (DANUBE),
Journées KAYAK, Paris, (Mars 1981)

[Met 76] R.M. Metcalfe, D.R. Boggs
Ethernet: Distributed Packet Switching for Local Computer Networks
CACM vol. 19 No 7 (Juillet 1976)

[NAF 79] N. Naffah
Présentation du projet Pilote Burotique KAYAK.
Publication INRIA, projet pilote KAYAK, ORG 2.507 (juin 1979)

[NEL 81] B.J. Nelson
Remote procedure call.
Technical Report CSL-81-9, Xerox Palo Alto Research Center. 81

[POU 75] POUZIN Louis
Les réseaux, concepts et structures
Note SCH 577, INRIA, Décembre 1975, 101 p.

[POU 79] POUZIN Louis, ZIMMERMANN Hubert
Protocols and Components
CYCLADES - Chap. III
24 September 1979

[QUI 79] V. Quint, N. Naffah
Protocole de transport pour Réseaux locaux.
Publication INRIA, Projet Pilote KAYAK, Rel 2.504.1 (Février 1979)

[RAS 81b] RASHID Richard F., Robertson George G.
Accent: A communication oriented network operating system kernel
Note CMU-CS-81-123, Carnegie Mellon University
Draft, April 1981, 22 p.
et:
8th Symposium on Operating Systems Principles
ACM, Vol. 15 No 5, Pacific Grove, California, December 1981, pp. 59-63

[SCH 80] B. Scheurer, N. Naffah et Al
Description fonctionnelle du réseau expérimental DANUBE,
Publication INRIA, projet pilote KAYAK, (Juillet 80)

[SHO 78] J.F. Shoch
Inter-Network Naming, Adressing, and Routing.
Compcon 78

[SHO 79] J.F.
Tactics and Strategy for Packet Fragmentation in internetwork Protocols.
Computer Networks (79)

[SLO 79] L.J. Sloan
Limiting Lifetime of packets in Computer Networks,
Proc. 4th CLCN, (Octobre 79)

[SPE 82] A.Z. Spector
Performing remote operations efficiently on a local computer network.
Communication ACM 25,4 (Avril 82)

[SPR 78] R.F. Sproull, D. Cohen
High Level Protocols,
Proc. IEEE 66, (Novembre 78)

[SUN 75] C.A. Sunshine
Interprocess communication Protocols for Computer Networks
DSL Technical Report 105, Standford University, (Decembre 75)

[SUN 76] C.A. Sunshine
Factors in Interprocess communication Protocol Efficiency for Computer Networks,
Proc. AFIPS Conf. 45, (76)

[SWA 77] Richard J. Swan
A modular Multi-Microprocessor,
Proc. AFIPS Conf. 46, (77)

[TOT XX] Informaticien Inconnu
Désignation d'une référence dans un programmme (procédure, fichier, variable.
Source inconnue, remonte dans la Nuit des Temps Informatiques.

[WAR 80] S.A. Ward
TRIX: a network-oriented operating system
Compcon 80, San Franscisco, California (Fevrier 1980)

[ZIM 75] H. Zimmermann
The Cyclades end-to-end protocol,
Fourth Data Communications Symposium, Québec City, (Octobre 1975)

[ZIM 77] ZIMMERMANN Hubert
The CYCLADES Experience - Results and Impacts
IFIP Congress, Toronto, Canada, August 1977

[ZIM 81] ZIMMERMANN H., BANINO J.S., CARISTAN A., GUILLEMONT M., MORISSET G.
Basic Concepts for the support of distributed systems: the CHORUS approach
2nd International Conf. on Distributed Computing Systems, Versailles 1981, pp. 60-66

ANNEXE 1

ANNEXE 1

Le modèle ISO pour l'interconnexion de systèmes ouverts introduit la notion de système_ouvert (connectable à d'autres systèmes ouverts) et de sous-systèmes qui sont les parties du système ouvert.

Le but principal de ce modèle est d'offrir une architecture de convergence pour la construction de systèmes organisés en systèmes ouverts interconnectés.

La caractéristique principale en est le découpage en couches fonctionnelles de chaque système ouvert. Les couches sont hiérarchisées. De ce fait, elles sont identifiées par un rang dans la hiérarchie et une couche de rang N est un ensemble de sous-systèmes de même rang. Pour le lecteur non familiarisé avec ce modèle, nous donnons d'abord la structure et les grands principes de construction du modèle qui fait l'objet de documents édités en français, notamment par l'AFNOR, et dont nous tirons la documentation qui suit.

1/ Principe général

Le modèle consiste en un certain nombre de définitions et de principes fondamentaux à la description et à la conception d'une architecture de système ouvert structuré en couches.

On trouve successivement présentés:

- les principes de la structuration en couches; c'est une suite de définitions des termes et des concepts employés pour décrire le modèle,
- les principes de la communication; c'est l'ensemble des définitions et principes pour la communication entre les éléments actifs de l'architecture,
- les principes d'identification; c'est l'ensemble des définitions et principes pour la désignation et l'identification des

éléments de l'architecture,

- les principes pour les unités de données; c'est l'ensemble des principes et des définitions qui régissent les formats des données échangées'
- les principes pour le fonctionnement d'une couche; c'est l'ensemble des définitions et des principes se rapportants aux fonctions réalisées dans une couche,
- les principes du routage; la définition du routage et le prinicipe de son installation sur un niveau.

Certaines définitions ont été volontairement supprimées puisque notre but est essentiellement d'indiquer ce que le modèle nous apporte d'éléments pour décrire et construire la fonction de communication de Chorus.

2/ Description

2.1/ Principes de la structuration en couches

2.1.1/ Définitions

Les principales définitions des termes utilisés pour décrire cette architecture sont:

sous-système (N)	:	élément d'une division hiérarchique d'un système n'ayant d'interaction qu'avec les éléments des niveaux immédiatement supérieur et inférieur de cette division, avec (N+1) et (N-1),
couche (N)	:	ensemble de sous-systèmes de rang (N),
entité (N)	:	élément actif d'un sous-système (N),
entité homologue	:	entité appartenant à une même couche,
service (N)	:	ensemble des fonctionnalités possedées par couches N (et les couches < N) et fournies aux entités (N+1),
facilité (N)	:	élément de service,

fonction (N) : élément participant à l'exécution d'un service,

point d'accès à des services (N) : le point où les services (N) sont fournis par une entité (N) à une entité (N+1),

protocole (N) : règles et formats régissant les échanges entre entités (N),

connexion (N) : liaison établie par la couche (N) entre deux ou plusieurs entités de couche (N+1). plusieurs entités.

Nous utiliserons ces termes et définitions pour identifier (nom et fonctions) les éléments de Chorus qui présentent une certaine analogie avec eux.

2.1.2/ Description

La description reprend les définitions précédentes et en donne une explication descriptive.

Un système ouvert est composé de sous-systèmes ordonnés verticalement. Les sous-systèmes adjacents doivent communiquer à travers leur frontière commune. L'ensemble des sous-systèmes d'un même rang (N) constituent la couche (N) du modèle de référence de l'ISO.

Chaque couche fournit des services aux entités de la couche supérieure (à l'exception de la couche la plus élevée). On peut adapter chaque service fourni par une couche (N) en choisissant une ou plusieurs facilités (N) qui déterminent les attributs dudit service.

Quand une entité ne peut assurer intégralement par elle-même un service demandé par une entité (N+1), elle fait appel à la coopération d'autre entités (N). Pour coopérer, les entités (N) d'une couche (sauf celles de la couche la plus basse) communiquent au moyen de l'ensemble des services fournis par la couche (N-1).

Les services d'une couche (N) sont fournis à la couche (N+1) grâce aux fonctions effectuées à l'intérieur de la couche (N) et suivant le besoin, avec l'aide des services offerts par la couche (N-1).

Une entité (N) peut fournir des services à une ou plusieurs entités (N+1) et utiliser les services d'une ou plusieurs entités (N-1).
Un point d'accès à des services (N) est un point où se rejoignent deux entités situées dans des couches adjacentes.

La coopération entre entités (N) est régie par un ou plusieurs protocoles (N).

2.2/ Principe de communication

2.2.1/ Définitions

Les définitions données ici nous intéressent sous deux aspects; elles définissent un concept important qui est la connexion (qui permet "mode connecté"),
elles introduisent des termes qui nous permettront désigner clairement certaines entités ou certains concepts que nous utiliseront.

La connexion

Connexion (N)	: association établie par la couche (N) entre deux ou plusieurs entités de la couche (N+1) pour le transfert de données.
Extrémité de connexion	: terminaison d'une connexion (N) en un point d'accès à des services (N).
Connexion multipoint	: connexion comportant plus de deux extrémités de connexion.

Les autres concepts

Entités correspondantes : entités (N) reliées par une connexion (N-1).

Relais (N) : fonction (N) au moyen de laquelle une entité (N) retransmet des données reçues d'une entité correspondante (N) à une autre entité correspondante (N).

Source de données : entité (N) qui envoie des unités de données du service (N-1) sur connexion (N-1) (voir plus loin "unité de donnée de service").

Collecteur de données : entité (N) qui reçoit des unités de données du service (N-1) sur connexion (N-1).

Transmission de données (N) : service (N) transportant des unités de données du service (N) d'une entité (N+1) à une ou plusieurs entités (N+1) via des connexions (N).

Communication de données : fonction (N) transférant des unités de données du protocole (N) sur une ou plusieurs connexion (N-1) et conformément à un protocole (N).

Communication bidirectionelle simultanée (N) : communication de données dans les deux sens à la fois.

2.2.2/ Description

Pour pouvoir échanger des informations entre deux ou plusieurs entités (N+1), il faut établir entre elles une association dans la couche (N), en suivant un protocole (N).

Chaque association est appelée une connexion (N). Les connexions (N) sont établies entre au moins deux points d'accès à des services (N). La terminaison d'une connexion (N) à un point d'accès à des services (N) est appelée extrémité de connexion.

Les entités (N+1) ne peuvent faire de la communication entre elles qu'en se servant des services de la couche (N). Elles utilisent pour cela les services de transmission de données de la couche (N). Dans

certaines circonstances, les services fournis par la couche (N) ne permettent pas des liaisons directes entre toutes les entités (N+1) ayant à communiquer. Dans ce cas, la communication peut avoir lieu si d'autres entités (N+1) sont capables de réaliser la fonction de relais entre elles.

2.3/ Principes d'identification

Ces principes sont importants car ils permettent de nommer les éléments de l'architecture.

2.3.1/ Les définitions

Appellation : identificateur permanent d'une entité.

Domaine d'appellation : sous-ensemble de l'espace d'appellation de l'environnement ISO (exemple: transport)

Nom de domaine d'appellation : identificateur désignant de manière unique un domaine d'appellation dans l'environnement ISO (exemple: couche transport).

Adresse (N) adresse de point d'accès à des services (N) : identificateur indiquant où se trouve un point d'accès à des services (N).

Mise en correspondance d'adresse (N) : fonction (N) assurant la mise en correspondance des adresses (N) et des adresses (N-1) associées à une entité (N).

Routage : fonction d'une couche traduisant l'appellation d'une entité (ou l'adresse du point d'accès (aux services) auquel l'entité est reliée) en un itinéraire permettant d'atteindre cette entité.

Identificateur d'extrémité de connexion (N) : identificateur de l'extrémité d'une connexion (N) destiné à identifier, en un point d'accès à des services (N), la connexion (N) correspondante.

2.3.2/ Description

Nous ne nous étendrons pas sur cette partie qui se comprend relativement bien d'elle-même et nous résumons l'adressage dans la figure qui suit.

figure 2.1 : Description

2.4/ Principe de structuration des données

2.4.1/ Les définitions

Informations de contrôle du protocole (N) : informations échangées entre entités (N), via une connexion (N-1), pour coordonner leur travail commun.

Données utilisateur (N) : Données transférées entre entités (N) pour le compte d'entités (N+1) auxquelles les entités (N) sont en train de fournir des services.

Unité de données du protocole (N) : Unité de données spécifiée dans un protocole (N) et consistant en informations de contrôle du protocole (N) et éventuellement, des données utilisateur (N).

Informations de contrôle de l'interface (N) : Informations transférées entre une entité (N+1) et une entité (N) pour coordonner leur travail commun.

Données de l'interface (N) : Informations transférées d'une entité (N+1) à une entité (N) pour transmission sur une connexion (N) à une entité (N+1) correspondante. Réciproquement, informations transférées d'une entité (N) à une entité (N+1) après avoir été recues d'une entité correspondante (N+1) sur une connexion (N).

