N° d'ordre: 76

THESE

présentée devant

L'UNIVERSITE DE RENNES I
U. E. R. MATHEMATIQUES ET INFORMATIQUE

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE RENNES I
(ARRETE MINISTERIEL DU 5 JUILLET 1984)

MENTION INFORMATIQUE

par

José LEGATHEAUX MARTINS

Sujet de la Thèse :

LA DESIGNATION ET L'EDITION DE LIENS DANS LES SYSTEMES D'EXPLOITATION REPARTIS

Soutenue le 26 Novembre 1986 devant la Commission d'Examen composée de :

MM.	J. P.	VERJUS	Président
	F.	ANDRE	Rapporteurs
	C.	KAISER	
	J. P.	BANATRE	Examinateurs
	J. S.	BANINO	
	M.	GUILLEMONT	

Thèse préparée au sein du projet CHORUS - SYSTEMES INFORMATIQUES REPARTIS
de l'INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

N° d'ordre: 76

THESE

présentée devant

L'UNIVERSITE DE RENNES I
U. E. R. MATHEMATIQUES ET INFORMATIQUE

pour obtenir le titre de

DOCTEUR DE L'UNIVERSITE DE RENNES I
(ARRETE MINISTERIEL DU 5 JUILLET 1984)

MENTION INFORMATIQUE

par

José LEGATHEAUX MARTINS

Sujet de la Thèse :

LA DESIGNATION ET L'EDITION DE LIENS DANS LES SYSTEMES D'EXPLOITATION REPARTIS

Soutenue le 26 Novembre 1986 devant la Commission d'Examen composée de :

MM.	J. P.	VERJUS	Président
	F.	ANDRE	Rapporteurs
	C.	KAISER	
	J. P.	BANATRE	Examinateurs
	J. S.	BANINO	
	M.	GUILLEMONT	

Thèse préparée au sein du projet CHORUS - SYSTEMES INFORMATIQUES REPARTIS
de l'INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET AUTOMATIQUE

Je remercie très sincèrement:

M. Jean-Pierre VERJUS, Professeur de l'Université de Rennes et Directeur du CNRS, qui me fait l'honneur de présider le jury de cette thèse.

M. Jean-Pierre BANATRE, Professeur à l'INSA-Rennes, de la confiance qu'il m'a témoignée en dirigeant mon travail et en participant au jury de cette thèse.

Madame F. ANDRE, Professeur à l'Université de Rennes, M. J. S. BANINO, Directeur du Departement de Recherche et Développement de MATRA Datasystèmes et M. C. KAISER, Professeur au CNAM-Paris, qui ont accepté de participer au jury de cette thèse.

M. Marc GUILLEMONT, pour sa participation au jury. Je tiens à lui témoigner ici ma reconnaissance pour son intérêt à ce travail ainsi que pour l'aide constante et l'amitié qu'il m'a apportées tout au long de mes recherches.

Tous mes collègues du projet CHORUS qui ont participé à l'expérience, dont une partie est décrite dans cette thèse, en particulier MM M. ROZIER, F. ARMAND et F. HERRMANN pour leur aide et les critiques constructives qu'ils m'ont apportées.

Madame Madalena QUIRINO et M. Luis MONTEIRO, Professeurs à l'Université Nouvelle de Lisbonne et M. José GRAÇA MARTINS, Professeur à l'Université Classique de Lisbonne, ont guidé mes débuts dans la recherche. Je les remercie très sincèrement pour leur soutien constant et amical.

Mes travaux ont été partiellement supportés par la FONDATION CALOUSTE GULBENKIAN de Lisbonne. Je tiens à exprimer ici ma reconnaissance à Madame M. R. ALMEIDA DIAS, Directrice du Service de Bourses d'Etudes, pour l'intérêt qu'elle m'a témoigné tout au long de mon séjour en France.

UNIVERSITE de RENNES I

U.E.R. Sciences et Philosophie

DOYENS HONORAIRES

M. MILON Y.
M. LE MOAL H.
M. MARTIN Y.
M. BOCLE J.

PROFESSEURS HONORAIRES

M. FREYMANN R.
M. ROHMER R.
M. MILON Y.
M. SALMON-LEGAGNEUR F.
M. VALLET P.
M. PHILIPPOT A.
M. VENE J.
Mlle CHARPENTIER M.
M. VACHER M.
M. VILLERET S.
M. VIGNERON L.
M. LE MOAL H.
M. PELTIER D.
Mlle DURAND S.
M. LE BOT J.
M. MARTIN Y.
M. RAZET P.
M. MAILLET P.
M. ALLEGRET P.
Mlle GOAS M.
Mlle GOAS G.

MAITRES de CONFERENCES HONORAIRES

Mlle HAMON M. R.

/...

MATHEMATIQUES et INFORMATIQUE

Professeurs

Mme ANDRE Fr.
M. BERTHELOT P.
M. CAMUS J.
M. CONZE J. P.
M. COSTE M.
M. CROUZEIX M.
M. FERRAND D.
M. GERVAIS M.
M. GIORGIUTTI I.
M. GUERINDON J.
M. GUIVARC'H Y.
M. HOUDEBINE J.
M. KOTT L.
M. LEGOUPIL J.
M. LENFANT J.
M. LERMAN I.
M. MARIE R.
M. METIVIER G.
M. MIGNOT A.
M. RAYNAL M.
M. SEGUIN J.(IUT L.)
M. TOUGERON J. C.
M. TOURNEMINE G.
M. TRILLING L.
M. VERJUS J. P.
M. WOLF J. (IUT L.)

Docteurs d'Etat

Mme ALLAIN M.F.
M. CIAVALDINI J. F.
Mme COSTE-ROY M. F.
M. DIVAY M. (IUT L.)
M. GRAS R.
M. HENNION H.
M. LE PAGE E.
M. MAHE L.
M. MEMIN J.
M. MERRIEN J.
M. NOURRIGAT J. F.
M. PERRIN Gérard
Mme TOUGERON M.

PHYSIQUE

Professeurs

M. ARQUES P.Y.
M. BARON A. (IUT R.)
M. BENIERE F.
M. BERTEL L.(IUT L.)
M. BOULET Ch.
M. BRUN P.
M. COLIN Y.(IUT R.)
M. CORAZZA M. (IUT L)
M. DANIEL J.P.
M. DECAMPS E.A.
M. DUBOST G.
M.FOUCHE F.(IUT R.)
M.FUCHS J.J.
M. GROSVALD (IUT R.)
M. GUIDINI J.
M. HAEUSLER Cl.
M. DURAND A.
M. LE FLOCH A.
M. LE MEN J.F.(IUT L.)
M. LEROUX E.
M. LE TRAON A.
M. LEVASSEUR M.(IUT R.)
M. MALHERBE J.C.(IUT L)
M. MEINNEL J.
M. MEVEL J.Y.
M. NUSIMOVICI M.
M. RIAUX E. (IUT R.)
M. ROBIN S.
Mme ROBIN S.
née SALOMOND
M. STEPHAN G.
M. TERRET Cl.
M. THOMAS G.
M. VEZZOSI G.

Docteurs d'Etat

M. ANDRIAMIRADO C.—
M. BALCOU Y.
M. BAUDOUR J.
M. BERNARD D.
M. BERTAULT M.
M.BESNIER G.
M. BIDEAU D.
M. BOULIOU A.
M. CAILLEAU H.
M. CHAGNEAU(IUT R)
M. CHARBONNEAU
M.CHASSAY G.
M. COATRIEUX J.L.
M. DAUDE A.
M. DEFRANCE A.
M. ECOLIVET Cl.
M. GIRARD A.
M. GOMET J.Cl.
M. GOULPEAU L.
M. HAGENE B.
Mlle HAGENE M.
M. HOUDEAU J.P.
M. JEZEQUEL G.
M. JOUBERT P(IUT L)
M. LAMBERT B.(IUT L.)
M. LANGOUET L.
M. LARVOR M.
M. LECLEAC'H (IUT L.)
M. LE COMTE A.
M. LE DOUCEN R.
M. LENORMAND J.M.
M. MESSAGER J.C.
M. PILET J.C.
M. POEY P.
M. PRIOL M.
M. QUEFFELEC J.L.
M. RABACHE P.(IUT L.)
M. REBOURS B.(IUT L.)
M. RILLET Y.(IUT L.)
M. SAILLARD J.
M. SEIGNAC A.
M. TANGUY P.
M. T'KINT de ROODENBEKE A. (IUT R.)
Mme T'KINT de ROODENBEKE M. (IUT R.)
M. TROADEC J.P.(IUT R.)
M. TONNARD F (IUT R)

/...

CHIMIE

Professeurs

M. BARIOU B (IUT R.)
M. BRAULT A. (IUT R.)
M. CARRIE R.
M. DABARD R.
M. DIXNEUF P.
M. FOUCAUD A.
M. GRANDJEAN D.
M. GUERILLOT Cl.
M. HAMELIN Jack
M. LANG J.
M. LAURENT Y.
M. LE CORRE M.
M. LE GUYADER M. (IUT R)
M. LEVAS E.
M. LISSILOUR R.
M. LUCAS J.
M. MARTIN G. (ENSCR)
M. MAUNAYE M. (ENSCR)
M. PATIN H. (ENSCR)
M. SOYER N. (IUT R.)
M. TALLEC A.

Docteurs d'Etat

M. AUFFREDIC J.P
Mme BARS O. née BEAULIEU
M. BOTREL A.
Mme DANION R. née BOUGOT
Mme LE ROUZIC A. née BELLEVRE (ENSCR)
Mme TEXIER Fr. née BOULLET
M. BROCHU R.
M. CAILLET P.
M. CAREL Cl.
M. CARO B. (IUT L.)
Mme POMMERET M.F. née CHASLE (IUT R)
M. CORRE F.
M. DANION D.
M. DARCHEN A.
M. DORANGE G (ENSCR)
M. FAYAT Ch.
M. GADREAU Cl.
M. GAUDE J.
Mme LOUER M. née GAUDIN
M. GUERIN R.
M. GUILLEVIC J.
M. HAZARD R.
Mme PAPILLON née JEGOU D (IUT R.)
M. JOUCLA M.
M. JUBAULT M.
M. LAPLANCHE A. (ENSCR)
M. LEBORGNE G.
M. LE COQ A.
M. LE FLOCH Y. (ENSCR)
Mme UTJES M. née LE GALL
Mme RIVET P. née LE GUELLEC
Mlle LEPLOUZENNEC M.
M. MARTELLI J.
M. MEYER A.
M. MOINET Cl.
M. MORVAN J. (ENSCR)
M. PERSON H.
Mme de COURVILLE née PICHEVIN A.
M. PICOUAYS B.
M. PLUSQUELLEC D. (ENSCR)
M. POCHAT F.
M. POULAIN M.
M. PRIGENT Y. (IUT R.)
M. RAOUL E.
M. RAPHALEN D. (ENSCR)
M. RAULET Cl.
M. ROBERT A.
Mme CARLIER J. née ROLLAND (IUT R.)
M. SAILLARD J.Y.
M. SARRAZIN J.
Mme TEXIER Fr.
M. VENIEN F. (ENSCR)
M. VERDIER P.

GEOLOGIE

Professeurs

M. BONHOMMET N.
M. CHAUVEL J.J.
M. CHOUKROUNE P.
M. COGNE J.
M. HAMEURT J.M.
M. JAHN B.M.
M. LARDEUX H.
M. WILLAIME Ch.

Docteurs d'Etat

M. AUVRAY B.
Mme ESTEOULE née CHOUX J.
M. HENRY J.L.
Mme MORZADEC née KERFOURN M.T.
M. LE CORRE Cl.
M. LEFORT J.P.
M. MORZADEC P.
Mme OLLIVIER M.F. née PIERRE
M. MARTIN H.

/...