Unité de données de l'interface (N) : Unité d'information transférée, en une seule interaction, par le point d'accès à des services (N) d'une entité (N+1) à une entité (N). Chaque unité de données de l'interface (N) contient des informations de contrôle de l'interface (N) et peut contenir tout ou partie d'une unité de données du service (N).

Unités de données du service : Ensemble de données de l'interface (N) dont l'identité est préservée d'une extrémité à l'autre d'une connexion (N).

2.4.2/ Description

Les unités de données du service contiennent toutes les informations caractérisant le service. Elles sont construites lors de l'accès au service et ne peuvent être modifiées (dans leur structure) qu'à la sortie du service.

La description se fait par le tableau qui suit.

	Pour le contrôle	Pour les données	leur combinaison
Entités homologues (N) - (N)	Informations de contrôle du protocole (N)	données utilisateur (N)	unité de données du protocole (N)
Entités de couches	Informations de contrôle de	données de	unité de données

adjacentes (N+1) - (N)	l'interface (N)	l'interface (N)	de l' interface (N)

figure 2.2 : Correspondance entre les unités de données

2.5/ Principes de fonctionnement d'une couche

2.5.1/ Définitions

Identificateur de protocole (N) : Identificateur utilisé entre les entités (N) correspondantes pour déterminer quel protocole (N) particulier utiliser sur une connexion (N-1) donnée.

Multiplexage : fonction d'une couche (N) permettant de prendre en charge plusieurs connexions (N) sur une seule connexion (N-1).

Démultiplexage : fonction inverse du multiplexage.

Eclatement : fonction de la couche (N) permettant d'utiliser plusieurs connexions (N-1) pour prendre en charge la connexion (N).

Recombinaison : fonction inverse de l'éclatement.

Contrôle de flux : fonction contrôlant le flux des données au sein d'une couche ou entre deux couches adjacentes.

Segmentation : fonction accomplie par une entité (N) pour mettre en correspondance une unité de données du service (N) avec plusieurs unités de données du protocole (N).

Réassemblage : Fonction inverse de la fonction de segmentation.

Le modèle définit ensuite une série de définitions de principes bien connus de tout le monde et qui ne peuvent conduire à aucune ambiguité quant à leur interprétation. C'est pourquoi nous nous contenterons de les énumérer:

- le maintien en séquence des unités de données,
- l'accusé de réception d'une unité de données,

2.6/ Principe du routage

Une fonction de routage, à l'intérieur d'une couche (N), permet de relayer la communication à l'aide d'une chaîne d'entités(N). Le fait qu'une communication soit acheminée par des entités intermédiaires (N) n'est pas connu des couches adjacentes. Une entité (N) qui participe à une fonction de routage peut comporter une ou des tables de routage.

3/ Le choix des couches

Les principes qui ont dicté ce choix sont les suivants:

- la décomposition en couche est intéressante, mais il convient de limiter celles-ci,

- créer des couches séparées :
  * afin de séparer les différentes fonctions (traitement/technologie),
  * afin de regrouper les fonctions similaires dans des domaines clairement identifiés,
  * afin d'obtenir le même niveau d'abstraction de manipulation de données,
  * afin que la modification d'une couche ne modifie pas les services fournis ou attendus par les couches adjacentes.

Nous ferons la description de l'architecture en couches en partant des sous-systèmes physiques d'interconnection pour en arriver aux sous-systèmes logiques de traitement des données.

La normalisation fait ainsi apparaître sept couches, numérotées de 1 à 7, dont voici les principaux critères de distinction:

couche 1 : Description et intégration des supports physiques auxquels sont associés les modems, les lignes et les procédures de contrôle de transmissions.

couche 2 : Abaissement du taux d'erreur par les procédures de contrôle des liaisons de données.

couche 3 : Ensemble des systèmes de terminaison de liaison de données (terminaux et noeuds). Cette couche regroupe les fonctions d'acheminement des blocs de données (paquets).

couche 4 : Optimiser l'utilisation des services du réseau et offrir un contrôle de bout en bout sur l'acheminement des messages.

couche 5 : Organisation, synchronisation et gestion des échanges des données.

couche 6 : Définir et assurer la représentation et la manipulation des données.

couche 7 : Tout ce qui ressort du traitement direct des données et qui est propre à l'application.

3.1/ Les différentes couches du modèle

Avant de décrire les différentes couches, nous en donnons la représentation graphique dans la figure qui suit:

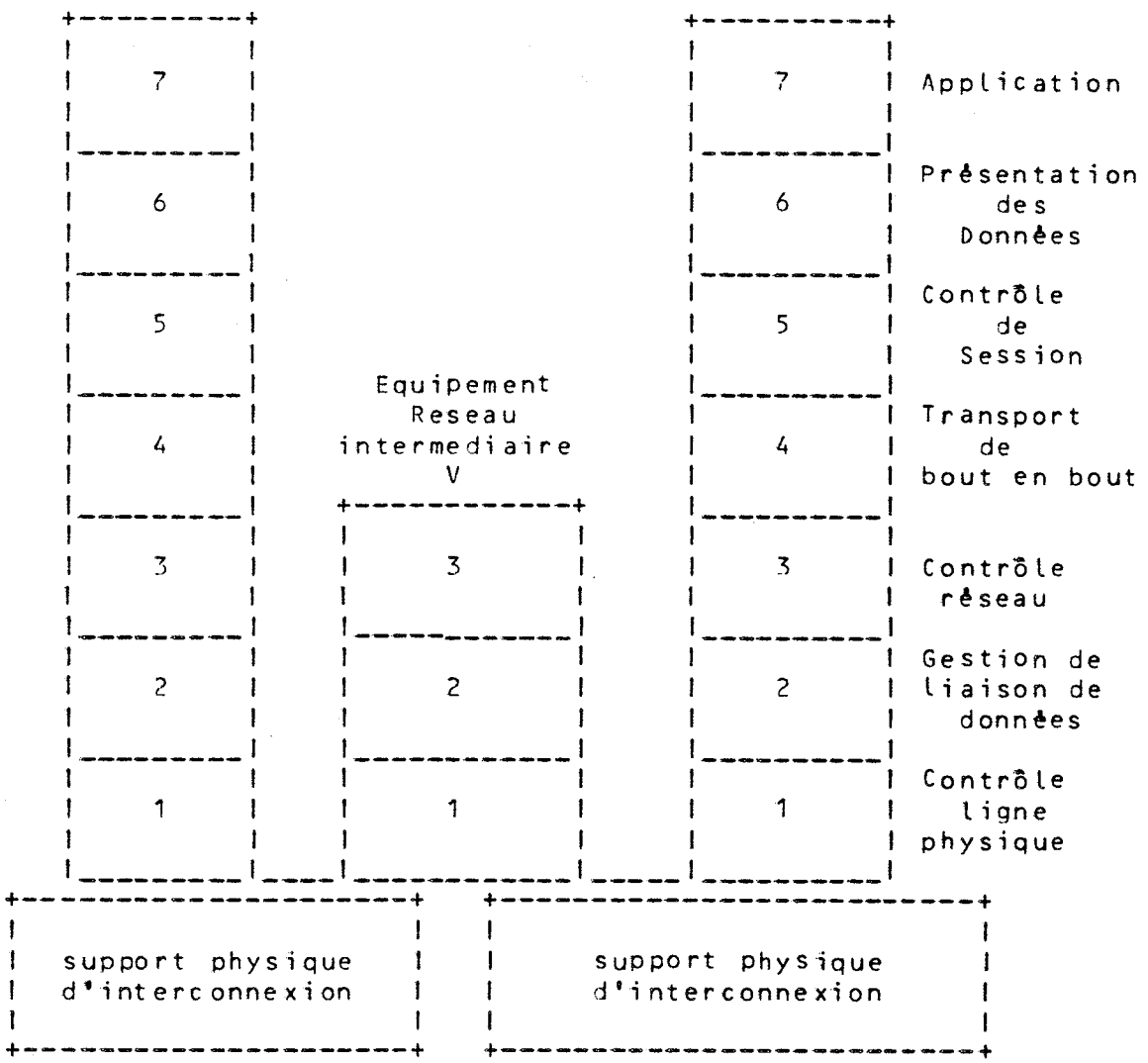


figure 3.1 : Les couches du modèle OSI

3.1.1/ Le niveau contrôle de la ligne physique

Il fournit les moyens mécaniques, électriques, procéduraux et fonctionnels pour l'établissement de connexions physiques entre deux extrémités d'une ligne de transmission. Cette organisation des éléments physiques identifie le circuit de données qui est donc le service offert par le niveau physique.

Les fonctions réalisées à ce niveau sont le codage électrique des données, l'utilisation des différentes techniques de transmission de bit (parallèle, série), les différents modes de transmission synchrone/asynchrone.

3.1.2/ Le niveau gestion de la liaison de données

Une liaison de données est une connexion logique entre deux entités de la couche supérieure, construite sur une ou plusieurs connexions physiques.
Ses fonctions sont d'assurer le transfert de l'information sur le circuit de données, la structuration des données et leur délimitation (caractères de commandes [stx...etx], transparence au code, etc...), l'identification de l'origine et de la destination, le contrôle de liaison de données (supervision, protection, reprise).

Le niveau gestion de la liaison de données fournit les moyens procéduraux et fonctionnels pour établir, gérer et terminer les liaisons de données. C'est à ce niveau qu'appartiennent les procédures de transmissions tels que HDLC ou X25.2. Ce que l'on retiendra surtout c'est le principe de l'utilisation de trame de format standard des procédures de transmission point à point, c'est-à-dire entre deux extrémités d'une connexion physique.

3.1.3/ Le niveau contrôle du réseau

La couche réseau fournit:
- d'une part les moyens d'établir, de maintenir et de libérer des connexions de réseau entre des systèmes contenant des entités d'application devant communiquer,
- d'autre part les moyens fonctionnels et procéduraux d'échanger entre entités de transport des unités de données du service de réseau sur des connexions de réseau.

Cette couche rend les entités du niveau supérieur (transport) indépendantes des problèmes de routage et de relais liés à l'établissement et au fonctionnement d'une connexion de réseau donnée. Elle masque aux entités de transport la façon dont les ressources des couches de niveau inférieur (connexion de données, par exemple) sont utilisées pour fournir des connexions de réseau.

Cette couche se voit attribuer un grand nombre de fonctions dont la tâche principale est de permettre la prise en compte d'une grande variété de configurations correspondant à des connexions de réseau. Les connexions de réseau prises en compte peuvent ainsi aller de celles correspondant à une configuration point-à-point, jusqu'à celles qui mettent en jeu des combinaisons complexes de sous-réseaux aux caractéristiques différentes.

La couche réseau assure le routage et la commutation des paquets sur le réseau de communication ainsi que le contrôle de congestion du réseau. Dans les réseaux généraux, on retrouve à ce niveau des fonctions qui sembleraient plutôt être du ressort de couches supérieures (gestion du réseau), telles que les fonctions de supervision du réseau de communication.

La fonction de routage, quant à elle, peut varier d'un réseau à l'autre et apporter ainsi de grandes différences dans l'utilisation du réseau et dans la qualité du service rendu par la couche réseau à la couche transport (routage fixe ou adaptatif, sur circuit ou en datagramme).

3.1.4/ Le niveau transport de bout en bout

Son rôle est d'assurer un transfert de données transparent entre les entités de niveau supérieur (session) en les déchargeant complètement des détails d'exécution d'un transfert de données et d'un bon rapport qualité/prix.

Les transferts de données se font au moyen des unités de données du service de transport de longueur quelconque échangées de bout en bout ("on rencontre souvent le terme de lettres"). On distingue deux façons de les réaliser:

- un transfert de lettre sans synchronisation initiale des deux correspondants et sans contrôle de la qualité de la transmission [mode non-connecté],
- un transfert de lettre sur une connexion de transport qui peut alors être exécuté avec un contrôle de flux (asservissement entre utilisateur sur le débit des lettres échangées), un contrôle d'erreur (récupération si possible des incidents de transport) et un contrôle du séquencement (conservation de l'ordre des lettres lors de la remise au récepteur) [mode connecté].

3.1.5/ Le niveau session

Il se charge d'organiser et de synchroniser les dialogues entre les entités de présentation coopérantes en leur offrant les services nécessaires à l'établissement de connexions de session et la prise en charge des interactions ordonnées lors des d'échanges de données.

Le niveau session s'appuie sur les connexions de transport pour établir les connexions de session.

3.1.6/ Le niveau présentation des données

Il se charge de la représentation des informations que s'échangent et auxquelles se réfèrent deux entités d'application au cours de leur dialogue.

Ce niveau intervient sous deux aspects complémentaires:
- celui de la syntaxe des données, qui correspond à la représentation des données transférées entre entités d'application,
- celui de l'image de présentation, qui correspond à la représentation, d'une part de la structure de données à laquelle des entités d'application se réfèrent au cours de leur dialogue, et d'autre part de l'ensemble des actions qui peuvent être effectuées sur cette structure de données.

Le niveau présentation fait en sorte qu'une représentation commune des informations soit utilisée entre les entités d'application. Ceci évite aux entités d'application d'avoir à se soucier elles-mêmes de ce problème. Cela permet de rendre les entités d'application indépendantes syntaxiquement. On peut ainsi considérer que:
- le niveau présentation fournit les éléments syntaxiques communs utilisés par les entités d'application,
- les entités d'application peuvent utiliser n'importe quelle syntaxe,

car le niveau présentation assure la transformation entre cette syntaxe et la syntaxe commune nécessaire à la communication entre entités d'application.

3.1.7/ Le niveau application

Il regroupe toutes les fonctions nécessaires à la communication entre systèmes ouverts qui ne sont pas réalisées par les niveaux inférieurs. Il sert de fenêtre entre les processus d'application qui correspondent à travers l'architecture OSI.

4/ Conclusions

Ce modèle d'architecture est grandement influencé par la nature des réseaux généraux de transmission de données et des applications très spécifiques qui y sont implantées.
Cependant, il faut bien admettre que ce sont les études de réseau qui ont conduit à la définition des systèmes téléinformatiques comme un ensemble d'équipements informatiques, reliés entre eux par des liaisons de données.
Les principales caractéristiques de toutes ces constructions sont:

- la relative lenteur des communications,
- l'absence de mémoire commune et l'absence de référentiel de temps commun.

Tout ceci prêche évidemment dans le sens d'une définition et d'une modélisation des systèmes qui soient relativement strictes et modulaires. A côté de ce découpage en niveau de service, il est une autre distinction qui s'impose de par la nature du sujet à réaliser: celle qui consiste à séparer autant que possible le traitement et le transport de l'information.
C'est cette raison qui fait que la plupart de ceux qui s'interresent à ce modèle reconnaissent deux grands groupes dans le modèle de décomposition en couche:

- les couches basses (physique, liaison de données, réseau),
- les couches hautes (application, présentation, session).

Ces couches s'articulent autour de la couche de transport qui constitue une véritable passerelle entre les moyens de traitement de l'information et les moyens de transport de l'information.

ANNEXE 2: STATISTIQUES DU PROTOCOLE C-S

ANNEXE 2

Dans cette annexe on trouvera le calcul des statistiques sur le protocole client serveur.

1/ Définition des probabilités calculées

Nous ne tenons compte que des pertes de message durant le transport. Nous établissons des formules littérales, ce qui nous permet ensuite de déterminer les résultats en fonction des caractéristiques de l'implantation.

On désigne par P1, la probabilité de bonne arrivée d'un message.

La probabilité de perte d'un message sera alors donnée par 1-P1.

Dans les paragraphes suivants ces deux probabilités seront utilisées pour calculer les probabilités d'arriver dans état donné du protocôle C-S.

Pour chaque type de transport on estimera les statistiques sur les pertes de messages, puis les incidences sur le protocôle (doublon chez le client et chez le serveur).

2/ Calcul des statistiques

2.1/ Statistiques du transport sans diagnostic

2.1.1/ Statistiques de transport

Dans ce type de transport il n'y a que les messages de demande et de réponse qui transitent pour le compte du protocôle C-S.

Sur la demande client

P(arrivée de la demande chez le serveur) = P1

P(perte de la demande) = 1-P1

Sur_la_réponse_serveur

La réponse du serveur ne peut être émise que si la demande est arrivée. La probabilité pour que le client ait une réponse est donc:

P(réponse chez le client) = P(arrivée de la demande)
x P(arrivée de la réponse)

= P1**2

La probabilité pour que le client n'ait pas de réponse est:

P(non réponse chez le client) = P(arrivée demande) x P(perte réponse)
+ P(perte de demande)
= P1(1-P1) + (1-P1)
= 1-P1**2

2.1.2/_Statistiques_sur_le_protocole_C-S

Abandon_du_protocole/_service_rendu

Cela est possible si le client ne tente pas de reprendre le dialogue C-S après échéance de la temporisation. On a alors:

P(abandon du client) = P(arrivée demande) x P(perte réponse)

= P1(1-P1)

Exemple:

Si l'on suppose un taux de perte de 1 message sur 1000, P1 = 0.999
On a alors:

P(abandon du client) = 0.999 x (1-0.999) = Environ 10**(-3)

Doublement_des_messages

Avec ce type transport cela n'est possible que pour la demande et seulement si le client la reémet après échéance de la temporisation. On a alors:

P(doublon de demande) = P(arrivée demande)
x P(perte réponse)
x P(arrivée deuxième demande)

soit:

P(doublon demande) = P1(1-P1)P1 = (P1**2)(1-P1).

Exemple:
Toujours avec P1 = 0.999:

P(doublon demande) = environ 10**(-3)

2.2/ Statistiques_transport_avec_diagnostic_négatif

Dans ce mode de transport les erreurs de transport provoquent l'émission d'un message supplémentaire que nous désignons par NACK(message ayant subit l'erreur). Les messages du protocole C-S sont donc:

- la demande, le Nack(demande), la réponse, le Nack(réponse).

2.2.1/ Statistiques_sur_le_transport

Demande_client

P(arrivée de la demande) = P1

P(perte de la demande) = 1-P1

Il faut y ajouter les statistiques sur le NACK(demande) La probabilité pour que le client reçoive un Nack(demande) est liée à la perte de cette demande et au bon acheminement de ce Nack. On a donc:

P(Nack(demande)) = P(perte demande) x P(reception Nack).

= (1-P1)P1

P(perte Nack(demande)) = P(perte demande) x P(perte NacK),

= (1-P1)(1-P1)

= (1-P1)**2

Sur_la_réponse

P(réception réponse) = P1**2

P(perte réponse) = 1-P1**2

Nous y ajoutons les statistiques sur le Nack(réponse). Ce cas correspond à la réception de la demande, suivie de l'émission de la réponse qui est perdue lors du transport, et enfin de la réception du Nack(réponse) par le serveur.

P(Nack Réponse) = P(arrivée demande) x P(perte réponse)

x P(arrivée Nack(réponse))

= (1-P1)P1**2

Dans le cas suivant le serveur n'est pas averti de la perte de la réponse.

P(pas de Nack réponse) = P(arrivée demande)
x P(perte réponse)
x P(perte Nack(réponse))

= P1(1-P1)**2

2.2.2/ Statistiques sur le protocôle C=S

Abandon définitif du client

On considère toujours le cas où le service a été réalisé mais le client n'en a jamais été averti et il n'a pas reémit sa demande.

P(abandon client) = P(perte Nack(réponse))
= (1-P1)P1**2

Exemple:

avec P1 = 0.999:

P(abandon client) = 0.98*10**(-3)

Doublement de messages

Le client peut reémettre une demande si la réponse ne lui parvient pas. Cela se produit dans le cas où la réponse et le Nack(réponse) se sont perdus.

P(doublement demande) = P(perte Nack(réponse))
x P(arrivée demande no 2)

= (1-P1)P1**3

Exemple:

P(doublement demande) = 0.9*10**(-3)

2.3/ Statistiques transport avec diagnostic positif

2.3.1/ Statistiques de transport

Les probabilités se calculent comme dans les cas précédents ce qui nous permet de donner tout de suite leurs valeurs.

Réception de l'ack demande

P(reception Ack(demande)) = P(arrivée demande)
x P(arrivée Ack(demande))

= P1**2

Non réception de l'Ack(demande)

P(non Ack(demande)) = P(perte demande)
+ [P(arrivée demande) x P(perte Ack(demande))]

= (1-P1) + P1(1-P1) = 1-(P1**2)

Réception de la réponse

P(réception réponse) = P(arrivée demande) x P(arrivée réponse)
= P1 x P1= P1**2

réception de l'Ack(réponse)

Si le serveur ne recoit pas l'Ack(réponse), il peut reémettre la réponse et provoquer ainsi un doublon chez le client.

P(non Ack(réponse) = [P(arrivée demande)
x P(arrivée réponse)
x P(perte Ack(réponse))]
+ [P(arrivée demande) x P(perte réponse)]

= (P1**2)(1-P1) + P1(1-P1)

= P1(1-P1**2)

2.3.2/ Statistiques sur le déroulement de C=S

Abandon du protocole

Il s'agit toujours de l'éventualité selon laquelle le client ne recoit jamais la réponse.

Cela ne peut arriver que si la réponse se perd de façon réitérée, c'est-à-dire que le serveur est obligé de reémettre plusieurs fois la réponse sans succès. Dans tous les autre cas il y redondance d'information entre le message de réponse et son Ack.
Si le serveur reémet N fois la réponse, la probabilité pour que le client abandonne avec un service réalisé est égale à : P1**N. On voit donc que cette probabilité peut être rendue très faible.

Doublement des messages

Dans ce type de transport le client et le serveur peuvent recevoir des doublons.

1) Les doublons de la demande se produisent si le Ack(demande) se perd. On a donc:

P(doublon demande) = P(arrivée demande)

x P(perte Ack(demande))

x P(arivée demande no 2)

= P1**2(1-P1)

Exemple:

P(doublons demande) = 10**(-3)

2) Les doublons des réponses du serveur:

P(doublons réponse) = P(non Ack(réponse))

Exemple:

P(doublon réponse) = P1(1-P1**2) = 2x10**(-3)

ANNEXE 3

ANNEXE 3

1/ Mécanismes de base

Le modèle de communication que nous avons décrit sert à spécifier le service de communication et son utilisation par les acteurs utilisateurs. Il présente néanmoins quelques imprécisions (voulues) quant à la façon d'implanter, par exemple, les portes du service de communication et d'échanger des unités de données entre elles.

Cela permet de laisser au concepteur d'une application le choix de l'implantation tout en respectant les principes de base.

A titre d'exemple, nous donnons ici les choix de l'implantation du noyau et des acteurs du service de communication sur une machine mono-processeur à base d'INTEL 8086.

L'implantation de Chorus sur INTEL 8086 avec la pascal UCSD (choix des structures de programmes, variables Chorus, acteurs systèmes et procédures d'interfaces, etc...) a été largement décrite dans une thèse [GUI 82].

C'est dans le cadre de cette implantation que nous avons réalisé les services de communication du noyau et les acteurs de communication. Dont nous ne décrivons ici que les caractéristiques propres à:

- l'implantation dans le noyau des fonctions du service de communication qu'il réalise,
- aux spécifications et à l'implantation de l'acteur de commutation distante,
- aux spécifications et à l'implantation de l'acteur de transport.

Nous terminons par une brève description de la structure des acteurs utilisateurs.

1.1/ Conventions

Les portes sont sédentaires, ce qui simplifie leur désignation. On prend le parti de désigner les portes du noyau et les portes du système connues de tous les acteurs par; :NOMPORTE. Les autres portes sont désignées par le nom de site concaténé au nom de porte [X]NOMPORTE.

Les algorithmes sont donnés en français et structurés comme le langage Pascal.

Nous ne donnons ici que les procédures d'interface propres au service de transport sans redéfinir toutes les conventions de construction des noms de variables.

1.2/ Le noyau Chorus

Le noyau Chorus contient toutes les fonctions logiques qui sont dépendantes des caractéristiques physiques de la machine. Il est perçu par les acteurs de son environnement comme un ensemble de services accessibles par des messages.

1.2.1/ Communication acteur/noyau

1.2.1.1/ Les messages et la primitive RETURN

L'acteur et le noyau communiquent par message. Les messages sont préparés par l'acteur au cours de l'étape de traitement. Ils permettent d'adresser tous les services du noyau à savoir:

- la sélection,
- l'aiguillage,
- la temporisation,
- le transport local,
- le transfert local,

Nous rappelons que la distinction entre transfert local et transport local provient du fait que l'on distingue le service du noyau rendu à un acteur du service de communication de celui rendu à un acteur utilisateur.

L'acteur signale la fin de l'étape de traitement par la primitive RETURN qui est l'unique primitive de communication car elle marque la fin de la phase de prépraration des messages et signale au noyau qu'il peut effectivement tenir compte des messages préparés par l'acteur. L'exécution revient alors au noyau qui commence à traiter les messages de l'acteur.