Professeurs

Docteurs d'Etat

BIOLOGIE CELLULAIRE et GENETIQUE

M. FOLLIOT R.
M. GOURANTON J.
M. JEGO P.
M. JOLY J.M.
M. LE PENNEC J.P.
M. PHILIPPE M.
M. WROBLEWSKI H.

M. BERNARD J.
M. BOISSEAU Cl.
M. COILLOT J.P.
M. GOURRET J.P.
M. GUILLET J.Cl.
M. HAMON Cl.

BIOLOGIE des ORGANISMES

Professeurs

M. CITHAREL J.
M. CLAUSTRES G.
M. DAGUZAN J.
M. GAUTIER J.Y.
M. HUON A.
M. LARHER Fr.
Mme LEMOINE C.
M. LE RUDULIER D.
M. NENON J.P.
M. TOUFFET J.
M. TREHEN P.

Docteurs d'Etat

M. BARBIER R.
M. BERNARD Th.
M. BERTRU G.
M. BRIENS M.
M. CANARD A.
M. CHAUVIN G.
M. DENIS Ch.
M. GLOAGUEN J.Cl.
M. GUYOMAR'CH J.Ch.
Mme HUBERT M.
née GUERGADY
M. LE GARFF B.
M. MICHEL R.
M. SAVOURE B.

PHILOSOPHIE

Professeurs

M. JACQUES Fr.
M. ORTIGUES E.
M. VETO M.

Docteurs d'Etat

M. CLAIR A.

/...

PERSONNEL C.N.R.S.

Directeurs de Recherche	Maîtres de Recherche	Chargés de recherche	
MATHEMATIQUES			
			M. CREPEL P.
			M. HARDY J.
INFORMATIQUE			
			M. DARONDEAU
CHIMIE			
M. SERGENT M.	M. CHEVREL R.	M. BATAIL P.	M. MARTIGNY P.
M. SIMONET J.	M. COEURET (ENSCR)	M. DEMERSEMAN B.	M. MATECKI M.
	M. DENIS J.M.	Mme BAUDY M.	M. MOREL G.
	M. GREE R.	née FLOC'H	M. NOEL H.
		M. FONTENEAU G.	M. PADIOU J.
		M. GUYADER J.	M. PENA
		M.HAMON J.R.	M. PERRIN A.
		M. LAPINTE Cl.	M. POTEL M.
		M. LE BOZEC H.	M. SIMONNEAUX G.
		M. LOUER D.	M. VAULTIER M.
		M. MARCHAND R.	
PHYSIQUE			
		M. DANG TRAN Q.	
		M. SANQJER M.	
GEOLOGIE			
M. CAPDEVILLA R.		M. BERNARD-GRIFFS J.	
M. COBBOLD P.		M. PARIS Fl.	
		M. PEUCAT J.J.	
		M. ROBARDET M.	
BIOLOGIE CELLULAIRE et GENETIQUE			
	Mlle GARNIER D.		
BIOLOGIE des ORGANISMES			
Mme GAUTIER A.		Mme CLOAREC A.	
M. GAUTIER J.P.		M. DELEPORTE P.	
		Mme DELEPORTE S.	
		M. DELETTRE Y.	
		M. DEPUTTE B.	
		Mlle RIVAULT C.	
		Mlle EYBERT M. C.	
		M. VANCASSEL M.	
		M. VIDAL J.M.	
ANTHROPOLOGIE			
M. GIOT P.R.	M. BRIARD J.	M. MONNIER J. L.	

TABLE DES MATIERES

TABLE DES MATIERES

TABLE DES FIGURES

INTRODUCTION

INTRODUCTION

Une architecture répartie est une collection de processeurs munis de mémoire, reliés par un réseau de communication et logiquement complétés par un (plusieurs) système d'exploitation.

Potentiellement, ces systèmes permettent l'amélioration de certaines propriétés des systèmes informatiques, en particulier:

- l'efficacité - car ils offrent la possibilité d'utiliser simultanément plusieurs processeurs;
- la fiabilité - parce que l'arrêt d'une partie du système ne conduit pas nécessairement à l'arrêt de tout le système;
- le partage de ressources - parce qu'ils offrent la possibilité de partager des composants matériels ou logiciels;
- l'extensibilité, la flexibilité et la modularité - car ils permettent une expansion souple et progressive du matériel et du logiciel en fonction des besoins.

Quelques uns de ces avantages ne sont que potentiels. Des travaux de recherche se développent à l'heure actuelle afin de les rendre effectivement disponibles. Ces travaux portent sur plusieurs thèmes, parmi lesquels l'étude des systèmes d'exploitation répartis joue un rôle important.

1. Systèmes d'exploitation répartis

Pour libérer l'usager de la gestion des ressources, une architecture répartie doit être complétée par des couches de logiciel. Cette architecture doit aboutir à une machine virtuelle répartie qui masque la localisation des ressources et permette leur partage.

Les choix actuellement disponibles pour intégrer au niveau du système d'exploitation la répartition peuvent être ainsi classés:

1/ Réseaux de systèmes faiblement intégrés, tels que Arpanet ou Transpac. Cette approche consiste à compléter des systèmes centralisés non modifiés en leurs ajoutant une nouvelle couche ayant pour rôle d'offrir un mécanisme de communication entre usagers.

Ce choix, même s'il a abouti à des services comme l'accés interactif à des systèmes distants, le transfert de fichiers et le courrier électronique, possède des limitations importantes (cf [Kaiser 84] par exemple). En particulier elle n'offre aucune transparence du réseau et le partage de ressources dans ces systèmes, passe par des actions explicites des usagers, comme par exemple, le transfert explicite des fichiers.

2/ Systèmes fonctionnellement répartis, comme par exemple les systèmes de gestion de fichiers répartis tels que NFS [Sandberg 85] ou ITC [Satyanaray 85]. Cette approche consiste à compléter des systèmes centralisés, en les modifiant partiellement, avec une nouvelle couche permettant aux processus application d'accèder de la même façon un sous-ensemble de ressources (par exemple les fichiers) indépendamment de leur localisation. Cette transparence (et partage) ne s'étend pas aux autres ressources (processeurs et mémoire).

3/ Systèmes d'exploitation répartis (SER), comme par exemple Locus [Popek 81] ou Apollo/Domain [Leach 83]. Cette approche consiste à construire un système d'exploitation réparti unique, s'exécutant sur tous les calculateurs du réseau, offrant un environnement intégré de telle sorte que le réseau et la localisation sont cachés. Ces systèmes privilégient l'échange de messages comme forme de communication et de synchronisation entre processus. Ils sont souvent implantés sur des réseaux locaux à haute performance.

Plusieurs SER essayent d'offrir l'environnement intégré correspondant à l'approche 3/, cf [Stankovic 84] ou [Guillemont 84] par exemple; parmi ceux-ci se situe le système Chorus†, système au sein duquel se sont développés nos travaux de recherche.

2. L'architecture Chorus

Chorus ([Banino 80], [Zimmermann 81], [Guillemont 84a]) est une architecture répartie d'exécution et de communicaton conçue pour une grande variété de machines et de réseaux et une large classe d'applications.

Cette architecture est le résultat des recherches du projet Chorus, menées depuis 1980 à l'Institut National de Recherche en Informatique et Automatique (INRIA). Depuis Juillet 1984 une implantation sur des multi-processeurs SM90 [Finger 81] connectés par un réseau local Ethernet [Metcalfe 78] est disponible. Cette réalisation a été diffusée à plusieurs equipes de recherche. En même temps des études ont été poursuivies afin de l'éprouver et de l'enrichir.

Au printemps 1985 l'équipe Chorus a décidé de compléter cette architecture avec un ensemble de nouveaux serveurs et services pour construire un système complet, le système d'exploitation Chorus, aussi appelé Chorus Version 2, cf [Armand 85, 86].

3. Le système d'exploitation Chorus

Le système d'exploitation Chorus est un environnement pour le développement et l'exécution d'applications réparties. Il résulte de l'intégration de l'architecture Chorus et de l'interface du système Unix [Ritchie 74] ††.

Il hérite de sa version précédente de l'ensemble des concepts sur lesquels son architecture repose, à savoir:

- le concept d'acteur - un processus séquentiel qui exécute des étapes de traitement déclenchées par l'arrivée de messages;
- le concept de porte - permettant aux acteurs d'émettre et de recevoir des messages;
- et le concept de message - des ensembles de données identifiés et protégés que les acteurs échangent.

En effet, Chorus offre à ses clients la communication par messages au travers des portes, la diffusion de messages sur des groupes de portes, l'appel de procédures distantes, etc.

D'autre part, il offre aussi les services caractéristiques de l'interface Unix étendus à la répartition: un système de gestion de fichiers répartis, l'exécution à distance, des groupes répartis de processus , etc., voir [Guillemont 86].

Chorus hérite aussi d'Unix l'environnement de développement disponible sur ce dernier car il peut exécuter un processus Unix comme cas particulier d'acteur.

---

† Chorus est une marque déposée de l'INRIA.

†† Unix est une marque déposée de Bell Labs.

4. Le thème de notre étude

Le thème de notre étude concerne l'analyse des méthodes et des facilités pour la désignation (*naming*) et l'édition de liens (*binding*) offertes par les SER ainsi que leur application dans le cadre du système Chorus.

Un système de désignation doit permettre l'affectation de noms aux ressources disponibles dans le système pour permettre leur désignation, accès et partage. L'espace des noms doit aussi être structuré de telle sorte que l'on puisse établir des relations entre les entités qu'ils désignent.

D'autre part, pour que l'on puisse accèder aux ressources, il faut pouvoir traduire leurs noms en des points d'accès permettant de les manipuler. Le système doit donc fournir les facilités nécessaires pour que les différents processus actifs puissent se connaître mutuellement et établir des liaisons afin qu'ils puissent communiquer ou s'appeler mutuellement.

Ces facilités doivent être offertes de telle sorte que les avantages potentiels de l'environnement réparti soient respectés. En effet, un système de désignation réparti doit répondre aux besoins suivants:

- il doit permettre la désignation, l'accès et le partage de ressources, indépendamment de leurs localisations; en particulier, il doit permettre que l'espace des noms soit structuré en fonction de contraintes administratives et non en fonction de contraintes de localisation;
- il doit permettre que les ressources changent de localisation ou soient dupliquées sans changer de nom;
- il doit garantir que les pannes ne modifient pas la rélation entre les noms et les entités qu'ils désignent;
- il doit enfin permettre qu'une application soit conçue indépendamment du nombre de processeurs disponibles au moment de son exécution, ainsi que la reconfiguration dynamique de l'application pendant son exécution.

Cette thèse propose une hiérarchie de couches de désignation (des portes, processus, groupes de portes, groupes de processus et fichiers) qui répond à ce cahier des charges. Une importance spéciale est donnée par cette proposition au problème de l'édition de liens, chargement, contrôle et reconfiguration d'applications réparties construites sur des réseaux de processus communicants. En particulier, le rôle des groupes de portes et de la diffusion de messages, pour l'édition de liens et le contrôle d'applications réparties, est mis en évidence.