1.2.1.2/ Interface de programmation

On peut considérer que la nouveauté (dans la programmation de l'acteur) vient de la nécessité de communiquer par message et donc de construire des messages.
Certains d'entre eux ont une syntaxe standard définie par les règles d'accès aux services du système. Dans ce cas, pour aider à la programmation de l'acteur et faciliter la création des messages, le système Chorus offre un ensemble de procédures qui sont partagées par le noyau et les acteurs. Ces procédures, appelées procédures d'interface, mettent en forme les messages standards.

Dans l'implantation sur 8086, chaque procédure d'interface constitue, dans la zone dynamique de l'acteur, un message qui représente une "demande de service" et le place dans une liste qui est, du point de vue du transport, la file d'attente d'une porte spécifique du noyau.

1.2.1.3/ Les portes du noyau

Les portes du noyau identifient les points d'accès aux services du noyau. Elle permettent à ce dernier de recevoir sélectivement les différentes demandes d'un acteur.

Outre les portes de SELECTION, D'AIGUILLAGE, de TEMPORISATION, le noyau possède donc les portes qui reçoivent les messages destinés à être acheminés par le service de communication.

Les deux types de communication réalisés par le noyau (transfert, transport) sont identifiés par deux portes distinctes.

Echanges acteur utilisateur/noyau

La porte :PC_TRANSP ("TRANSPORTER MESSAGE") du noyau reçoit les messages préparés par la procédure d'interface (cf SEND) associée à une demande de transport.

Echanges acteur de communication/noyau

Un acteur participant au service de communication demande le transfert physique d'une unité de données depuis son espace de travail vers une file d'attente d'une autre porte du système. Dans ce cas la procédure d'interface associée à ce type de service noyau prépare un message pour la porte :PC_TRANSF ("TRANSFERER MESSAGE").

1.2.2/ Organisation du noyau

1.2.2.1/ Le programme Noyau

Le noyau est écrit comme un programme Pascal UCSD normal. Schématiquement, il a la structure suivante:

```
Initilisations
FAIRE INDEFINIMENT
 lecture des messages de service
 traitement des messages de service
 cadencement
 passage à un acteur
FIN
```

La lecture des messages correspond à la prise en compte par le noyau des messages de demande de service générés pendant l'étape de traitement d'un acteur.

1.2.2.2/ Organisation interne

Dans le noyau les services d'un niveau Chorus donné sont réalisés par un ensemble de procédures qui consomment les messages de demande. Dans l'implantation décrite, les portes du noyau qui identifient les services visibles depuis les acteurs sont implantées sous la forme de listes que le noyau parcoure après chaque étape de traitement.
On a donc:

- une liste pour les messages à transporter (traitée par la procédure IRANSPORI du noyau),
- une liste pour les messages à transferer (traitée par la procédure IRANSFERI du noyau).

Le noyau réalise toutes les opérations de transport local, il possède donc toutes les portes du service de communication local décrites dans

le modèle: nous avons déja cité la porte de transport local et la porte de transfert local.
Il ne manque que la porte "COMMUTER MESSAGE" qui n'est pas connue des acteurs. Elle n'est donc pas accessible par une liste et elle est implantée dans le noyau sous la forme d'une procédure de commutation COMMUTER appelée par la procédure TRANSPORT.

On peut, schématiquement, représenter le fonctionnement du noyau comme suit (en distinguant l'exécution d'un acteur utilisateur de celle d'un acteur de communication):

Fin_d'éxécution_d'un_acteur_utilisateur

```
 ! Acteur en cours !
         !
         V
       1) Exécution de l'acteur utilisateur
      2)dépot message sur porte transport
       3) RETURN
         !
         V
 ! Activation du noyau !
         !
         V
   (* Début de la communication dans le noyau *)
      4) TANT QUE file de transport non vide FAIRE
         DEBUT
            5) appel de la procédure TRANSPORT
            6) appel de la procédure COMMUTER
            7) appel de la procédure TRANSFERT
         FIN du tant que file transport non vide
```

(* Fin de la communication dans le noyau *)

I

V

Fin_d'éxécution_d'un_acteur_du_service_de_communication

I Acteur communication en cours I

I

V

1) Exécution de l'acteur de communication

2) préparation d'un message pour le transfert

RETURN

I

V

I Activation du noyau

I

V

(* Début de la communication dans le noyau *)

1) TANT QUE file de transfert non vide FAIRE

DEBUT

2) appel de la procédure TRANSFERT

FIN du tant que

(* Fin du transfert par le noyau *)

1.3/ Les unités de données

1.3.1/ Unités de l'interface Utilisateur/Noyau

Dans l'implantation sur Intel 8086, le transport de base est en datagramme sans diagnostics, les unités de données sont de taille fixe. Cela nous conduit à avoir une unité de données de l'interface Utilisateur/Noyau très réduite.

(Pr,Pe,Texte)

Pr = porte réceptrice de texte,

Pe = porte émettrice de texte,

Texte = données de l'utilisateur.

1.3.2/ Les unités du service de communication

Le noyau, au moyen de la procédure transport, constitue l'unité de données du service de transport local qui est exactement le bloc fourni par l'interface Utilisateur/Noyau c'est-à-dire /Pr,pe,Texte/. Le "Message" n'est pas modifié par le noyau si le transport et la commutation restent locaux. S'il doit être transporté à distance il est déposé "tel quel" dans la file d'attente de la porte par défaut (porte de transport distant).

Remarque:

Dans l'implantation sur Intel 8086, la porte par défaut reçoit les messages à transporter à distance sans que ceux-ci lui soient réellement destinés. C'est-à-dire que la porte réceptrice stipulée dans les messages reçus sur cette porte est la porte distante et non la porte par défaut. Ce choix d'implantation facilite le travail du noyau (qui n'a pas à modifier le message pour préciser la porte réceptrice réelle. Il ne fait pas, non plus, les contrôles de cohérences entre nom de porte réceptrice et file d'attente sur

laquelle est déposée le message). Dans une implantation où aucune dérogation aux règles de contrôles ne serait tolérée, le noyau qui ne peut commuter localement un message devrait constituer une unité de données de l'interface transport local/transport distant constituée comme suit:

:PC_DEFAUT, :PC_TRANSP,(Pr,Pe,Texte)

:PC_DEFAUT = porte de transport par défaut (ce nom est initialisé par l'acteur de transport qui possède la porte par défaut (porte réceptrice),

:PC_TRANSP = porte de transport du noyau (porte emettrice).

1.4/ Les procédures d'interface

1.4.1/ La procédure SEND

Dans le transport de base, les acteurs utilisateurs ne disposent que de la procédure SEND pour préparer des unités de données à transporter.

```
PROCEDURE SEND (port_srce, port_rcpt : TC_NOM;
                Var texte : TC_MESSAGE);
```

port_srce = porte émettrice (de l'utilisateur initial) du message,
port_rcpt = porte réceptrice (de l'utilisateur final) du message,
texte = données de l'utilisateur.

La procédure SEND construit le message par concaténation du nom de la porte réceptrice, du nom de la porte émettrice et du texte. Elle dépose le message ainsi constitué dans la file des messages en attente de transport dans l'espace dynamique de l'acteur (nous avons déjà

expliqué pour qu'elles raisons la porte du noyau qui reçoit le message n'est pas indiquée dans le message lui-même).

1.4.2/ La procédure TRANSFERER

Cette procédure n'est accessible que par les acteurs de communication.

```
PROCEDURE TRANSFERER (port_rcpt : TC_NOM;
                      VAR message : TC_MESSAGE);
```

port_rcpt = nom de la porte dont la file d'attente doit recevoir le message,

message = unité de données du service de communication qui doit être transférée.

Cette procédure copie le texte des données transmises ainsi que le nom de la porte émettrice dans la file d'attente associée à port_rcpt.

Remarque:

Dans le cas de l'implantation la plus simple avec un transport local sans diagnostic et sans contrôles de transport, cette procédure est équivalente à la procédure SEND.

1.5/ Gestion de la mémoire

A l'émission

Les messages utilisateurs sont préparés dans l'espace de données de l'utilisateur et recopiés au moment de RETURN dans un tampon de message du noyau. Les tampons sont de longueur fixe. Si la mémoire est saturée, le message n'est pas pris en compte et l'on considère cet incident comme une perte de message durant le transport.

L'implantation sur Intel 8086 tendait d'abord à prouver la faisabilité de Chorus. Le nombre de messages disponibles est largement dimensionné pour le type d'application que nous avons réalisé sur cette première machine, ce qui a permis d'éviter les problèmes d'allocation de ressources à l'émission.

A la remise

Les messages sont chaînés par le noyau à la file d'attente de la porte réceptrice. On peut donc considérer que les files d'attente sont infinies. Cependant, nous avons trouvé utile de définir deux types de files.

- les files de longueur 1, pour lesquelles tout nouveau message écrase le précédent (information répétitive donc seule la dernière occurence est significative : ACK Danube, par exemple),
- les files de longueur supérieure à 1, pour lesquelles la longueur de la file est infinie (il faut cependant veiller à la consommation des messages par l'acteur c'est-à-dire à son activation).

1.6/ Conclusions

Les renseignements fournis ici permettent surtout de voir que la rigidité et les complications du modèle ne sont qu'apparentes si l'on regarde la simplicité des moyens mis en oeuvre pour réaliser la communication dans le noyau. Malgré les quelques simplifications faites à l'implantation, nous voyons que le principe des portes et des couches distinctes peut être respecté, même dans une organisation centralisée comme celle du noyau.

La description que nous venons de faire apporte suffisamment d'éléments pour décrire le comportement d'un acteur vis-à-vis des communications. Les acteurs qui participent au service de

communication seront donc des acteurs comme les autres, à la différence près qu'ils n'invoquent pas eux même le transport et disposent donc à cet effet d'une procédure d'interface particulière avec le noyau (TRANSFERT).

2/ Les acteurs du service de communication

Les acteurs du service de communication réalisent toutes les opérations de communication qui ne sont pas exécutées par le noyau.

Nous distinguons volontairement un acteur de commutation et un acteur de transport pour l'étude de l'implantation afin de clarifier l'exposé. Dans l'implantation, pour limiter les échanges de messages et les basculements de contexte, on peut regrouper toutes les fonctions décrites dans un seul et même acteur de transport.

2.1/ La commutation distante

L'acteur de commutation distante, sur Intel 8086, interface la machine avec un réseau à diffusion Danube, réalisé à l'INRIA par le projet pilote Kayak.

L'acteur de commutation distante est un acteur Chorus spécialisé qui dispose d'une interface privilégiée avec le réseau. Nous sommes donc confrontés ici au double problème de la programmation d'un acteur (en utilisant les primitives et les messages de communication avec le noyau) et de la gestion, dans un programme Pascal, des communications avec un coupleur réseau.

Nous commençons par la description de la voie de communication Danube, puis nous décrivons l'acteur de communication.

2.1.1/ Description de la voie de communication

2.1.1.1/ Le réseau local Danube

Le réseau Danube permet le raccordement d'une machine par l'intermédiaire d'un "communicateur". Ce communicateur offre du point de vue physique, aux machines connectables, une interface asynchrone série de type V24. Cette interface permet des vitesses de transfert allant jusqu'à 19200 bauds.

Les communicateurs offrent, du point de vue logique, les services d'une boite "intelligente" qui leur permet de voir la machine raccordée:

- soit comme un terminal,
- soit comme un ordinateur.

Il ont enfin la possibilité de fontionner de façon transparente aux échanges sur le réseau.

Nous ne présentons ici que la version transparente qui est la seule fonction retenue pour notre application.

2.1.1.2/ La version transparente

La boîte "intelligente", en version transparente, gère les communications avec le calculateur connecté grâce à un protocole d'échange d'unités de données.

La boîte intelligente émet, vers le réseau, les paquets qui lui sont remis par le calculateur et remet au calculateur les paquets qu'elle reçoit du réseau. La boîte intelligente ne se préoccupe que des adresses réseau de l'émetteur et du récepteur du paquet. Elle effectue un filtrage des messages reçus du réseau grâce à une table de noms logiques qu'elle maintient. Ces noms sont déclarés par le calculateur. Dans Chorus, ces noms logiques sont donnés par l'acteur de commutation pour identifier et déclarer son propre site et

éventuellement d'autres adresses logiques de son site.
Une machine raccordée peut donc être identifiée avec plusieurs noms (groupes de calculateurs).