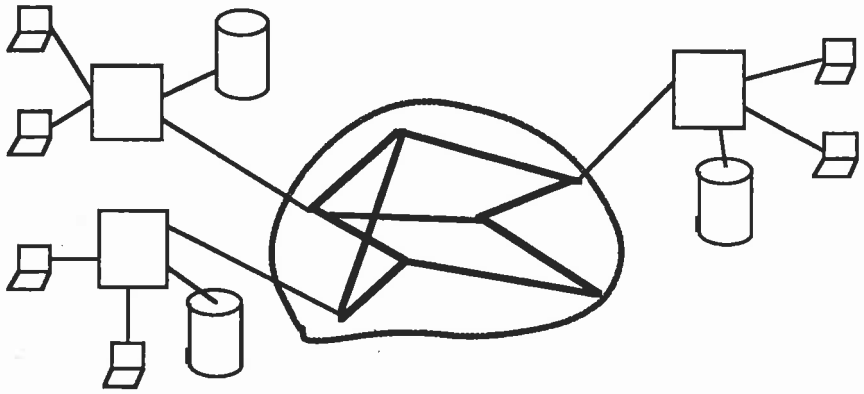
D'autre part, la désignation symbolique des portes et des groupes de portes est introduite par une extension du type des noeuds présents dans l'arbre de fichiers. Cette extension, et un protocole standard de redirection symbolique de requêtes, permettant l'interprétation répartie des noms symboliques, sont simultanément à la base de la construction du graphe réparti de désignation symbolique et du système de gestion de fichiers répartis de Chorus.

5. Plan de l'ouvrage

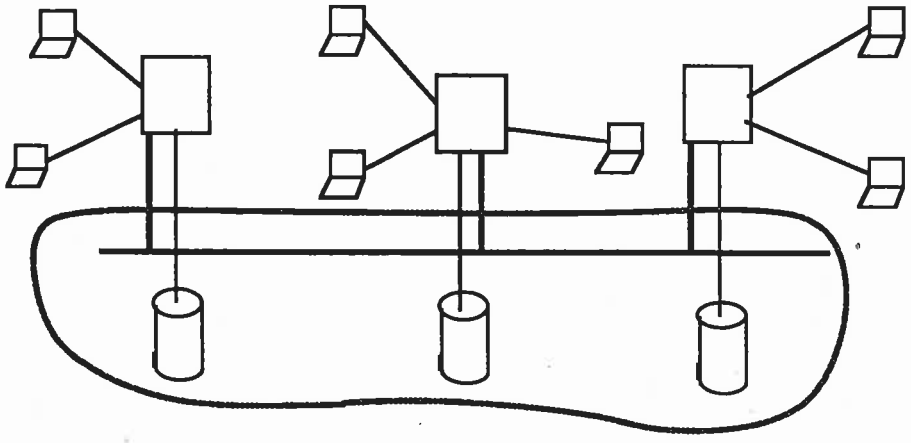
Les chapitres I et II de cet ouvrage présentent, respectivement, les problèmes soulevés par la répartition en matière de désignation et d'édition de liens entre processus communicants. Le chapitre III introduit les aspects essentiels de l'architecture et du système Chorus nécessaires pour comprendre la suite de cet ouvrage. Le quatrième chapitre présente la désignation, la gestion et l'utilisation des portes de communication,

des groupes de portes et des groupes de processus. Dans le chapitre V nous présentons les méthodes proposées par Chorus pour étendre la désignation symbolique d'Unix vers la répartition. La conclusion dresse un bilan des travaux de recherche que nous avons menés.

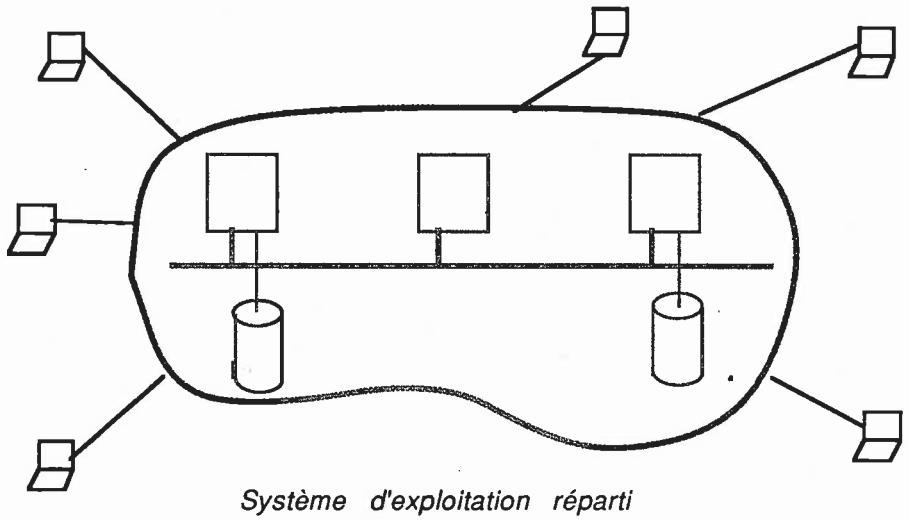
Trois annexes complètent cet ouvrage. L'annexe I donne une liste des abréviations qu'il utilise. L'annexe II contient une description sommaire du système Unix; cet annexe servira au lecteur non familiarisé avec ce système. L'annexe III présente la définition complète d'un ensemble d'appels système Chorus concernés par notre thème d'étude.



Réseau faiblement couplé



Système de gestion de fichiers réparti



Système d'exploitation réparti

Choix pour intégrer la répartition dans un système d'exploitation

CHAPITRE I

LA DESIGNATION EN ENVIRONNEMENT REPARTI

CHAPITRE I

LA DESIGNATION EN ENVIRONNEMENT REPARTI

Les systèmes informatiques sont souvent conçus et implantés comme une hiérarchie d'abstractions: les entités d'une couche étant implantées sur celles des couches inférieures (cf [Brown 84] par exemple). Ils utilisent des mécanismes de traduction de noms pour transformer un nom désignant une entité d'une couche, dans une référence vers l'entité qui la matérialise dans la couche au-dessous. Tel est ce qui se passe au moment de l'édition de liens d'un programme (les symboles sont traduits dans les adresses d'implantation), quand on ouvre un fichier (le nom du fichier est traduit dans une référence vers son déscripteur), quand une adresse émise par le processeur est transformée dans l'adresse réelle par un système de gestion de mémoire, ou quand on veut accéder à une ressource en environnement réparti.

Naturellement, l'environnement réparti introduit de nouvelles couches d'abstraction dans un système et des problèmes nouveaux vis-à-vis des mécanismes et outils de désignation, traduction de noms et d'édition de liens. Afin d'introduire un cadre d'analyse de ces problèmes nous commencerons par distinguer trois niveaux d'observation d'un système réparti†.

1. Notions de Nom, Adresse et Route

Une nomenculature a été introduite [Shoch 78] pour séparer les différents niveaux de manipulation d'une ressource en environnement réparti: un nom de ressource désigne l'entité que l'on veut utiliser; une adresse est une référence interne qui permet de l'accéder (adresser); une route indique comment arriver jusqu'où elle se trouve.

Selon ce modèle, les noms concernent surtout la vision externe, de plus haut niveau; ils sont en général matérialisés par des symboles (des chaînes de caractères avec signification mnémonique) qui désignent une ressource, un usager, un ensemble de ressources, ... Les adresses sont des références internes ou points d'accès aux ressources; elles sont attachées aux entités adressables dans le système et sont interpretées par les mécanismes de communication. Les routes concernent la façon dont le routage est réalisé par le sous-système de transport.

Afin d'introduire le vocabulaire de la désignation nous présentons ensuite l'application de ce modèle d'analyse au service téléphonique.

2. Illustration: le système téléphonique

L'ensemble des utilisateurs du service téléphonique est répertorié dans des annuaires téléphoniques. A ce niveau un nom peut désigner une personne ("J.A.Dupont"), une institution ("INRIA"), un représentant local d'un service national ("Police"), un service ("Medecins généralistes" dans les pages jaunes). D'autres sortes

† Dans [Comer 86] on trouve un traitement formel de cette structuration en couches des noms.

de catalogues coexistent avec ceux-ci: les catalogues des entreprises, les agendas personnels, etc. Ces derniers sont des catalogues privés; ils offrent une flexibilité supplémentaire dans la mesure où ils permettent aux usagers d'avoir des visions contextuelles (selon l'utilisation) du réseau téléphonique.

Les catalogues fournissent une certaine structuration externe des entités qu'ils décrivent (par Pays, par Département, par entreprise, par service, par profession, par relation d'amitié, ...). L'information qu'ils contiennent évolue dans des échelles de temps très différentes: les services publics (pompiers, urgences, ...) ont des numéros très stables, les numéros de téléphone des personnes changent quand elles changent d'adresse postale, le numéro de téléphone associé au nom "ma copine" ("mon copin") peut changer très vite dans l'agenda de certains (certaines), moins vite dans l'agenda d'autres.

Pour obtenir un numéro de téléphone il faut consulter un annuaire et faire l'édition de liens, i. e., traduire un nom dans une adresse. Quelques situations sont à considérer:

- un même numèro peut être partagé par plusieurs noms différents (les membres d'une famille habitant ensemble);
- un même nom peut correspondre à plusieurs numéros différents (une grande entreprise): il faut alors en choisir un;
- un nom peut correspondre à une liste de noms ("Medecins" par exemple), desquels il faut en choisir un et répéter à nouveau l'opération de recherche du numéro;
- afin de trouver le numéro, il faut d'abord chercher le numéro des renseignements téléphoniques et ensuite l'appeler pour obtenir le numéro désiré.

Le réseau téléphonique ne connaît pas la notion d'usager, son domaine se limite aux adresses, c.a.d. les numéros de téléphone désignant des appareils téléphoniques. Les numéros ont une structure hiérarchique qui facilite le routage et leur affectation décentralisée aux abonnés. Chaque ressource (personne, ordinateur, ...) accessible par le réseau téléphonique est représentée par un appareil téléphonique; seuls ces derniers sont accessibles. Nous pouvons les assimiler à la notion de porte réseau, matérialisant un point d'accès à une ressource. Le service téléphonique ne sait que mettre en communication deux appareils, il ignore les personnes qui par ce biais peuvent communiquer. La relation entre les personnes et les appareils n'est pas statique: si une personne change d'adresse postale (si une ressource migre), elle ne conserve pas généralement son numéro, celui-ci sera tôt ou tard réutilisé.

3. Couches de désignation dans les systèmes d'exploitation répartis

Notre expérience d'usagers du système téléphonique nous montre qu'il nous arrive quelquefois d'être obligés de commencer une conversation téléphonique par: "Est ce que je suis bien chez Mr ...". Celà arrive parce que les adresses que nous avons mémorisées ont été réutilisées, ou parce que le catalogue que nous avons consulté n'est plus à jour.

Ces problèmes ne sont pas trop importants car l'échelle de temps dans laquelle évolue notre environnement téléphonique et les informations supplémentaires que nous avons sur nos interlocuteurs les minimisent. Dans un SER (système d'exploitation réparti), l'échelle de temps, d'une part, et les caractéristiques des interlocuteurs, d'autre part, ne permettent pas l'ignorance de ces problèmes. Les adresses réseau, en

tant que couche de désignation, présentent dans cet environnement quelques défauts majeurs vis-à-vis du cahier des charges présenté au §4 de l'Introduction:

- Les adresses ne sont pas transparentes vis-à-vis de la localisation. Une adresse réseau n'est pas indépendante de la localisation de la ressource à laquelle elle donne accès; si une ressource change de localisation, elle doit changer d'adresse.
- Une même adresse est partagée par plusieurs ressources et donc ne suffit pas pour les distinguer.
- Les adresses sont réutilisées. Si une entité a mémorisé une adresse réutilisable, elle n'est jamais sûre que cette adresse donne toujours accès à la même ressource.

Ainsi, à cause de ces défauts, les adresses réseau ne servent pas aux SER comme couche de désignation. Cependant, la couche de désignation symbolique, elle seule, n'est pas non plus suffisante. Par exemple, quand un processus commence la manipulation d'un fichier (quand il ouvre le fichier), il doit fournir son nom symbolique, néanmoins, il serait trop lourd de le donner à nouveau, à chaque fois qu'il aurait à le lire. Un SER identifiant le récepteur de chaque message échangé (assimilable à une phrase d'une conversation téléphonique) par son nom symbolique, serait aussi trop lourd.

Pour résoudre ces deux problèmes, à savoir, l'inadéquation des adresses réseau comme couche de désignation et la lourdeur de l'utilisation systématique des noms symboliques, les SER introduisent entre les deux une nouvelle couche de noms: nous les appelons noms internes ou noms de bas niveau.

Dans un SER nous trouvons donc deux couches essentielles de désignation: la couche de désignation symbolique, qui est constituée par un ensemble de catalogues associant des attributs aux noms des ressources et la couche de désignation interne qui offre des moyens d'accéder aux ressources en cachant leurs adresses.

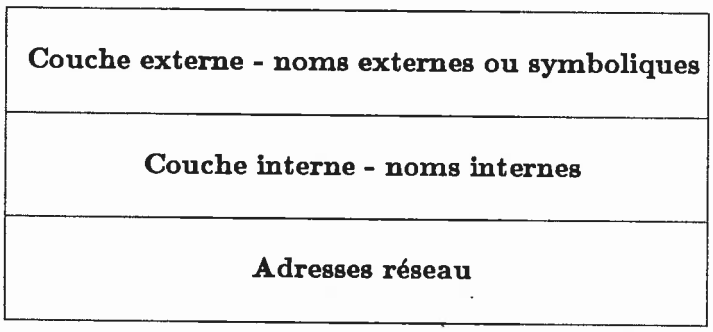


fig. 1.1 - Couches de désignation dans un SER

Nous commençons par analyser les méthodes utilisées pour construire la couche de désignation interne dans les SER.

4. Noms internes dans les SER

Un nom interne se présente comme un identificateur (en général un numéro) qui désigne une ressource ou un groupe de ressources. Nous distinguons essentiellement deux types de noms internes: les noms uniques et globaux et les noms contextuels ou locaux (les deuxièmes étant, en général, complémentaires des premiers comme nous le montrerons par la suite).

4.1. Noms internes uniques et globaux

Un nom interne unique et global est un nom indépendant de la localisation (i. e., transparent à la localisation) et non réutilisé (i. e., qui désignera pour toujours la même entité). Ces noms sont donc uniques dans l'espace et dans le temps. Ils sont souvent appelés identificateurs uniques ou UID (*Unique IDentifiers*).

Par exemple, le système Apollo/DOMAIN [Leach 82, 83] est un SER qui offre un système de gestion de mémoire virtuelle intégrant à un seul niveau et dans un ensemble unique toute la mémoire de masse de toutes les machines du réseau: un processus peut introduire dans son espace d'adressage n'importe quel segment logique de mémoire (objet) quelle que soit sa localisation. Les objets sont identifiés de façon interne par une structure de 64 bits dont 20 bits identifient le site où l'objet a été créé, 32 bits enregistrent l'heure de création de l'objet, et les autres 8 bits enregistrent des informations sur le type de l'objet ainsi désigné.

fig. 1.2 - Un UID dans le système Apollo/DOMAIN

Dans ce système les objets sont aussi désignés symboliquement. Le sous-système de désignation symbolique est capable de traduire un nom symbolique en un UID; une fois qu'il connaît l'UID de l'objet, un processus peut l'introduire dans son espace d'adressage quelle que soit sa localisation (à des restrictions de protection près).

Apollo/DOMAIN utilise aussi des UID pour désigner de façon interne les usagers, des groupes d'usagers, etc. Plusieurs autres systèmes utilisent des noms internes uniques et globaux pour désigner plusieurs sortes d'entités; par exemple, le V-Kernel [Cheriton 84] utilise des noms uniques et globaux pour désigner de façon interne des processus et des groupes de processus; Chorus les utilise pour désigner de façon interne les portes et les groupes de portes; le système DEMOS/MP [Powell 83] les utilise pour désigner des processus. Cette méthode de désignation est aussi souvent utilisée pour construire la désignation interne des fichiers dans des systèmes de gestion de fichiers répartis, voir [Dion 80] et [Radia 83] par exemple.

4.2. Méthodes de génération de noms internes uniques et globaux

Les identificateurs uniques et globaux sont toujours générés par le système à la création de l'entité qu'ils désignent. Il y a essentiellement deux méthodes pour les produire [Liskov 81]:

- les produire par un service centralisé - méthode abandonnée à cause de son manque de souplesse, surtout en cas de panne;
- et les produire par une méthode décentralisée - cette méthode consiste en l'introduction dans le nom d'une estampille du créateur et d'une estampille unique par rapport au créateur; souvent, de telles estampilles sont le numéro du site de création et un numéro unique et non réutilisé par rapport à ce site.

La génération d'une estampille unique de site peut être faite par administration manuelle, en utilisant le numéro de série du processeur (c'est le cas de Apollo/DOMAIN) ou en utilisant un générateur de nombres aléatoires (c'est le cas du V-Kernel). La génération d'estampilles uniques par site peut aussi être faite par plusieurs méthodes: en utilisant l'horloge de la machine (c'est le cas de Apollo/DOMAIN et de Cedar [Birrel 84] par exemple) ou en utilisant un générateur de nombres aléatoires (c'est le cas du V-Kernel).

Aucune de ces méthodes n'est capable de garantir une unicité absolue des noms [Liskov 81]. Toutes les implantations passent donc par l'utilisation d'un nombre assez important de bits afin de diminuer la probabilité de réutilisation.

4.3. Motivations pour l'utilisation de noms internes uniques et globaux

Les noms uniques et globaux ont des propriétés intéressantes:

- Ils sont intrinsèquement indépendants de la localisation: si un processus a mémorisé un nom global d'entité, il n'a pas à mémoriser sa localisation; l'entité peut se déplacer dans le système en conservant son identité.
- Ils sont non réutilisables: si un processus a mémorisé un nom global et unique d'entité, il n'a pas à confirmer à chaque accès que le nom désigne toujours la même entité. En effet, cette propriété rend possible la mémorisation pendant longtemps d'un nom unique; à l'occasion d'un nouvel accès, soit l'entité existe et possède ce nom, soit elle n'existe plus et il n'y aura pas d'ambiguité.
- Ces noms peuvent être passés d'un processus à l'autre ou d'un site à l'autre, sans avoir besoin de les traduire, car leur signification est indépendante de la localisation.

Ces propriétés rendent l'utilisation de noms globaux intéressante. Par exemple, on peut à l'aide de ces noms introduire des liens entre fichiers qui traversent les frontières des volumes, et même des sites, sans besoin de mécanismes supplémentaires de traduction de noms; on peut faire migrer un processus ou une porte de site sans avoir à traduire toutes les références qui les pointent dans les autres sites; on peut déplacer des volumes démontables ou des périphériques d'un site tombé en panne, vers un site actif, sans avoir à traduire les références qui les pointent et en étant sûr que l'on accède toujours aux mêmes entités; les processus peuvent s'échanger des noms globaux et uniques dans des messages, ces noms conservant leurs significations.

Il y a des systèmes qui utilisent des UID mais qui ne permettent pas la migration d'entités. Ces systèmes ne mettent en valeur que quelques-uns des avantages énoncés ci-dessus. Dans ces systèmes, il n'y a pas de problèmes de localisation car le nom d'une entité contient le nom de son site de création, qui est aussi son site de résidence.

Les systèmes qui mettent en valeur tous les avantages de la répartition doivent permettre la migration dynamique d'entités. Cette fonctionnalité est exploitée par plusieurs SER. Par exemple, le système Apollo/DOMAIN [Leach 83] permet la migration des fichiers, DEMOS/MP [Powell 83] et le V-Kernel [Theimer 85] permettent la migration des processus, Chorus, voir le chapitre IV, permet la migration des portes.

L'interêt de la migration est multiple, par exemple:

- La répartition de la charge - les différents processus peuvent se répartir dynamiquement par les différents sites, en fonction de leur charge;
- La résistence aux pannes - si un site tombe en panne, on peut transférer sur un autre site les ressources ou les processus qu'il gérait.

Pour que la migration soit intéressante, il faut qu'elle soit cachée aux clients et/ou partenaires de l'entité nomade. Celà concerne deux aspects: le transfert le l'état des entités et leur localisation dynamique par leurs partenaires. Nous illustrons ci-dessous la méthode utilisée par DEMOS/MP pour cette localisation.

4.4. Illustration: désignation et localisation dans le système DEMOS/MP

DEMOS/MP [Powell 83] est un SER basé sur la communication par messages. Un noyau implémente dans chaque site les opérations primitives du système: l'exécution de processus, la communication entre processus (locaux ou distants) par échange de messages et des canaux de communication appelés liens. Les processus DEMOS peuvent migrer.

Les messages sont échangés de processus à processus par le biais des liens. Un lien est essentiellement un canal de communication unidirectionnel qui relie le processus émetteur au processus récepteur. A chaque processus est associée une file d'attente qui reçoit tous les messages envoyés via tous les liens qui ont ce processus comme récepteur. Les liens sont donc fondamentalement une forme d'adressage de processus.

Un lien est désigné chez l'émetteur par un numéro d'entrée dans une table qui contient, parmi d'autres informations, le nom interne unique et global du récepteur. Il existe une de ces tables par processus.

Un processus DEMOS/MP est donc caractérisé par un programme, des données, un contexte d'exécution, une file d'attente et une table de liens ou contexte d'adressage pour la communication.

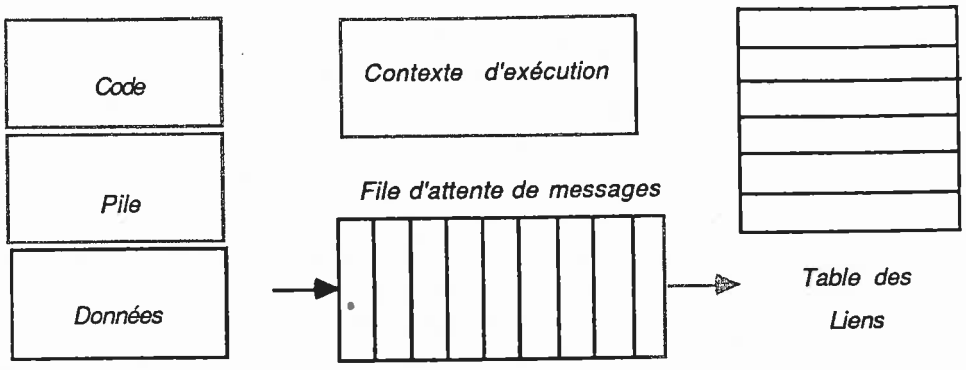


fig. 1.3 - Composants d'un processus DEMOS/MP

Associé à chaque entrée dans une table de liens il y a le nom interne global et unique du récepteur et son dernier site de résidence connu. Le nom global et unique (UID) associé à un lien est immuable. Le nom du site de résidence (dernier connu) peut changer car il ne joue que le rôle d'une sorte de suggestion (*hint*).

La table des liens joue, du point de vue de la communication, le même rôle qu'une table de pages en mémoire du point de vue de l'adressage de la mémoire. Quand un message est adressé via un lien, il est envoyé au processus récepteur en étant acheminé vers son dernier site de résidence connu. Nous pouvons faire une analogie avec le processus de traduction d'adresses dans les systèmes à mémoire virtuelle.

partie immuable	*partie modifiable*
UID du récepteur	Dernier site de résidence connu (*hint*)

fig. 1.4 - Entrée dans une table d'adressage

Initialement, quand un processus est créé, toutes les entrées dans une table de liens qui le référencent ont comme suggestion de résidence (*hint*) le site de création. Les messages envoyés via ces liens sont envoyés vers ce site et ils aboutissent.

Quand ce processus migre, le système transfère vers le nouveau site de résidence tous ses composants. Cependant, il garde dans le site de création une indication d'indirection signalant le nouveau site de résidence du processus. Cette

indication d'indirection est un résidu de l'état du processus migré; elle contient exactement les mêmes informations qu'une entrée dans une table de liens, i. e., l'UID du processus et une suggestion (*hint*) sur le nouveau site de résidence.