2.1.1.3/ Unités de données de la boîte Danube

Les unités de données échangées entre le communicateur et la machine sont des blocs d'octets délimités par le caractère de début <STX> et le caractère de fin <ETX>.

- Chaque échange d'unité de données entre le communicateur et la machine est coordonné par l'émission d'un <ACK> par le partenaire qui est prêt à accepter une nouvelle unité.
- Chaque unité de données est composée d'un octet de fonction suivi d'un bloc de données de longueur variable.

Quatre fonctions sont définies.

- l'une qui permet d'échanger un paquet réseau entre la boîte et le calculateur,
- les trois autres qui sont des commandes de mise à jour de la table des noms (remise à blanc de la table, ajout d'un nom, suppression d'un nom).

2.1.1.4/ Communications avec la boite DANUBE

Le protocole de communication entre la boîte Danube et la machine règle les échanges de messages entre l'ordinateur et le réseau de façon très élémentaire et permet:

-le transfert bi-directionnel de "messages',
-l'émission de message sans anticipation,
-l'acquittement message par message,

Initialisation du dialogue

L'ordinateur initialise la boîte. A sa mise sous tension, celle-ci émet des acquittements (ACK) jusqu'a ce que l'ordinateur envoie un ACK.

La boîte répond alors par une séquence de RESET (<stx><2><etx>) qui prévient l'ordinateur qu'elle est prête a communiquer.

A partir de ce moment:

- lorsque la boite émet un ACK, cela signifie qu'elle est prête a recevoir un message et un seul (venant de l'ordinateur).
- lorsque l'ordinateur émet un ACK, cela signifie qu'il est prêt a recevoir un message et un seul venant de la boite.

Calculateur		Boîte DANUBE
Début initialisation		
	<--- ACK -----	
	<--- ACK -----	
	---- ACK ---->	
	<-- RESET ----	
Fin initialisation		
dialogue normal		

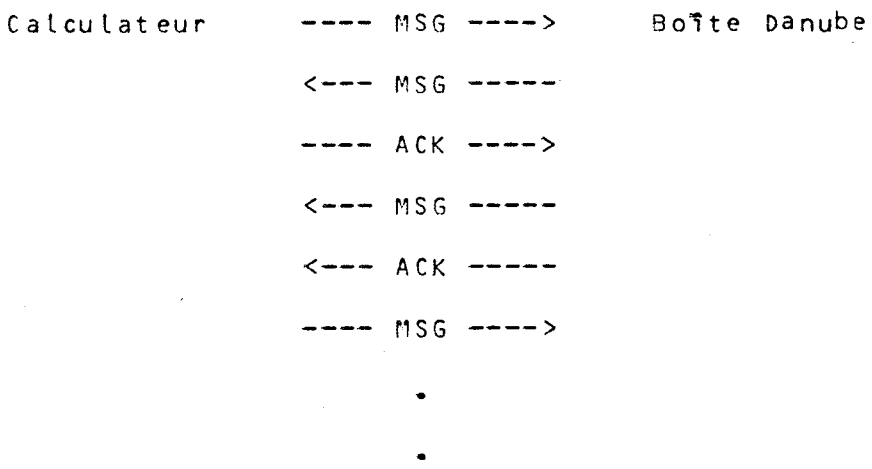


figure 2.1 : Protocole Boîte DANUBE - Calculateur

Remarque

Chaque ACK ne donne pas forcément le droit d'émettre un message parce qu'il annule le précédent s'il n'y a pas eu d'émission d'un message entre temps.

2.1.1.5/ Commande d'envoi de message

Les messages circulant sur le réseau Danube sont de taille variable avec un maximum fixe a 255 octets. Le format du message envoyé par l'ordinateur a la boite doit avoir le format suivant:

<stx><0><text><etx>

<stx> = octet de début de message,

<0> = code caractérisant un paquet (pour la boite Danube),

<text>= message a envoyer sur Danube,

<etx> = octet de fin de texte.

2.1.1.6/ Conclusions sur la voie de communication

Le dialogue avec le réseau par l'intermédiaire de la boîte se fait comme une sur ligne de transmission c'est-à-dire en utilisant un protocole qui fonctionne comme une procédure de transmission sans anticipation, avec acquittement sélectif des blocs de données.
L'acteur de commutation devra donc attendre la réception d'un ACK de la boîte pour émettre un message Chorus vers le réseau.
L'acteur de commutation cadence les réceptions des messages venant du réseau grâce à l'envoi des ACKs vers la boîte DANUBE. Nous passons maintenant à la description de l'acteur de commutation distante.

2.1.2/ L'acteur de commutation distante

L'acteur de commutation distante envoie sur le réseau les messages recus sur la porte "COMMUTER A DISTANCE". Ces derniers sont appelés "messages Chorus" par opposition aux messages qui circulent sur le réseau qui sont appelés messages Danube. L'acteur de commutation distante coopère avec les acteurs de commutation des autres sites pour acheminer à travers le réseau les messages Chorus qui sont contenus dans des messages Danube.

2.1.2.1/ Réception des messages de Danube

La réception des messages Danube est faite par une procédure de l'interpréteur qui remplit, caractère par caractère, un tampon initialisé par l'acteur de commutation. Le début du message est reconnu par le caractère <stx>, la fin de "message" est reconnue par la réception d'un <etx> qui provoque une interruption CHORUS détectée par le noyau. Ce dernier dépose un message d'IT [arrivée d'un message Danube] dans la file de la porte de l'acteur de commutation associée a l'IT Danube. Lorsque l'acteur de commutation est activé pour traité

ce message d'IT, il lit le tampon qui contient le message recu.
Un message d'interruption a la structure suivante:

{ :INTERR,Pr,i }

:INTERR = porte émettrice des messages d'interruption,

Pr = porte réceptrice du message d'interruption,

i = numéro de l'interruption logique.

L'acteur de commutation traite les interruptions suivantes:

-Réception d'un message de DANUBE,

-Réception d'un ACK de la boite DANUBE.

2.1.2.2/ Unités de données de commutation distante

Les messages Danube sont échangés en mode datagramme sans diagnostic entre les sites. Chaque acteur de commutation est identifié sur le réseau DANUBE par un nom logique déclaré par lui dans la boite (pour la filtrage).

L'acteur de commutation distante indique dans l'unité de données du service de commutation distante transmise à la boîte DANUBE le nom logique de l'acteur récepteur. Ce nom logique est actuellement le nom du site Chorus qui abrite la porte réceptrice du message Chorus (On peut concevoir une implantation dans laquelle les noms logiques déclarés à la boîte seraient les noms des portes ouvertes sur la machine raccordée, jusqu'à concurrence de la taille maximale de la tables. Cette solution facilerait le routage des messages, dans le cas des portes migrantes, par exemple).

L'acteur de commutation transmet à la boite le "message Danube" à diffusé à toutes les boîtes. C'est la boîte qui complète le message par l'adresse physique de la source.

```
  1 octet   adresse physique du communicateur récepteur
  1 octet   adresse physique du communicateur émetteur
  1 octet   longueur du paquet (qui suit)

---------- début du paquet ------------

  1 octet   longueur du nom du site récepteur
 10 octets  Nom du site récepteur
  1 octet   longueur du nom du site émetteur
 10 octets  Nom du site émetteur

---------- suite du paquet--------------

 variable   texte du paquet
```

figure 2.2 : Format du message Danube

Le "texte du paquet" contient le message Chorus recu sur la porte COMMUTER A DISTANCE, ainsi que des données éventuelles du protocole entre les acteurs de commutation distante. Dans l'implantation décrite le protocole est en datagramme sans diagnostic. Il n'y a donc pas de données du protocole de commutation distante.

2.1.2.3/ Interface de commande

En dehors des messages Chorus et des messages reçus de Danube, l'acteur de commutation distante reçoit des messages "de contrôle" liés à l'activité de commutation distante.

Les messages destinés a l'acteur de commutation sont:

- le message initial qui sert à lancer l'acteur après sa création,
- les messages qui permettent de gérer dynamiquement la table des noms logiques dans la boîte. Ces messages demandent à l'acteur de commutation de déclarer ou d'enlever un nom dans la boîte DANUBE.

2.1.2.3.1/ Les portes

L'acteur de commutation possède des portes, connues du noyau et de l'acteur de transport.

Porte connue par le transport et le noyau

:DANUBE = porte de réception des messages à envoyer sur DANUBE (porte par defaut),

Porte connue par le noyau

:DANMES = porte de réception des messages d'IT message DANUBE,

:DANACK = porte de réception des messages d'IT acquittement DANUBE,

Pomb = porte ombilicale : recoit le message initial. Ce nom est donné par le gestionnaire des portes a la création de l'acteur.

Autre Porte

:DANOUV = porte de réception des messages d'ouverture de noms (on considère que la gestion de la table des noms logiques de la boîte est dynamique et chaque modification est demandée par un message adressé à l'acteur de commutation distante).

2.1.2.4/ Structure de L'A.C.D

2.1.2.4.1/ Algorithme de l'acteur réseau

L'acteur réseau, comme tous les acteurs de Chorus est organisé en étapes de traitement distinctes et c'est ainsi que nous allons le décrire. Le principe général de cet acteur est le suivant.

(* Initialisation *)

Phase 1 ...

```
Ouvrir (PS/PA_OUV_POR) les portes de l'acteur;

Initialiser le dialogue avec la boîte Danube;
      Envoyer ACK;
      Attendre Reset Danube (* c'est un message *);

Phase 2 ...

Envoyer les messages d'ouverture de nom vers
    la porte Danouv (un nom de site spécifique
                + un nom de diffusion);

Declarer (SWITCH) les aiguillages;

Poser (SELECT) les premières Sélections;

(* Fin d'initialisation *)

(* +-------------------------------------------------------+ *)

(* En régime de "croisière" *)

Traiter les messages reçus sur les différentes portes
et aiguillés vers les différentes étapes de traitement;
```

2.1.2.4.2/ Les étapes de traitement

Avant de passer à la description des étapes de traitement, nous allons nous pencher sur la technique employée pour la gestion des portes de l'acteur qui est particulière afin de respecter le protocole de communication avec la boîte. En effet, dans ce protocole, un message ne peut être émis vers Danube sans qu'un ACK ait été au préalable reçu de la boîte. Les messages reçus sur la porte Danube ne peuvent donc être émis que lorsque l'acteur de commutation a reçu un

ACK sur la porte Danmes. Or, pour Chorus, l'acteur est "activable" dès qu'il recoit un message sauf indication de sélection contraire.

On utilise donc les règles de sélection pour que le traitement associé à la porte Danube ne soit déclenché que dans le cas où la boîte sera effectivement prête à recevoir un message venant du calculateur.

L'acteur de commutation ne sélectionne donc la porte :DANUBE que s'il a recu un ACK de la boîte. Ainsi le noyau ne lancera-t-il le traitement sur cette porte que lorsque le message Chorus pourra être émis vers Danube.

Il est possible de dégager deux intérêts particuliers à cette technique:

- l'acteur de commutation utilise la gestion de file d'attente du noyau pour stocker les messages Chorus en attente d'envoi sur le réseau.

- l'acteur de commutation ne vérifie pas s'il peut émettre ou non sur le réseau. Lorsqu'il est réveillé sur la porte DANUBE, il peut émettre un message Chorus en attente sur cette porte.

On distingue 7 étapes de traitement pour l'acteur de commutation.

Iraitement_No_1

Le traitement No 1 est le traitement du message initial. Il s'agit de:

- l'ouverture des portes :DANMES, :DANACK,
- l'initialisation du dialogue avec la boite Danube,
- la mise en attente du message d'initialisation venant de DANUBE. Cette opération est réalisée par l'aiguillage de la porte DANMES vers le traitement No 2.

Traitement No 2

Traitement principal

Le traitement No 2 consiste en la réception du Reset Danube. Cette étape est également utilisée pour ouvrir les portes :DANUBE et :DANOUV et pour sélectionner la porte :DANMES.

Messages envoyés

Les messages envoyés dans cette étape de traitement concerne la déclaration des noms de site. Ils sont envoyés directement sur la porte Danouv.

Les aiguillages

L'acteur de commutation aiguille les portes de la façon suivante:

danmes --> traitement 3,

danack --> traitement 4,

danube --> traitement 5,

danouv --> traitement 6,

Les sélections

Les portes sélectionnées à la fin de cette étape pour la suivante sont:

:DANACK, :DANMES, :DANUBE et :DANOUV.