Quand un message adressé au processus qui a migré arrive au site de création, le système ne trouve que le résidu; le message est alors redirigé vers le nouveau site de résidence. Le processus peut ainsi continuer à recevoir les messages qui lui sont adressés, bien que la seule partie du système qui sache qu'il a migré soit le site de création.

Cependant, pour des raisons de performance, le site qui réalise la redirection signale au site de l'émetteur que le processus vient de migrer. Le site de l'émetteur peut ainsi mettre à jour la suggestion de résidence dans les différentes tables de liens où le processus ayant migré figure. Naturellement, les liens créés après la migration seront à jour (du moins au moment de leur création).

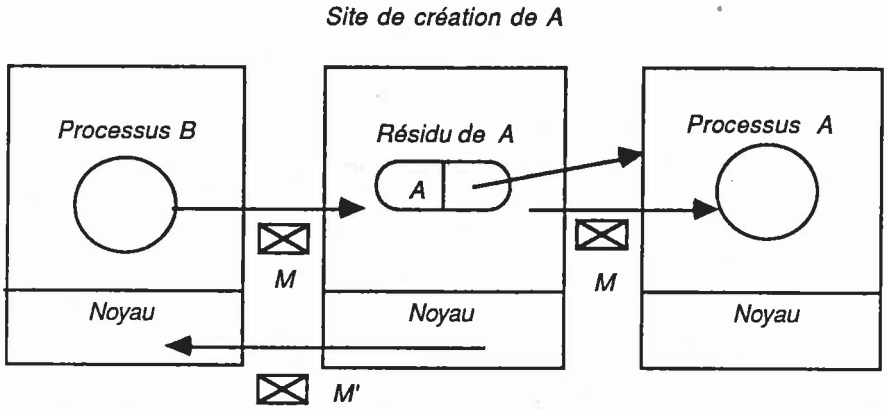


fig. 1.5 - Redirection de messages et mise à jour des suggestions de localisation

Cet algorithme réparti de localisation dynamique du récepteur d'un message est bâti sur le fait que le système contrôle la visibilité des UID en les cachant derrière des noms internes contextuels ou locaux (i. e., une entrée dans la table des liens du processus). L'objectif qui peut présider à l'introduction de cette indirection supplémentaire entre l'émetteur et les adresses réseaux ne se résume pas à celui que nous venons d'illustrer. En effet, le visibilité directe des noms internes présente plusieurs inconvénients.

4.5. Insuffisances de la visibilité directe des noms internes uniques et globaux

Si les clients du système peuvent enregistrer dans leurs données les UID, le système peut ne pas être capable de contrôler les noms connus d'un processus. Cependant, comme nous l'avons vu ci-dessus, le système DEMOS/MP avait besoin de contrôler les noms internes connus d'un processus pour localiser le récepteur d'un message.

D'autres raisons existent pour ne pas donner la visibilité directe des UID aux clients du système:

1/ Si dans un système, la connaissance de l'UID d'une entité est condition suffisante pour pouvoir lui adresser un message (tel est le cas du V-Kernel par exemple), il n'y a aucun contrôle du droit d'émission sur une entité, car connaissant la structure des UID, rien n'empêche un processus de générer tous les noms possibles. L'espace d'adressage des messages est donc non protégé.

2/ Si les noms peuvent être passés librement d'un processus à l'autre, sans que le système contrôle "qui connaît qui", il est moins simple de construire des fonctionnalités de notification d'exceptions signalant aux processus qu'une entité qu'ils connaissent vient d'être détruite ou est inaccessible.

3/ Un UID est une information enregistrée sur un certain nombre d'octets. Si au cours de la vie du système ce nombre d'octets doit varier (par élargissement du nombre ou de la taille de ses champs par exemple), un problème supplémentaire de portabilité se pose.

4/ Finalement, comme il en sera discuté au chapitre suivant à-propos de l'édition de liens, le contrôle par le système des noms connus par un processus facilite l'implantation de l'édition de liens au chargement et la reconfiguration de programmes.

Une façon d'éviter ces problèmes consiste en l'introduction d'un contexte de désignation interne par processus, contrôlé par le système, et qui cache les UID (ou les adresses) sous des noms internes contextuels ou locaux comme le fait DEMOS/MP. Ces contextes constituent une nouvelle couche de désignation interne qui permet la traduction des noms locaux en UID (ou dans des adresses) et vice versa. Ces espaces d'adressage contextuels sont de la même nature que les espaces protégés d'adressage de segments mémoire (ou d'objets).

Accent [Rashid 81] est un système qui exploite les avantages de la désignation interne contextuelle. Son analyse nous permettra de les mettre en évidence.

4.6. Illustration: désignation interne contextuelle dans le système Accent

Accent est un noyau de SER qui a été développé à l'Université de Carnegie-Mellon. Il permet l'exécution de processus qui communiquent entre eux en échangeant des messages au travers de portes. Accent utilise la méthode contextuelle pour désigner de façon interne les portes.

Une porte Accent est un objet système protégé dans lequel des messages peuvent être mis par un processus et à partir duquel des messages peuvent être retirés. Tous les services et fonctionnalités offerts par un processus se présentent aux autres processus derrière des portes.

A chaque porte sont associés trois droits:
- le droit "possession" (de la porte),
- le droit "réception" (de messages),
- et le droit "émission" (de messages).

Pour pouvoir manipuler ou adresser une porte, un processus doit avoir une capacité ayant les droits adéquats associés. Cette capacité est un nom contextuel de la porte, tel que dans Unix (cf annexe II) un descripteur de fichier ouvert est un nom contextuel d'un objet géré par le système. Ce nom n'a de sens que s'il est utilisé par le processus qui le connaît. Bien que les droits associés à un nom puissent varier, à un instant donné, un processus ne possède qu'un nom contextuel par porte qu'il connaît.

Un processus crée une porte par l'opération *AllocatePort,* laquelle retourne le nom de l'objet créé, suite à l'appel. Initialement, le créateur possède tous les droits sur la

porte. Les droits peuvent être transmis aux autres processus dans des messages. Possession et droit de réception sont des droits qui peuvent être transmis mais qui ne peuvent en aucun cas être partagés. La distinction entre ces deux droits a pour but d'autoriser un processus à reprendre à son compte un service rendu auparavant par un autre. Cette possibilité permet la reconfiguration dynamique des applications et du système.

Un processus relache l'accès à une porte par le biais de l'opération *DeallocatePort (nom)*. Si l'appelant a les droits possession et réception, la porte est détruite et tous les processus qui possèdent le droit émission reçoivent du système des messages d'urgence qui leur notifie de la non validité des noms locaux correspondants.

Si le processus qui relache l'accès à la porte possède le droit réception, mais pas le droit possession, le processus qui a le droit possession reçoit le droit réception dans un message d'urgence (et vice-versa).

Cette méthode de désignation contextuelle est à la base de la protection et du traitement d'exceptions liées à la destruction ou à l'inaccessibilité soudaine des portes. Ceci est possible parce que le système contrôle complètement qui connaît qui. Il s'agit d'une méthode de désignation et de protection de la même nature que celles qui utilisent les systèmes centralisés à capacités.

Cependant, elle allourdit le système car elle exige qu'il soit capable de faire, à chaque communication entre processus, la traduction entre les noms qui traversent les contextes.

Dans Accent, chaque fois qu'un processus envoie un message à un autre processus, le système effectue une double traduction:

nom local de l'émetteur → nom global

nom global → nom local du récepteur.

Les messages Accent sont typés. C'est ce typage qui est à la base du transfert de noms et de droits entre processus. Un processus ne peut donc cacher un nom, ni dans ses données, ni dans un message.

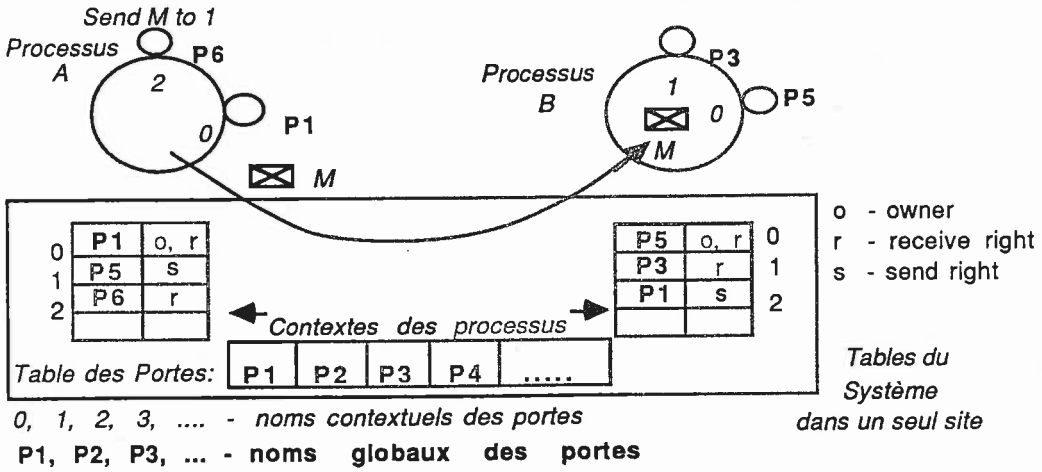


fig. 1.6 - Désignation des portes dans Accent

Accent offre aussi des fonctionnalités pour l'édition de liens au chargement et pour la mise au point. Ces fonctionnalités sont bâties sur la manipulation des contextes de désignation interne d'un processus par un autre processus. Elles seront discutées au chapitre II.

L'implantation de l'ensemble des fonctionnalités offertes par Accent en ce qui concerne les portes, s'avère simple dans un seul site, i. e., en local. Cependant, l'implantation de ces fonctionnalités dans un réseau n'est pas immédiate. En particulier, le contrôle et le transfert des capacités et la notification des exceptions dans le réseau, exigent des méthodes d'une lourdeur non négligeable. Nous y reviendrons au chapitre IV.

4.7. Conclusions sur la désignation interne dans les SER

Les SER utilisent des noms internes uniques et globaux (UID), cachant les adresses réseaux, pour désigner les entités. Ces noms peuvent être directement visibles des clients ou cachés sous des noms internes contextuels.

L'utilisation de noms uniques et globaux offre la transparence du réseau, la possibilité de faire migrer les entités, une méthode simple de détecter a posteriori l'absence d'une entité (comme le nom est unique, i. e., non-réutilisé, il n'y a jamais d'ambiguité de désignation), et permettent que les noms soient passés d'un processus à l'autre sans avoir à les traduire.

Pour des raisons de protection ou de portabilité, afin de faciliter la localisation et l'héritage de noms et de faciliter le traitement d'excéptions, les processus ne peuvent avoir qu'accès à des noms internes contextuels ou locaux. Ces noms ne peuvent être passés d'un processus à l'autre que s'ils sont traduits entre contextes. Ces espaces d'adressage sont de la même nature que les espaces protégés d'adressage d'objets dans les systèmes conventionnels.

Les couches de désignation présentes dans les SER sont donc celles qu'illustre la figure 1.7.

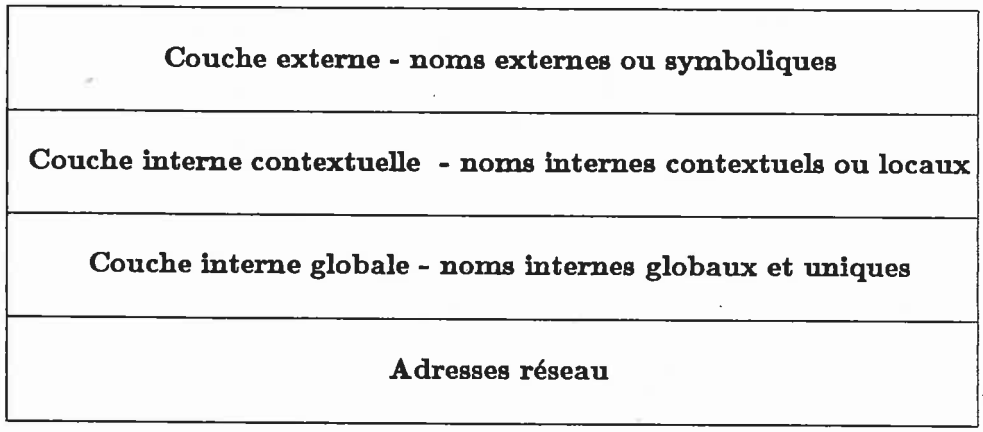


fig. 1.7 - Couches de désignation dans un SER

5. Couche de désignation externe ou symbolique

Comme nous l'avons vu ci-dessus, les noms internes sont toujours calculés par le système. Cependant, les usagers du système doivent pouvoir imposer des noms de leur choix aux entités qu'ils créent. Cette possibilité est offerte par la couche de désignation symbolique: cette couche de désignation joue un rôle central dans la facilité d'emploi d'un système. Elle est un sous-système, le sous-système de désignation symbolique, qui permet l'affectation de noms symbolique aux ressources, aux services et aux usagers, et offre les moyens de traduire ces noms en des références ou points d'accès (des noms internes) aux entités qu'ils représentent.

Un système n'offrant pas de fonctionnalités de désignation symbolique serait comparable à un langage de programmation qui n'offrirait que des adresses binaires comme moyen de désignation. En effet, la désignation symbolique est à la base: de la lisibilité, car les usagers préfèrent utiliser des chaînes de caractères, avec une signification précise, à des numéros dont la structure leur est cachée; de l'édition de liens, car celle-ci exige une indirection supplémentaire entre une entité logique et l'entité qui la matérialise à un moment donné; de l'établissement de relations entre entités en structurant la vision externe offerte par le système.

Caractérisation d'un système de désignation symbolique

Un système de désignation symbolique (SDS) est un ensemble de catalogues, ou une base de données, enregistrant des noms symboliques, associant des attributs ou propriétés à ces noms et établissant des relations entre eux.

Les fichiers "/etc/passwd" et "/etc/group" d'Unix (cf annexe II), par exemple, sont un SDS centralisé élémentaire† de désignation d'usagers. En effet, ces fichiers enregistrent des noms d'usagers, leur associent des attributs (numéro d'usager, numéro du groupe d'usagers, catalogue initial et interpréteur de commandes) et structurent leur

† La plupart des caractéristiques fonctionnelles externes d'un SDS s'appliquent aussi bien en environnement distribué que centralisé.

espace en groupes d'usagers.

De façon plus formelle, un SDS est un graphe, le graphe de désignation, dont toutes les arêtes sont étiquetées par des identificateurs, i. e., par des symboles ou chaînes de caractères (par exemple, "marc", "jose", ".profile", "teste.p", "laser", "vax11", "name-service", "hosts", ...). Les sommets ou noeuds de ce graphe sont les descripteurs des entités qu'il désigne: le descripteur d'une entité caractérise ses attributs ou propriétés.

Par exemple, l'attribut "porte" du descripteur d'une imprimante enregistre la valeur de son UID (son nom interne), permettant d'obtenir le nom d'une porte du serveur de cette imprimante; le descripteur d'un fichier†† enregistre diverses propriétés du fichier (type, date de création, taille, attributs de protection, des informations de représentation interne permettant de l'accéder, etc.).

Parmi les attributs associés à l'entité dans son descripteur on trouve souvent des adresses ou des noms internes permettant de l'accéder. Cependant, du point de vue de la désignation symbolique, une entité est assimilée à son descripteur. Le rôle spécifique du SDS s'arrête là.

Parmi les différents noeuds du graphe de désignation nous pouvons distinguer essentiellement deux catégories: celle des noeuds ayant des attributs que le SDS interprète et celle des noeuds dont le SDS ne manipule les attributs que pour le compte de ses clients, sans les interpréter. Parmi les premiers on trouve les catalogues de désignation (une fonction qui applique des identificateurs dans des descripteurs), les liens symboliques (dont un des attributs est un nom que le SDS doit ré-interpréter) et les groupes symboliques (un ensemble de noms symboliques).

Un nom symbolique d'une entité est une construction syntaxique qui dénote son descripteur. Cette construction syntaxique spécifie un chemin dans le graphe de désignation par le biais du couple: (Catalogue, Chemin d'accès). Le catalogue peut être donné explicitement ou implicitement; il spécifie où le chemin commence. Le chemin d'accès est une liste d'identificateurs (étiquettes des arêtes du graphe de désignation).

Une entité possède autant de noms dans un SDS, que de paires:

(Catalogue[i], Chemin d'accès[j])

conduisant à son descripteur, existantes dans ce SDS.

L'interprétation ou traduction d'un nom permet d'obtenir un descripteur à partir d'un nom. En environnement réparti cette traduction est un calcul réparti. Formellement, la fonction d'interprétation ou traduction applique le couple (Catalogue, Chemin d'accès) dans un descripteur.

(Catalogue[i], Chemin d'accès[j]) → Descripteur

fig. 1.8 - La fonction de traduction

†† Par exemple, le descripteur d'un fichier dans un système Unix (*i-node*) voir l'annexe II.

Comme nous l'avons signalé ci-dessus, un SDS possède un rôle important dans la structuration de l'espace des ressources offertes par le système à ses clients. Ceci concerne, pour l'essentiel, les conventions de désignation qu'il adopte.

6. Conventions de désignation symbolique

Les SDS des premiers systèmes d'exploitation centralisés comportaient un espace d'identificateurs structuré rigidement, avec des conventions syntaxiques peu souples. Afin d'augmenter leur souplesse, des notions telles que celles d'espace structuré de noms, avec des conventions syntaxiques régulières et de catalogue par défaut, sont devenues postérieurement des possibilités standard. Elles sont aussi disponibles dans les SDS des systèmes répartis. Fondamentalement, on distingue les types de noms symboliques suivants:

A) Noms globaux

Un nom est global si et seulement si son interprétation est (logiquement) toujours faite à partir du même catalogue indépendamment du contexte où il est interprété. Cette approche consiste à privilégier un catalogue du graphe de désignation et (logiquement) commencer toujours l'interprétation des noms globaux à partir de ce catalogue. Ce catalogue s'appelle souvent "catalogue racine" (*root node*). La spécification du catalogue où commence l'interprétation d'un nom global est donc implicite. Par exemple, dans le réseau Arpanet, une chaîne de la forme "user@host", par exemple "reagan@white-house", est un nom, équivalent au chemin d'accès ("white-house", "reagan") qui est toujours interprété par rapport au catalogue des sites de ce réseau.

B) Noms contextuels

Un nom est contextuel, si et seulement s'il peut dénoter différents descripteurs selon le contexte où il est interprété. Le catalogue où commence l'interprétation doit être indiqué par le client de façon explicite ou implicite. L'indication est implicite si le système où s'intègre le SDS maintient une notion de session: associé à cette session il y a un contexte de désignation symbolique du client qui spécifie le catalogue où commence l'interprétation des noms donnés par ce client.

Par exemple, le système de messagerie éléctronique du réseau "UUCP" (*unix to unix copy*) introduit la convention suivante: le nom du récepteur d'un message est de la forme:

"host1!host2!host3!...!hostN!user"

où "host1", "host2", ..., "hostN" sont des identificateurs de sites, "user" est un identificateur d'usager, relatif au site "hostN", et l'interprétation du nom commence au catalogue des sites du site de l'émetteur. Dans ce réseau le catalogue des sites d'un site A regroupe les identificateurs des sites avec lesquels le site A a une liaison réseau directe.

C) Noms synonymes

Un SDS comporte des noms synonymes si et seulement si deux noms, partant du même catalogue, sont traduits dans le même descripteur par des chemins d'accès différents. Cette fonctionnalité est introduite dans les SDS par des mécanismes tels que les liens, les synonymes (nick names), les alias et les abréviations. Elle sert surtout dans un souci de souplesse dans l'usage des noms.

Nous venons de classer les différents types de noms symboliques. Néanmoins, les SDS peuvent aussi se regrouper dans des classes, suivant les types de noms symboliques qu'ils utilisent†:

- SDS où tous les noms sont contextuels,
- SDS où tous les noms sont globaux, et
- SDS mixtes.

Les systèmes mixtes mélangent des noms globaux et des noms contextuels; par défaut, les noms dans ces systèmes sont contextuels; néanmoins, des syntaxes d'exception permettent de spécifier que le nom doit être interprété par rapport au "catalogue racine"††. Chacune de ces classes peut comporter, ou non, des noms ambigus.

Comme nous l'avons montré à-propos de la discussion sur la désignation interne, la notion de nom global, indépendant de la localisation, est très importante pour les SER dans la mesure où elle permet la désignation des entités indépendamment de la localisation. Comment introduire ces noms en environnement réparti? D'autre part, comme nous le savons, à partir de l'expérience des systèmes centralisés, un graphe de désignation est toujours obtenu par interconnexion de sous-graphes. La manière qui s'avère, à première vue, la plus simple pour interconnecter les sous-graphes en environnement réparti consiste en l'introduction d'un nouveau catalogue, le "catalogue réseau" ou "racine réseau".

6.1. Interconnexion de graphes de désignation par un catalogue réseau

Le réseau Arpanet illustre cette approche. Le catalogue des sites joue le rôle de catalogue réseau; il regroupe les catalogues des usagers des différents sites.

On trouve aussi cette approche dans les systèmes de gestion de fichiers répartis, compatibles avec Unix, qui utilisent le concept de "racine réseau". Dans ces systèmes, un chemin d'accès avec la syntaxe Unix standard (cf annexe II) est interprété contextuellement. Par exemple:

test.p	est interprété par rapport au catalogue courant de l'usager,
/users/jose	est interprété par rapport à la racine courante de l'usager (souvent la racine de l'arbre de son site).

Cependant, afin de pouvoir désigner un fichier par rapport à la racine réseau, ces systèmes introduisent des syntaxes d'exception, comme par exemple:

/../machine1/users/jose	(Newcastle Connection [Brownbridge 82])
//machine1/users/jose	(Apollo/DOMAIN [Leach 83])
machine1:/users/jose	(Ibis [Tichy 84])

† Quelques auteurs considèrent aussi les noms uniques et globaux, sans structure, comme une classe de noms au même titre que celles que nous venons de présenter.

†† Remarquons que la désignation de la racine Unix par le caractère "/", cf annexe II, qui sert aussi de séparateur des identificateurs d'un chemin d'accès, est une exception syntaxique (très élégante quand même).

%machine1/users/jose (Kayak [Radia 83])

qui sont autant de conventions syntaxiques spécifiant que le chemin d'accès ("machine1", "users", "jose") doit être interprété à partir du "catalogue réseau". Ces noms sont des noms globaux dans ces systèmes.