Traitement No 3

Traitement principal

Le traitement no 3 décode le paquet reçu de Danube. Il s'agit d'identifier le type du message du protocole calculateur/boîte (il est en effet possible de recevoir; soit des messages de Reset Danube, soit des paquets réseau). S'il s'agit du reset, il faut redéclarer les noms à la boîte Danube. S'il s'agit d'un paquet Réseau, l'acteur décode le paquet Danube et identifient les portes "émettrice et réceptrice".
Dans les deux cas, après avoir pris connaissance du contenu du tampon Danube, l'acteur de commutation le réinitialise.

Messages envoyés

S'il s'agit de la réinitilisation, les messages d'ouverture de nom sont envoyés sur la porte Danouv.
S'il s'agit d'un paquet Danube, le message utilisateur, qui a été extrait de ce paquet, est émis par l'acteur réseau vers la porte réceptrice locale au nom de la porte émettrice distante.

Les Sélections

Les portes sélectionnées sont :DANACK, :DANMES. Si l'acteur à le droit d'émettre vers Danube (drapeau droit_emis = 1[cf traitement no 4]) les portes :DANOUV et :DANUBE sont aussi sélectionnées.

Les aiguillages ne sont pas modifiés par cette étape de traitement.

Traitement_No_4

Le traitement no 4 est extrêmement court. Il s'agit de prendre en compte les ACKs de la boîte qui donne le droit d'émettre un message (si cela n'était pas encore le cas). Dans cette étape de traitement on met le drapeau droit_emis à 1.

Traitement_No_5

Traitements_principaux

Le traitement no 5 traite les paquets à envoyer vers Danube. L'acteur réseau construit un paquet Danube dans lequel il place le message Chorus à commuter.

Les_messages_envoyés

Il n'y a pas de message Chorus envoyé au cours de cette étape de traitement. Le paquet réseau Danube n'est pas émis comme un message Chorus.

Les_Sélections

Dans la mesure où il a émis un paquet Danube, l'acteur de commutation perd son droit d'émettre (==> droit_emis = 0). Les Sélections portent sur les portes :DANACK, :DANMES.

Les aiguillages ne sont pas modifiés par cette étape de traitement.

Traitement_No_6

Traitement_principal

Le traitement no 6 traite les messages de gestion de la table des noms de la boite Danube (ouverture, fermeture, remise à zéro).

Messages_envoyés

Il n'y a pas de message Chorus envoyé pendant cette étape de traitement. L'acteur de commutation émet un paquet de commande Danube vers la boîte.

Les_Sélections

L'acteur de commutation vient d'émettre un message vers la boîte Danube (==> droit_emis = 0). Les portes qui restent sélectables sont donc :DANMES, :DANACK.

Les aiguillages ne sont pas modifiés par cette étape de traitement.

2.1.2.4.3/_Les_priorités_des_portes

L'enchaînement des étapes de traitement de l'acteur dépend non seulement de la présence éventuelle de messages sur les files d'attente de ses portes, mais encore de l'ordre dans lequel le noyau va lancer l'acteur pour traiter ces messages. La priorité des portes joue un rôle essentiel dans le rendement optimal de l'acteur de commutation.

Il y a un certain nombre de portes dont la priorité n'est pas fondamentale pour le bon fonctionnement de l'acteur. Ce sont :
-la porte ombilicale Pomb,

-la porte d'ouverture de nom,

-la porte de debug,

En revanche les portes :DANACK, :DANUBE, :DANMES doivent être activées dans un ordre qui dépend de la direction (émission ou réception) que l'on veut priviligier.

Si l'on privilégie la réception, par priorités décroissantes on a l'ordre d'activation suivant; :DANMES, :DANACK, :DANUBE.

Si l'on privilégie l'émission, dans l'ordre de priorité décroissante on a :DANACK, :DANUBE, :DANMES. Il y a cependant une restriction, dans ce cas, aux possibilités d'émission de l'ordinateur vers la boîte, du fait de la capacité limitée de la boite Danube pour stocker les messages en transit.

Dans tous les cas, la porte :DANACK est plus prioritaire que la porte DANUBE.

La priorité actuelle des portes est : :DANACK = 1

:DANMES = 1

:DANUBE = 3

:DANOUV = 2

La porte DANOUV est plus prioritaire que DANUBE pour la seule raison qu'il est plus urgent de déclarer un nouveau nom afin de recevoir d'éventuels messages de Danube qui lui sont destinés.

2.1.3/ Remarques sur l'implantation du noyau et de L'A.C.D.

Nous voyons qu'un service réseau en datagramme est parfaitement réalisable avec un développement relativement simple. L'acteur arrive parfaitement à concilier les contraintes du dialogue machine/réseau avec les containtes propres à Chorus concernant l'enchaînement local des éxécutions d'acteur.

Nous rappelons les aspects système qui ont été abordés de façon originale dans la description de l'acteur de commutation Chorus.

Outre le mécanisme de communication acteur/noyau nous utilisons:

- les traitements des interruptions par message,
- l'utilisation des priorités, pour l'enchaînement des activations d'acteur,
- l'utilisation des SELECTIONs qui évitent de déclencher le traitement d'un message Chorus qui ne pourrait être envoyer "dans la foulée" sur le réseau, (on évite ainsi le stockage par l'acteur de commutation distante des messages Chorus non expédiés),

Nous pensons que les détails donnés ici suffisent à éclairé le lecteur sur la façon dont on peut réaliser le raccordement d'une machine à un réseau avec un acteur Chorus.

3/ L'acteur de transport

L'acteur de transport est chargé de la gestion du transport tel qu'il a été défini dans le modèle à savoir:

- offrir un mode non connecté en datagramme sans diagnostic pour l'utilisateur, mais avec diagnostic positif (ACK pour un transport "fiabilisé") entre acteurs de transport des différents sites,
- offrir un mode connecté pour les acteurs utilisateurs qui en font la demande explicite à l'acteur de transport.

Nous décrirons donc successivement le fonctionnement de l'acteur de transport pour le mode non connecté et pour le mode connecté.

Avertissement:
Notre propos n'est pas de décrire une "n + unième" station de transport car nous en supposons le principe bien connu aujourd'hui, mais de montrer l'influence de Chorus sur la conception d'une telle machine. Nous nous attachons donc principalement à la définition et à l'enchaînement des étapes de traitements nécessaires à la gestion du service de transport dans l'architecture Chorus.

3.1/ Le mode non connecté

3.1.1/ Les interfaces

Les_portes

L'acteur de transport recoit les messages à transporter à distance sur la porte par défaut (:PC_DEFAUT = :IRADISI). Il détermine leur routage et les remet à l'acteur de commutation sur la porte :DANUBE.

Il reçoit également les messages venant des autres sites et destinés à des portes locales à son site sur la porte de [X]ACTRANS qui permet au(x) acteur(s) de commutation de son site et aux acteurs de transport des autres sites de l'identifier.

Les_messages

Sur la porte :TRADIST le noyau dépose les messages utilisateur sans aucune modification.

Sur la porte [X]ACTRANS, les messages reçus des acteurs de commutation sont suivant le cas:

- un message de service, localisation de porte (interrogation ou réponse) ou acquittement de message,
- un message de données de transport (les données de transport sont le message utilisateur).

Les messages de transport distant ont la structure suivante:

[SiteR]ACTRANS,[SiteE]ACTRANS,Code_serv,Code_user,Données.

SiteE, SiteR = site d'émission et de réception du message,

Code_serv = code du type de message de service,

"DTGR_LOC" : où est la porte (donnée);

"DTGR_REP" : La porte (donnée reçue) est ici (donnée);

"DTGR_ACQ" : j'acquitte le code_user (donnée);

"DTGR_INC" : la porte (donnée) n'est pas sur mon site;

"DTGR_MSG" : je vous transmet un message utilisateur,

Code_user = code de référence des données qui suivent,

Données = données du message,

message utilisateur si code_serv = DTGR_MSG;

données de service si code_serv = ≠ DTGR_MSG;

3.1.2/ Structure de l'acteur de transport

3.1.2.1/ Les étapes de traitement

On distingue trois étapes de traitement:

- une étape d'initialisation de l'acteur de transport,
- une étape de traitement (No 2) des messages de la porte :TRADIST,
- une étape de traitement (No 3) des messages de la porte [X]ACTRANS.

Traitement_No_1

Ce traitement initialise les portes de l'acteur de transport, ainsi que les tables qui lui servent à mémoriser les messages en cours d'acquittement. Pour un message donné l'acteur de transport conserve:

- la référence du message (générée par lui),
- le nombre de tentatives pour le transmettre (on fixe un MAX).

L'acteur de transport dispose d'une table de routage qu'il met à jour régulièrement au cours de ses recherches. Cette table contient la localisation (nom du site) des portes nomades dans le système réparti. Les portes sédentaires sont localisées par interprétation du nom. Cette table est limitée en taille, toute nouvelle localisation efface la plus ancienne lorsque la taille maximale est atteinte.

L'acteur de transport génère une numérotation qui lui est propre pour référencer les messages transmis aux autres acteurs de transport.

Lorsque les initialisations sont terminées, les aiguillages sont définis ainsi:

:TRADIST --> traitement 2,

[X]ACTRANS --> traitement 3.

Traitement_No_2

L'étape de traitement No 2 est chargée de la prise en charge des messages destinés à des portes non locales. L'algorithme est le suivant:

```
DEBUT

   le nom de la porte est-il dans ma table de routage?
       SI OUI
          DEBUT
           numéroter le message;
           mémoriser le message;
           préparer le message de transport distant pour
                la porte [X]ACTRANS du site de localisation;
           envoyer le message à l'acteur de commutation distante;
          FIN
       SI NON
          DEBUT
           préparer un message de demande de localisation
                     pour tous les acteurs de transport connus;
           envoyer le message à l'acteur de commutation distante;
           (* On dispose de la diffusion au     *)
           (* niveau réseau en un seul message *)
           mémoriser le message utilisateur;
           compter le nombre de tentatives;
          FIN
    armer une temporisation sur la porte [X]ACTRANS;

FIN
```

Si l'acteur de transport ne peut plus mémoriser de message en attente d'acquittement, il ne sélectionne plus la porte :TRADIST.

Les Sélections sont donc les suivantes:

```
DEBUT
    SELECT ([X]ACTRANS,pnil);
    SI (place pour mémoriser message) ALORS
     SELECT (:TRADIST,pnil); (* selection quelquesoit *)
                             (* la porte émettrice   *)
FIN
```

Traitement_No_3

Cette étape de traitement concerne les messages arrivant des acteurs de transport distant des autres sites. Dans la description qui suit le message reçu est désigné par MsgR L'algorithme est le suivant;

```
DEBUT
  SI MsgR = ACK (ref) ALORS
      le message de référence ref est enlevé de la liste des
      messages en attente d'acquittement;
  SI MsgR = Ou Porte x ? ALORS
      SI la porte s est locale ALORS
         envoyer message "porte sur mon site N";
  SI MsgR = Msg (ref) ALORS
      DEBUT
       SI porte réceptrice locale ALORS
          DEBUT
           acquitter ref;
           mémoriser dernière ref acquittée pour site J;
           décoder message de transport;
           envoyer message utilisateur vers porte réceptrice locale;
          FIN
       SI NON envoyer Nack à acteur de transport émetteur;
```

```
SI MsgR = Nack (ref) ALORS rechercher site de résidence ;
SI MsgR = Porte sur site N ALORS
   DEBUT
    envoyer message en attente de localisation;
    mémoriser message sur attente de l'acquittement;
   FIN
FIN
```

A la fin de cette étape de traitement les règles de sélection sont identiques à celle de l'étape de traitement No 2).

3.1.2.2/ Traitement des exceptions

Les exceptions en cours de transport sont perçues au moyen des temporisations. Dans le cas de l'acteur de transport elles sont demandées après l'envoi d'un message qui doit être acquitté ou après l'envoi d'une demande de localisation. C'est deux exceptions sont traitées sur la porte [X]ACTRANS.

Dans le cas de l'attente de l'ACK, il y a N reémissions du message de données.
Dans le cas de la localisation, il peut y avoir N reémissions de la demande de localisation.

3.1.2.3/ Les priorités des portes

On n'observe aucune contrainte particulière. Il est souhaitable de traiter les messages de la porte [X]ACTRANS en priorité car un acquittement libère une ressource de mémorisation et permet donc de sélectionner éventuellement la porte :TRADIST.

3.2/ Conclusions sur le mode non connecté

Le mode non connecté ne demande que peu de travail supplémentaire par rapport au service de commutation. C'est pourquoi il est possible, dans le cas où il est le seul mode de transport implanté, de réunir dans le même acteur transport et commutation. La charge du traitement due au transport n'est pas pénalisante pour les performance des communications avec l'interface réseau.