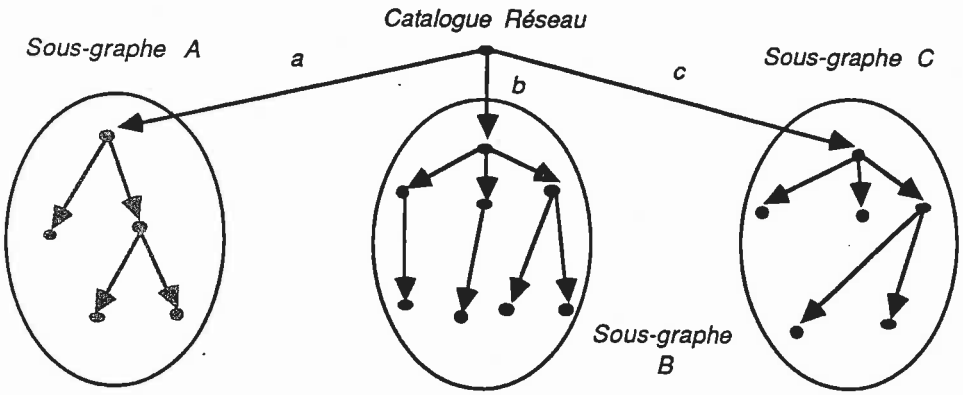


fig. 1.9 - Interconnexion de graphes par un catalogue réseau

Un nom symbolique global est un nom unique dans l'espace†. Ces noms présentent l'avantage d'être interprétés indépendamment d'un contexte (et donc de la localisation) ce qui permet leur passage d'un contexte à l'autre sans traduction. Ils se prêtent donc à cacher la localisation et suggèrent la transparence du réseau.

Cependant, l'approche "catalogue réseau" est caractérisée par le fait d'offrir une vision externe non unique. En effet, bien qu'offrant des noms globaux pour désigner les entités, la structuration de la vision externe offerte est délimitée par les frontières des sous-graphes, i. e., par les frontières des sites ou des serveurs. Ceci peut s'avèrer une limitation importante pour la reconfigurabilité du système et pour le partage de ressources. En particulier, ces noms globaux contiennent une indication logique de la résidence de l'entité, ce qui suggère qu'une entité doit changer de nom si elle change de site†. C'est souvent ce qui arrive si un fichier migre ou si un usager change de site de *login*.

D'autre part, cette approche est caractérisée par l'introduction de syntaxes d'exception (les préfixes "//", "/../", "%" ou "machine:" par exemple). Les syntaxes d'exception se sont révélées comme une limitation importante au transport de logiciel (cette limitation peut être tempérée par l'introduction de liens symboliques et d'abréviations) et une complication dangereuse de la fonction de traduction, cf [Pike 85].

† La notion de nom symbolique unique dans le temps n'a pas de sens car ces noms sont proposés par les clients et non calculés par le système.

† Ceci n'est pas obligatoire si l'entité est aussi identifiée par un nom interne unique et global et que le système est capable de la localiser à partir de son UID.

Finalement, la logique de cette approche conduit à un piège. Les noms globaux ne sont jamais globaux en absolu! Ces noms ne sont globaux que par rapport à un domaine bien précis, le domaine d'adressage symbolique des systèmes concernés; or, il arrive souvent que les SER soient interconnectés entre eux par des réseaux généraux. La notion de globalité disparaît alors complètement et n'est que mythique en face des besoins de communication et de partage des ressources qui traversent les frontières des SER. Naturellement, on ne peut pas introduire de successives syntaxes d'exception jusqu'à être capable de désigner une potentielle "racine inter-planétaire".

A cause de ces problèmes, un autre choix est alors considéré.

6.2. Interconnexion arbitraire de graphes de désignation

Cette autre approche consiste en l'introduction de "références" dans n'importe quel catalogue d'un sous-graphe qui référencent des catalogues (ou la racine) d'un autre sous-graphe. Ces références sont introduites à la demande et généralement de façon non coordonnée.

Ce choix est celui qui est pris par "UUCP" pour désigner les usagers (voir ci-dessus). Il est aussi pris par les systèmes de gestion de fichiers répartis, compatibles Unix, et bâtis sur la notion de "montage distant", comme par exemple NFS [Sandberg 85] et Unix V8 [Winberger 84]. Dans ces systèmes tous les noms ont la syntaxe conventionnelle Unix et sont toujours contextuels, i. e., toujours relatifs au catalogue courant ou à la racine courante (en général celle du site du client).

Cependant, ils permettent la désignation des fichiers distants dans la mesure où ils permettent qu'un noeud d'un arbre référence un catalogue d'un arbre distant, ceci grâce à l'extension de la notion de montage de volume vers la répartition (*remote mount*).

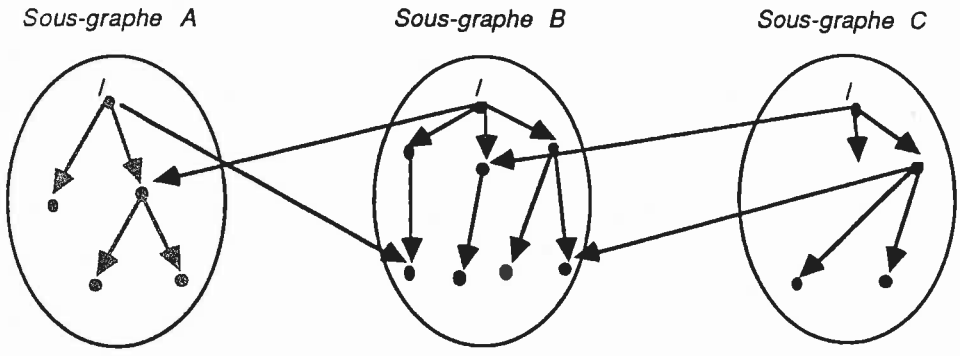


fig. 1.10 - Interconnexion arbitraire de graphes de désignation

Cette approche est caractérisée par le fait d'offrir une vision externe non unique, avec administration décentralisée et ne comportant pas de noms globaux. Les liaisons entre les différents sous-graphes s'établissent dynamiquement en fonction des besoins de communication et de partage de ressources des différents groupes d'usagers des différents sites.

Le fait que tous les noms soient contextuels soulève alors un problème d'ambiguité. Cette ambiguité peut être tempérée par des conventions externes au SDS, c.a.d., par une entente cordiale entre les usagers, soutenue par des conventions administratives communes.

Par exemple, dans le réseau "EUNET", sous-réseau "UUCP" d'Europe, tous les noms de la forme:

"mcvax!host1!host2!...!user"

sont par convention administrative "globaux" en Europe car la machine "mcvax" est un noeud de "référence" de ce réseau: tous les sites ont une liaison réseau (directe ou indirecte) avec ce site.

Pour adresser de façon univoque un usager de ce réseau, il suffit, par exemple, de concaténer un chemin d'accès partant de la machine de l'émetteur, et conduisant à la machine "mcvax", avec un chemin d'accès qui conduit de "mcvax" jusqu'à la machine du récepteur. Par exemple:

"chorus!inria!vmucnam!mcvax!ukc!man.cs.ux!neil"

Ce style de conventions peut être aussi introduit dans la gestion des fichiers. Le système de gestion de fichiers d'Unix V8 suggère que les montages distants soient regroupés dans le catalogue "/n" (abréviation de "/net") de chaque machine. Si par convention administrative, le catalogue "/n" de chaque machine regroupe des identificateurs qui "référencent" les racines des autres machines (cf figure 1.11) et si l'on force administrativement tous les catalogues "/n" de toutes les machines d'un réseau à être cohérents, les noms commencés par "/n" sont administrativement globaux dans ce réseau†.

† Ces conventions ne sont pas en réalité imposées par ce système. En effet, il n'empêche pas l'introduction des montages distants dans n'importe quel catalogue et n'oblige non plus la cohérence des catalogues "/n".

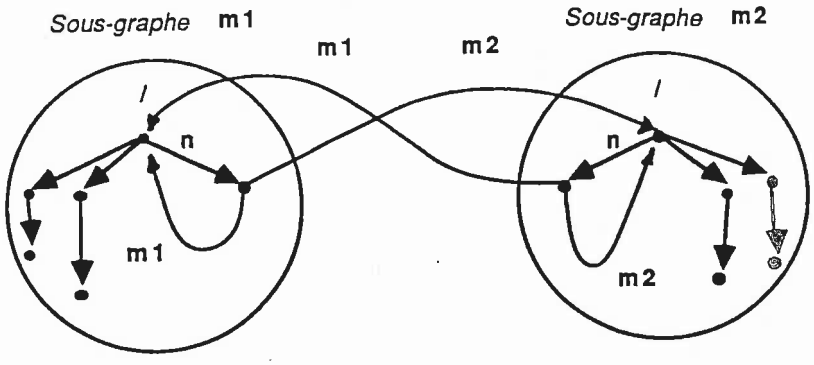


fig. 1.11 - Montages distants dans le catalogue "/n"

Ces exemples montrent que l'ambiguité introduite par la contextualité peut être levée par le biais de conventions administratives. Ces noms (administrativement) globaux, ne sont pas totalement indépendants de la localisation (tel que ceux proposés par l'approche "racine réseau") et présentent exactement les mêmes défauts: ils limitent la configuration externe du système par les frontières explicites des sous-graphes de désignation et ils obligent les entités à changer de nom si elles migrent. Ces limitations peuvent aussi être partiellement tempérées par l'usage de liens symboliques et d'abréviations.

Cependant, cette approche a l'avantage de n'introduire aucune nouvelle syntaxe d'exception, ce qui facilite la réutilisation du logiciel. D'autre part, la logique propre à cette approche comporte l'interconnexion du nouveau graphe de désignation avec d'autres graphes. Rien n'empêche, par exemple, de monter sur le catalogue "/super-net" la racine d'un autre système complètement étranger au domaine "intégré" courant.

Dans un environnement de stations de travail faiblement intégrées, où le partage intensif de ressources n'est ressenti que par des sous-groupes de systèmes géographiquement proches, cette approche est assez réaliste parce que la charge administrative commune est réduite (à part les problèmes de protection que nous discuterons plus tard). Cependant, les SER s'adressent surtout à des environnements avec un degré important de partage de ressources. Cette approche est dans cet autre environnement assez incomplète et légère. Son application dans un environnement qui exige des niveaux importants de partage, d'intégration et de transparence, n'est réaliste que si le système réussit à automatiser partiellement les tâches administratives: en garantissant que les catalogues "/n" sont effectivement cohérents et reflètent l'état actuel du système, en fournissant des moyens d'archivage sûr, en fournissant des utilitaires de diffusion de nouvelles versions de logiciel, etc.

Un niveau important de partage de ressources, la souplesse d'une administration décentralisée mais cohérente et la possibilité de structurer la vision externe du système, indépendemment des frontières des sites ou des serveurs, exigent l'introduction de SDS qui reflètent un espace unique de noms, dont la structuration n'est limitée que par les besoins administratifs, sans contraintes de localisation, d'accès, de partage ou de

migration.

6.3. Systèmes logiquement uniques ou logiquement centralisés

Cette approche consiste en l'introduction d'un graphe de désignation unique qui couvre tout le système. Les noms dans ces systèmes ne comportent aucune indication (explicite ou logique) de la localisation de l'entité qu'ils désignent et, comme nous l'avons signalé au préalable, l'organisation du graphe de désignation n'est conditionnée que par les besoins administratifs. Ces systèmes sont assimilables à des systèmes logiquement centralisés qui cachent complètement la répartition.

La façon la plus simple d'obtenir un tel système consiste en l'introduction d'un site central qui joue le rôle de serveur de noms et de fichiers pour un ensemble de stations de travail interconnectées par un réseau local. Des systèmes ainsi configurés sont actuellement commercialisés. Bien que n'étant pas capables d'offrir tous les avantages de la répartition, on a constaté qu'ils sont performants et réalistes si leur charge est faible.

Le système Spice utilise une solution qui s'apparente à celle ci. Un noeud central performant et partiellement rendu fiable par duplication, joue le rôle de serveur de noms, de fichiers et d'archivage pour un ensemble de stations de travail. C'est le serveur Sesame [Thompson 85]. Son arbre de désignation peut être prolongé vers le bas, à n'importe quel niveau, par les arbres de désignation des serveurs des stations de travail.

Des implantations vraiment décentralisées de cette approche supposent un certain degré de duplication (automatique ou manuelle). A l'état actuel, son implantation vraiment répartie s'avère un défi.