3.3/ Le mode connecté

La réalisation du mode connecté met l'accent sur deux difficultés supplémentaires dans l'organisation de l'acteur:
- d'une part la gestion des ressources nécessaires à la réalisation du service (ensemble des contextes de connexions, des portes privées, offrir l'égalité entre tous les utilisateurs du service),
- d'autre part la gestion d'une connexion U, c'est-à-dire le service rendu aux utilisateurs.

Nous présentons d'abord, l'aspect gestion des connexions U dans leur ensemble puis l'aspect gestion de la connexion U.

3.3.1/ Gestion des connexions

L'acteur de transport communique localement avec l'acteur utilisateur en connexion U et à distance avec l'acteur de transport du site de l'utilisateur distant de la même connexion U.

Les communications entre acteurs de transport utilisent:
- soit le transport en datagramme fiabilisé (service de transport distant de base),
- soit des connexions de transport distant (connexions T) qui peuvent être de type circuit ou liaisons datagrammes ST2 Cyclades.

Dans le cas où le transport distant utilise des connexions T, les connexions U sont multiplexées sur les connexions T établies entre les acteurs de transport des sites concernés. Les connexions T offrent un avantage en ce sens qu'elles permettent le contrôle de flux, d'erreur et l'anticipation d'un nombre n de messages et améliorent le débit du transport distant pour le rapprocher de celui du transport local.

Nous décrivons ici l'implantation qui utilise les connexions T.

3.3.1.1/_Gestion_des_connexions_I

Les acteurs de transport distant établissent les connexions T en suivant le protocole adéquat (ouverture, transfert).

Nous n'avons pas de protocole particulier à mettre en exergue. Celui que nous avons utilisé, pour cette expérience, est inspiré du protocole X25 (numérotation cyclique, fenêtre de crédit, messages referencés et acquittement par "prêt à recevoir").

Nous ne retenons que le fait suivant:

- lorsque la connexion T est établie, l'acteur de transport la considère, en émission, comme une simple file d'attente de longueur finie (paramètre variable: par exemple, crédit sur la connexion) dont il connaît le taux de disponibilité en fonction des crédits qu'il reçoit dans le sens réception. Cette remarque peut paraître simpliste, mais elle permet de justifier la façon dont les connexions U sont multiplexées sur les connexions T (cf. plus loin).

Les connexions T sont gérées à travers la porte [X]CONDIST qui identifie la porte de connexion distante entre les acteurs de transport. Le problème de gestion des ressources, au niveau des connexions T, réside dans la possibilité d'établir des connexions T en fonction des demandes de connexions U émanant des utilisateurs.

Les ressources nécessaires pour une connexion T sont les suivantes:

Une table de contexte décrite en Pascal qui contient, outre les informations d'état de la connexion T, les zones de stockage des messages en transit, les informations de contrôle de la connexion T.

```
CONT_CON_T =  RECORD
```

```
libre      : BOOLEAN; (* indicateur d'occupation *)
ouvert     : BOOLEAN; (* indicateur d'ouverture  *)
port_dist  : TC_NOM;  (* Nom de la porte         *)
                      (* transport distante      *)
time_out   : INTEGER; (* valeur du délai de      *)
                      (* temporisation           *)
outqueue   : RECORD   (* queue d'accs à la connexion T     *)
               tete     : INTEGER; (* pointeur tete de file  *)
               queue    : INTEGER; (* pointeur queue de file *)
               nbmess   : INTEGER; (* nombre de mes. en file *)
               long     : INTEGER; (* longueur max. de file  *)
               tab_mes  : TAB_EV_CV (* tableau des messages  *)
                                    (* memorises             *)
             END;
versite    : DEPARLI; (* contexte d'emission sur  *)
                      (* connexion T              *)
desite     : ARRIV_LI;(* contexte de recption sur *)
                      (* connexion T              *)

             END;
```

Avec :

```
ARRIV_LI = RECORD

ouvert     : BOOLEAN; (* indicateur d'ouverture sens emission *)
num_cv     : INTEGER; (* numero du circuit ouvert             *)
debfene    : INTEGER; (* debut fenetre emission               *)
finfene    : INTEGER; (* fin fenetre emission                 *)
tab_em_cv  : TC_TAB_CV; (* tableau des mess emis sur conn. T  *)
lastsend   : INTEGER; (* ref. du dernier message envoye       *)
evdanfen   : INTEGER; (* nombre de messages deja envoyes dans *)
                      (* la fenetre = credit restant          *)
```

```
            END;

DEPAR_LI = RECORD

ouvert     : BOOLEAN; (* indicateur d'ouverture sens reception *)
num_cv     : BOOLEAN; (* numero du circuit ouvert              *)
debfenr    : INTEGER; (* debut fenetre reception               *)
finfene    : INTEGER; (* fin fenetre reception                 *)
tab_rec_cv: TC_TAB_CV; (* tableau des mes. recus sur conn. T   *)
marq_cv    : TC_MARQ_CV; (* marquage des messages recu dans la *)
                         (* fenetre de reception               *)
                         (* gestion des ACK (paquet de BOOLEAN)*)

            END;
```

Notre implantation ne comportant que peu de sites, nous n'avons pas rencontré de problèmes de réservation de ressources pour réaliser l'acteur de transport en mode connecté utilisant des connexions T.

3.3.1.2/ Gestion des connexions U

Pour chaque connexion U, l'acteur de transport gère un contexte qui permet d'identifier les partenaires, le chemin emprunté par la connexion U (connexion D), et les informations en transit.

A un moment donné, l'acteur de transport est donc limité dans le nombre de connexions U qu'il peut traiter. Le mécanisme de sélection va être utilisé pour le contrôle de flux dans les deux cas suivants:
1) si l'on convient de dédicacer une porte privée à chacunes des connexions U, on peut facilement limiter le nombre de connexions U ouvertes (on ne selectionne plus la porte "publique" qui traite les demandes d'ouvertures),
2) s'il n'est pas possible, pour une raison quelconque, de traiter un nouveau message sur d'une connexion U particulière, alors on ne

sélectionnne pas la porte privée qui lui est allouée. (Les raisons invoquées peuvent être les suivantes; plus de crédit sur la connexion T utilisée pour transporter les messages de la connexion U concernée, pas de crédit sur la connexion U, etc.).

Pour que tous les utilisateurs soient à égalité, il faut aussi pouvoir éviter les interactions (un acteur utilisateur qui ne respecte pas le protocole peut saturer les portes de l'acteur de transport). Les portes privées dont le nom est généré (et fourni à l'utilisateur dans le message d'ouverture) par le système permettent de résoudre une partie du problème en éliminant les interactions entre utilisateurs. Le mécanisme de sélection en fonction des droits d'émettre ou non des données sur la connexion U réalise un mécanisme de contrôle de flux simple et absolu (du point de vue de l'acteur de transport). Si l'utilisateur ne respecte pas le protocole, il ne pénalise que sa propre porte privée (et à plus long terme le système).

3.3.2/ Gestion de la connexion U

Lorsque l'acteur de transport recoit une demande d'ouverture de connexion U (locale ou distante), il doit affecter un contexte à la connexion et lui ouvrir une porte privée. Tous les échanges locaux se font avec cette porte privée. Ils ne sont pris en compte que si les deux conditions suivantes sont réunies:

- le crédit est positif sur la connexion U,
- il y a de la place dans la file d'attente de la connexion T utilisée.

Cela signifie que l'acteur de transport ne traite les messages de connexion U que s'il a la possibilité de les faire progresser sur la connexions U.

Remarque:

L'acteur de transport d'un site anticipe sur le contrôle de flux distant en envoyant OK à chaque message qu'il consomme sur la porte privée, quitte à rester aveugle (SELECTION) si le crédit ne change pas par la suite.

Contexte_de_la_connexion_U

L'acteur de transport dispose d'un ensemble de contextes qu'il affecte aux connections U en cours.

```
CAR_PORT = RECORD
             libre      : BOOLEAN ; (* indicateur d'occupation  *)
             ouvert     : BOOLEAN ; (* indicateur d'ouverture   *)
             port_serv  : TC_NOM  ; (* nom porte privée         *)
             port_abo   : TC_NOM  ; (* porte locale en Con. U   *)
             port_dist  : TC_NOM  ; (* porte distante en Con. U *)
             message    : TC_MESSAGE ; (* message en attente    *)
                                       (* d'être délivré        *)
           END;
```

Dans cet implantation l'acteur de transport ne mémorise que le dernier les message distant reçu, jusqu'à ce qu'il puisse effectivement le remettre à l'abonné.

D'autre part, un message accepté par l'acteur de transport sur une porte privée est mémorisé par le transport distant. On considère qu'il est donc inutile de le conserver au niveau du contexte de la connexion U.

a) Lorsque l'acteur de transport traite un message sur la porte privée, il effectue le contrôle propre à la connection U (connexion ouverte, droit d'émettre sur la porte privée, vérification du destinataire demandé). Si les contrôles permettent l'émission, il

dépose le message dans la file d'attente de la connexion T utilisée par la connexion U.

b) Lorsque l'acteur de transport traite un message distant sur la connexion U, il effectue les contrôles associés puis émet le message vers l'abonné local. Il autorise alors l'acteur de transport distant à émettre un autre message.

3.3.3/ Etapes de traitement

Le mode connecté introduit trois nouvelles étapes de traitement. La première pour le traitement des demandes d'ouverture de connexions U, la seconde pour le traitement du protocole de transport distant (dans le cas d'utilisation des connexions T), la troisième pour le traitement en cours de connexion U (associé à la porte privée).

Initialisation

Pour initialiser le mode connecté utilisant des connexions T, l'acteur de transport doit déclarer deux portes supplémentaires:
la porte des demandes d'ouvertures de connexions U (:OUV_CU),
la porte de connexion T ([X]CONDIST).

Ces portes sont sélectionnées puisque les ouvertures locales et distantes sont possibles à l'initialisation de l'acteur (assez de ressources disponibles).

Traitement_U1

Ce traitement permet l'initialisation des connexions U après demande locale sur la porte "demande d'ouverture" (:OUV_CU). Le déroulement est le suivant:

```
DEBUT
 identification du demandeur;
 localisation de l'abonné distant;
 identification de la connexion T à utiliser;
 SI connexion non existe ALORS Ouverture de la connexion T;
  SI impossible ALORS refuser Connexion U;
  SI NON
  DEBUT
    envoyer demande d'ouverture de connexion U;
    reservation d'un contexte de connexion U;
  FIN;
FIN;
```

A la fin de cette étape la porte :OUV_CU est sélectionnée s'il reste des contextes disponibles.

Traitement_U2

Ce traitement permet de gérer la connexion T et les connexions U multiplexées dessus entre deux acteurs de transport. Le déroulement est le suivant:

```
DEBUT
 SUIVANT QUE message recu est :
 'ouverture de connexion T' : ALORS accepter ou non la connexion;
 'fermeture de connexion T' : ALORS
                              accepter la fermeture de connexion T;
 'donnees sur connexion T ' : ALORS
```

```
DEBUT
  acquitter message de connexion T recu;
  SI nouvelle connexion U demandees ALORS
    DEBUT
      initialiser contexte;
      ouvrir porte privée;
      envoyer demande a abonne local;
      selecter porte privee associee;
    FIN
  SINON
  DEBUT
    vérifier provenance et destination des
        donnees de la connexion U;
    SI message de crédit ALORS SELECT(port_priv de connexion U);
    SI message de donnees ALORS SEND vers port_abo local;
  FIN;
FIN;
```

Traitement_U3

Ce traitement permet de gérer la connexion U ouverte. Le déroulement est le suivant:

```
DEBUT
  SUIVANT QUE message recu est:
  'confirmation ouverture' : ALORS
                             DEBUT
                               confirmer l'ouverture à
                               l'acteur distant;
                               declarer connexion U ouverte;
                               selectionner la porte privee;
                             END;
```

```
'donnees sur connexion U': ALORS
                           DEBUT
                             mettre le message en file sur
                                                connexion T ;
                             repondre OK a l'abonne local;
                             deselectionner la porte privee;
                           FIN;
'fermeture connexion U  ': ALORS
                           DEBUT
                             informer l'acteur distant;
                             fermer le contexte local;
                             detruire la porte privee;
                           FIN;
FIN;
```

Autres_traitements

Ces traitements ne sont pas identifiés en tant qu'etape de traietment. Il se rapportent aux exceptions et aux sélections.