Dans la catégorie de systèmes avec un SDS qui offre une vision logiquement unique, implantés de façon décentralisée, on trouve, par exemple:

- Locus ([Popek 81], [LOCUS 84]) qui offre un système de gestion fichiers réparti avec duplication partielle; les usagers ne voient qu'un seul arbre de fichiers et la désignation des usagers est introduite par des fichiers logiquement uniques.
- Grapevine [Birrell 82] et Clearinghouse [Oppen 81] qui offrent des bases de données réparties, spécialisées dans la désignation, permettant la mise à jour décentralisée, et implantées selon l'approche "système logiquement unique".
- Le système de gestion de fichiers ITC [Satyanaray 85]. Ce système offre un serveur central de fichiers et de noms logiquement unique (pour des raisons de performance et de résistance aux pannes ce serveur est effectivement réparti). Cet espace de noms logiquement unique apparaît dans les stations de travail comme étant "monté" à distance sur un catalogue dont le nom est prédéfini (/cmu dans la figure 1.12).

Tous ces systèmes offrent, naturellement, des noms globaux.

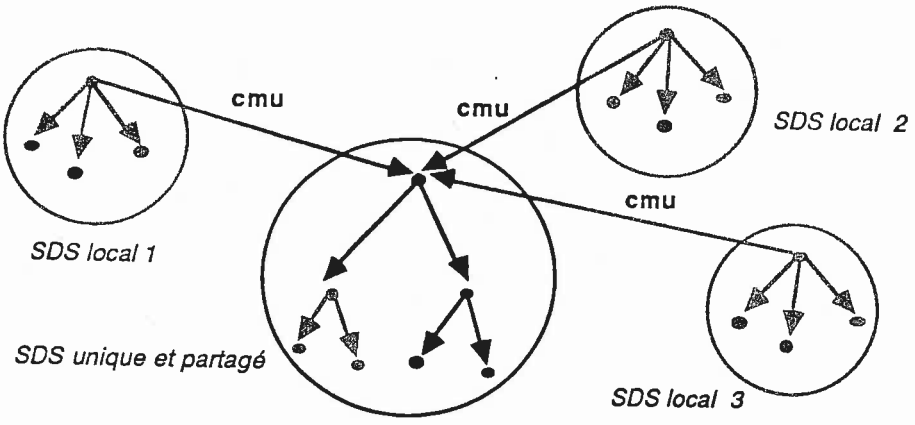


fig. 1.12 - SDS locaux et SDS partagé du système ITC

6.4. Conclusions sur les conventions de désignation

Des trois approches que nous venons de discuter pour construire un SDS réparti, les deux premières sont caractérisées par le fait d'offrir une vision non unique du système. La troisième suggère une vision unique du système mais son prix est un facteur qui doit être considéré avec soin en fonction de l'environnemnt où elle est utilisée.

En effet, le principal problème de l'intégration d'un SDS à un SER est celui de trouver les bons compromis entre modularité, uniformité, généralité d'application, extensibilité et fiabilité d'une part et, performance et facilité d'emploi d'autre part. Certains de ces objectifs sont contradictoires et en l'état actuel de nos connaissances, un bon compromis ne peut être obtenu qu'en face d'un cahier des charges bien précis.

Ceci nous amène à considérer le problème de l'implantation d'un SDS et de l'implantation de la fonction de traduction d'un nom.

7. Modèles de SDS répartis

Deux approches sont possibles pour introduire un SDS dans un SER. La première consiste à considérer le paradigme base de données répartie spécialisée dans la désignation, c'est l'approche serveur de noms. Cette approche se base sur les constatations suivantes:

- Un SDS est une base de données de descripteurs d'entités. On peut donc isoler cette fonction du système dans un service spécialisé pour promouvoir sa modularité et uniformité.
- Les quantités d'information en jeu et le fait que les attributs de beaucoup d'entités comme les usagers, les sites, les serveurs, etc. varient dans une échelle de temps assez large, rend possible la duplication de la base de données si l'on la spécialise dans la désignation de ce style d'entités. Ceci rend réaliste l'adoption par ces SDS des conventions de désignation des systèmes logiquement uniques.

L'autre approche relève d'un autre paradigme: la gestion des noms et des descripteurs des ressources doit être de la responsabilité des serveurs qui gèrent les ressources. Cette autre approche, que nous appellerons SDS explicitement réparti, part du point de vue que l'introduction d'un service

supplémentaire, le service de gestion de noms, est une complexité inutile.

Les SDS de deux systèmes illustrent particulièrement bien ces deux approches: le système Grapevine et le système V. Nous allons par la suite les présenter, analyser et comparer.

7.1. Illustration: le système Grapevine

Grapevine [Birrell 82] est un système réparti implanté sur un ensemble de réseaux Ethernet interconnectés (Xerox research internet). Il offre simultanément des fonctionnalités de messagerie électronique et un SDS qui permet la désignation d'entités ainsi que l'association d'attributs à ces entités. L'environnement de Grapevine va de la Californie jusqu'en Angleterre et comporte environ 1500 stations de travail et 3000 usagers. Nous nous intéressons ici essentiellement à son sous-système de désignation symbolique.

Ce sous-système prend l'approche de ne fournir que des noms globaux et non ambigus. L'espace des noms est structuré à deux niveaux. Logiquement, un catalogue racine contient des identificateurs de sous-catalogues ou partitions (*registries*). Chacune de ces partitions contient les identificateurs des entités enregistrées et désignées par Grapevine. A chacune de ces entités est associé un descripteur qui enregistre ses attributs.

Un nom Grapevine a la forme suivante: "E.P"

où "E" est l'identificateur de l'entité et "P" l'identificateur de la partition. Ce nom est toujours interprété comme étant le couple:

(Catalogue racine, "E.P") ou

(Catalogue racine, (Identificateur de la partition, Identificateur de l'entité))

Exemples: "jlm.rocquencourt", "printers.rocquencourt", "laser-bat11.rocquencourt", "hosts.rocquencourt", "mg.cnet", "vax2.rennes", "printers.public-servers", etc.

La notion de partition (sous-catalogue) ne correspond ni à celle de site ni à celle de serveur, elle est purement logique et peut être établie en fonction des besoins administratifs. Elle est conçue de telle sorte que la notion de localisation ou de serveur peut être complètement cachée. Une partition peut regrouper les entités d'une localisation géographique (rocquencourt, rennes, ...), d'une organisation présente dans plusieurs régions géographiques (cnet) ou d'un service (public-servers).

Attaché à chaque noeud terminal de l'arbre il y a un descripteur. Grapevine distingue deux types de descripteurs et donc deux types d'entités:

1/ Les individus (*individuals*). Leurs descripteurs enregistrent des listes de contrôle d'accès, des mots de passe, leurs adresses réseau et une liste de noms des serveurs qui gèrent leurs boîtes à lettres. Un noeud du type individu sert à désigner des usagers, des sites, des serveurs, etc.

2/ Les groupes. Leurs descripteurs enregistrent des listes de contrôle d'accès et une liste de noms, la liste de leurs membres (des individus ou des groupes). Un noeud du type groupe sert à établir une relation arbitraire entre un ensemble d'entités. Les groupes peuvent regrouper un ensemble de serveurs

qui rendent le même service (et donc servent à désigner le service), peuvent représenter une liste de diffusion de courrier et peuvent regrouper un ensemble d'usagers avec les mêmes droits (une liste de contrôle d'accès).

Descripteur d'un individu
listes de contrôle d'accès mot de passe adresse réseau liste de serveurs de messagerie

Descripteur d'un groupe
listes de contrôle d'accès liste de membres

fig. 1.13 - Descripteurs des entités dans Grapevine

Le SDS de Grapevine est une base de données répartie qui permet l'association d'attributs ou propriétés aux entités (entrées) qu'elle désigne. Elle offre des opérations (accessibles par le biais de l'échange de messages) qui permettent:

La reconnaissace de l'authenticité: [individu, mot de passe] → {vrai, faux}

Le test d'appartenance: [nom, groupe] → {appartient, n'appartient pas}

L'accès: [individu] → adresse réseau

L'énumération d'un groupe: [groupe] → liste des membres

La création et la destruction d'entrées et la modification de leurs attributs.

Cette base de données possède un caractère générique et se prête à beaucoup d'utilisations. Par exemple: le sous-système de messagerie† consulte le descripteur d'un individu pour obtenir la liste des serveurs de messagerie où il a des boîtes à lettres afin d'en choisir un; le client d'un serveur de fichiers consulte son descripteur pour obtenir son adresse réseau afin de l'utiliser; un serveur de fichiers se sert de Grapevine pour contrôler le mot de passe de ses clients et pour savoir s'ils peuvent l'accéder (en consultant une liste de contrôle d'accès qui est associée à son descripteur); un utilitaire d'impression consulte la liste des serveurs d'impression pour en choisir un; etc.

Le caractère global et unique de Grapevine fait qu'il se prête à fournir un référenciel unique de désignation. Cette caractéristique facilite l'introduction de schémas de protection bâtis sur des conventions communes. D'autre part, ce même caractère

† Rappelons que Grapevine est aussi un système de messagerie.

exige la duplication des catalogues et des descripteurs et des méthodes de résistance aux pannes.

L'implantation de Grapevine a introduit des innovations pour répondre à ces problèmes. Elle peut être ainsi caractérisée:

- Le service est dupliqué par plusieurs serveurs; le client peut s'adresser à n'importe quel serveur à-propos de n'importe quel nom; si le serveur contacté ne connaît pas le nom, le service redirige la requête vers un serveur qui le connaît.
- Le catalogue global est dupliqué et connu de tous les serveurs; chacune des partitions est dupliquée par N serveurs mais pas nécessairement par tous les serveurs.
- Le service Grapevine est lui-même décrit par une partition, la partition "gv"; dans cette partition, chaque individu représente un serveur Grapevine et chaque groupe représente une partition dans la mesure où il regroupe l'ensemble des serveurs qui connaissent cette partition; par exemple, le groupe "gv.gv" contient la liste de tous les serveurs Grapevine et désigne la partition "Grapevine".

7.2. Illustration: la désignation symbolique dans le V-system

Le V-System [Cheriton 84] est un SER implanté par un noyau, le V-Kernel, complété par des serveurs système. Ce système s'exécute sur un ensemble de stations de travail personnelles interconnectées par un réseau local. Il permet la communication entre processus par l'échange de messages. Les messages sont échangés directement de processus à processus (il n'y a pas de notion intermédiaire comme celle de porte). Les processus sont désignés de façon interne par des noms uniques et globaux (UID) directement visibles des clients.

Pour la désignation symbolique il prend l'option suivante [Cheriton 84a]: chaque serveur qui implante des entités permanentes, i. e., ayant besoin d'être désignées symboliquement, doit aussi implanter la désignation symbolique des entités qu'il gère. La désignation des entités est ainsi explicitement répartie par les serveurs qui les matérialisent. Chaque serveur possède son propre SDS; ces SDS sont indépendants.

Chaque entité permanente dans le système possède un nom interne global de la forme:

(UID du serveur, numéro de l'entité dans le serveur, type de l'entité)

Le numéro de l'entité dans le serveur est un numéro que le serveur s'éfforce de ne réutiliser que le plus tard possible. Le type de l'entité est optionnel. Les serveurs qui participent à la désignation symbolique, i. e., qui gèrent des entités désignées symboliquement, doivent aussi implanter des catalogues de désignation (au moins un s'il ne gèrent qu'une liste d'entités - un arbre dégénéré). Un catalogue possède aussi un nom interne.

Tous les noms symboliques se présentent comme un triplet:

(UID du serveur, numéro du catalogue, chemin d'accès)

Etant donné le caractère global et unique des noms internes des entités (et donc des catalogues), un serveur peut aussi associer un noeud de désignation à un nom interne de catalogue géré par un autre serveur. L'ensemble des graphes de désignation devient alors une forêt entrelacée arbitrairement.