Les exceptions identifiées et traitées par le transport en mode connecté sont l'absence de message; sur les portes [X]CONDIST et sur la porte privée.

Une temporisation dépassée sur la porte CONDIST provoque la reémssion du dernier message envoyé sur la connexion T.
Une temporisation dépassée sur la porte privée provoque la reémission du dernier message envoyé qui est; soit le OK, soit le dernier message de données rec,u.

Les Sélections sont effectuées à la fin de chaque étape de traitement en fonction d'un diagramme d'état (sous forme de tableau) qui représente en fait le_masque_d'activation des portes de l'acteur.

Le tableau de masque a la structure suivante:

entrée n : nom de porte,

condition; 0 selectable, 1 non selectable.

La condition est déterminée comme il est indiqué plus haut (gestion de la connexion U).

La procedure SELECTION (qui pose les selections en fonction de la table) est appeleé à la fin de chaque étape de traitement.

3.3.4/ Format des messages

Nous donnons ici le contenu du texte des messages utilisateur pour le service de transport en mode connecté.

Tout les messages de connexion U rentre dans un modèle commun qui contient une en-tête de paramètres de connexion U:

```
mess_conU  = RECORD

code_serv : TC_SERV; (* code de service connexion U       *)
port_abo  : TC_NOM ; (* porte de l'utilisateur demandeur *)
port_dist : TC_NOM ; (* porte de l'utilisateur distant   *)
port_priv : TC_NOM ; (* porte privee de connexion U      *)
text      : PACKED ARRAY [1..MAX_DATAU] OF CHAR ;

            END;
```

MAX_DATAU = taille maximale des données sur connexion U.

Le tableau text recoit les données pour chacunes des services demandés sur la connexion U.

Dans cet implantation, toutes les données (crédit, taille de message, etc.) sont implicites à l'ouverture. Il n'y a donc pas de paramètres supplémentaires à fournir dans le message et le texte est vide pour l'ouverture, la confirmation d'ouverture et de fermeture.

La fermeture peut comporter un diagnostic qui est alors inclu dans le texte qui se décrit comme suit:

```
textferm = RECORD
             diag_ferm : TC_DIAG; (* diagnostic de fermeture *)
           END;
```

Dans les messages de données utilisateurs, le texte est rempli par les données utilisateur.

3.4/ Conclusions sur l'acteur de transport

L'acteur de transport que nous venons de décrire ne présente aucune difficulté de réalisation. Le déroulement du mode connecté semble affecter raisonnablement le délai de transfert entre deux machines. La modularité de Chorus permet une description et une implantation aisée des fonctions à remplir. Nous pensons que la description faite ici aura convaincu le lecteur.

4/ Les acteurs utilisateurs

Nous donnons dans ce paragraphe un exemple d'organisation et de programmation des acteur utilisateurs. Afin de faciliter cette construction, un certains nombres de procédures d'interface pour le mode connecté sont mises à disposition du programmeur.

4.1/ Les procédures d'interface

Ouverture de connexion U

```
PROCEDURE PA_OUV_CU( port_abo  : TC_NOM;
                     port_dist : TC_NOM;
                 VAR port_priv : TC_NOM)
```

Procédure asynchrone d'ouverture de connexion U.

```
PROCEDURE PS_OUV_CU( port_abo  : TC_NOM;
                     port_dist : TC_NOM;
                 VAR port_priv : TC_NOM;
                 VAR diag      : TC_DIAG)
```

Procédure synchrone d'ouverture de connexion U. En retour le diagnostic est DC_OK si la connexion est ouverte.

Confirmation d'ouverture

```
PROCEDURE PA_CONF_OUV_CU( port_abo  : TC_NOM;
                          port_dist : TC_NOM;
                          port_priv : TC_NOM)
```

Procédure asynchrone de confirmation d'ouverture.

Envoi_de_données

```
PROCEDURE PA_ENV_CU( port_abo  : TC_NOM;
                     port_dist : TC_NOM;
                     port_priv : TC_NOM;
                     mess      : TC_TEXT_CONU)
```

Procédure asynchrone d'envoi de données sur connexion U.

Fermeture_de_connexion_U

```
PROCEDURE PA_FERM_CU( port_abo  : TC_NOM;
                      port_dist : TC_NOM;
                      portpriv  : TC_NOM)
```

Procédure asynchrone de fermeture de connexion U.

```
PROCEDURE PS_FERM_CU( port_abo  : TC_NOM;
                      port_dist : TC_NOM;
                      portpriv  : TC_NOM;
                      diag      : TC_DIAG)
```

Procédure synchrone de fermeture de connexion U. L'utilisateur ou le service de transport peuvent donner un code diagnostic à la fermeture.

Confirmation_de_fermeture

```
PROCEDURE PA_CONF_FERM_CU( port_abo  : TC_NOM;
                           port_dist : TC_NOM;
                           port_priv : TC_NOM)
```

Procédure asynchrone de confirmation de fermeture.

Acceptation_nouveau_message

```
PROCEDURE PA_OK_CU( port_abo  : TC_NOM;
                    port_dist : TC_NOM;
                    port_priv : TC_NOM)
```

Procédure asynchrone d'acceptation de message en réception sur la connexion U.

4.2/ Organisation de l'acteur utilisateur

Dans sa structure un acteur utilisateur est identique aux acteurs que nous venons de décrire. Le modèle de programmation est fixé par Chorus pour une implantation donnée L'organisation des étapes de traitement, les portes choisies et le format des messages sont définis par le concepteur de l'acteur. Nous pensons que l'organisation de l'acteur en mode non connecté n'apporte pas de précision supplémentaire et nous nous attachons donc à décrire l'utilisation du mode connecté. Nous prenons l'exemple de l'impression d'un fichier avec connexion utilisateur entre le serveur d'imprimante et un client.

Le client demande l'ouverture et envoi le fichier par blocs. A la fin de la lecture il ferme la connexion U. On suppose l'ouverture et la fermeture asynchrones.

4.2.1/ Programme client d'imprimante

```
Program Client imprimante;
DEBUT
...
...
 etape 1;
```

```
        DEBUT
          (* initialisations *)
          OUVRIR(PCONIMP,...) (* porte connexion vers imprimante *);
          SWITCH (PCONIMP,2);
          PA_OUV_CU(PCONIMP, IMPRIME, PORTCON1,diag);
          (* PORTCON1 = porte privee de connexion U *)
        END;

etape 2;
        DEBUT
          (* attente de confirmation *)
          SI confirmation ouverture alors
          DEBUT
            enregistrer nom de porte privee allouee (PORT_PRIV);
            n := 1;
            lire blocn;
            PA_ENV_CU(PORT_PRIV,PCONIMP,PORTCON1,bloc1);
            SWITCH(PCONIMP,3);
          FIN;
          SINON AUTODESTRUCTION;
        FIN;
etape 3;
        DEBUT
          (* attente sur connexion U *)
          SI message = OK_CU ALORS
            DEBUT
              n := n+ 1;
              lire blocn;
              SI fin fichier alors PA_FERM_CU(PORT_PRIV,PCONIMP,...);
              SI NON
                 DEBUT
```

```
                    lire blocn;
                    PA_ENV_CU(PORT_PRIV,PCONIMP,PORTCON1,blocn);
                  FIN;
            FIN;
          SI message = CONF_FERM ALORS AUTODESTRUCTION;
        SI message = TEMPOrisation ALORS
                    reemettre dernier message sur connexion;
        FIN;

FIN. (* program client imprimante *)
```

Nous donnons maintenant un exemple du même acteur avec ouverture et fermeture synchrones.

```
Program Client imprimante;
DEBUT
...
...
 etape 1;
        DEBUT
          (* initialisations *)
          OUVRIR(PCONIMP,...) (* porte connexion vers imprimante *);
          SWITCH (PCONIMP,2);
          PA_OUV_CU(PCONIMP, IMPRIME, PORTCON1,diag);
          (* PORTCON1 = porte privee de connexion U *)
          SI diag <> DC_OK alors AUTODESTRUCTION;
          (* On commence la transmission du fichier *)
            n := 1;
            lire blocn;
            PA_ENV_CU(PORT_PRIV,PCONIMP,PORTCON1,bloc1);
            SWITCH(PCONIMP,2);
        FIN;
```

```
etape 2;
        DEBUT
          (* attente sur connexion U *)
          SI message = OK ALORS
            DEBUT
              n := n+ 1;
              lire blocn;
              SI fin fichier alors
                DEBUT
                  TANT QUE diag <> DC_OK FAIRE
                  DEBUT
                     PS_FERM_CU(PORT_PRIV,PCONIMP,PORTCON1,diag);
                     si diag = DC_OK ALORS AUTODESTRUCTION;
                  FIN;
                FIN;
              SI NON
                 DEBUT
                   lire blocn;
                   PA_ENV_CU(PORT_PRIV,PCONIMP,PORTCON1,blocn);
                 FIN;
            FIN;
          SI message = TEMPOrisation ALORS
            reemettre dernier message sur connexion;
        FIN;

FIN. (* program client imprimante *)
```

4.2.2/ Programme de serveur d'imprimante

Le programme du serveur d'imprimante se présente ainsi:

```
Program serveur imprimante;
DEBUT
  etape 1:
          DEBUT
            (* initialisation pour attente de demandes *)
            OUVRIR(IMPRIME,....);
            SWITCH(IMPRIME,2);
          FIN;

  etape 2:
          DEBUT
            (* traitement d'une demande d'ouverture *)
            SI message = demande ouverture ALORS
            DEBUT
              PA_CONF_OUV_CU(IMPRIME,PCONIMP,PORT_PRIV1);
              SELECT(PORT_PRIV1,IMPRIME);
              SWITCH(IMPRIME,3);
            FIN;
          FIN;

  etape 3:
          DEBUT
            (* impression du fichier *)
            SI message = donnees connexion ALORS
              DEBUT
                imprimer bloc recu;
                PA_OK_CU(IMPRIME,PCONIMP,PORT_PRIV1);
                SELECT(PORT_PRIV,IMPRIME);
              FIN;
```

```
            SI message = fermeture ALORS
              DEBUT
                PA_CONF_FERM_CU(IMPRIME,PCONIMP,PORT_PRIV);
                SWITCH(IMPRIME,2);
              FIN;
            FIN;
FIN; (* Program serveur imprimante *)
```

5/ Conclusions sur l'implantation

Les informations fournies ici reflètent l'approche qui a prévalu pour la première implantation de Chorus. Le lecteur aura compris qu'étant donnée la grande liberté de manoeuvre qu'offre le système Chorus, nous nous sommes surtout attaché à expliquer la démarche et les possibilités offertes lors de la conception d'une application répartie.

RÉSUMÉ:

Cette étude s'inscrit dans le cadre des recherches sur les systèmes répartis et porte sur l'analyse des principes et mécanismes de la communication dans l'univers réparti.
Les problèmes de performances des communications (fiabilité et rapidité) n'y sont pas négligeables et nous proposons ici un modèle de communication qui affranchit le concepteur d'une application des difficultés liées au transport de l'information sans lui en masquer les conséquences.

Le service de communication décrit est construit dans et avec l'architecture du système d'exploitation CHORUS, fondé sur la communication par message entre des "ACTEURS". Le modèle d'I(nterconnexion) de S(ystèmes) O(uverts) sert de base à notre modèle, dans la mesure où il nous permet de construire des couches distinctes pour organiser la communication dans Chorus.

La première partie de ce mémoire situe le service de communication dans un système réparti et énumère les premières hypothèses sur le service de communication. La deuxième partie présente les éléments essentiels de l'architecture CHORUS, puis décrit les caractéristiques du modèle de communication choisi pour CHORUS.
La troisième partie fait l'analyse des besoins minimaux, du point de vue du transport, d'une application répartie s'exécutant sur CHORUS.
La quatrième partie décrit les mécanismes de base nécessaires à la construction du service de communication (aspect machine dépendant et réeau dépendant). La cinquième partie décrit le service de transport de base offert par CHORUS, puis la façon dont il est possible de construire un service de transport plus élaboré qui offre, par exemple, les connexions entre les portes des acteurs.

Le lecteur trouvera dans la sixième partie une évaluation des problèmes et solutions rencontrés lors de l'étude du service de communication de l'architecture CHORUS.

MOTS CLÉS: **CHORUS – Système Réparti – Acteur – Transport – Réseau – Protocole Client-Serveur – Protocole Réseau – Protocole Transport – Connexion**

ISBN 2 - 7261 - 0384 - 7