

© [Helder Almeida] / [Fotolia]

# TELEINFORMATIQUE

## TOME 2 LOGICIELS DE COMMUNICATION

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

FORMATION

IF  
Création 1995  
Révision 2001

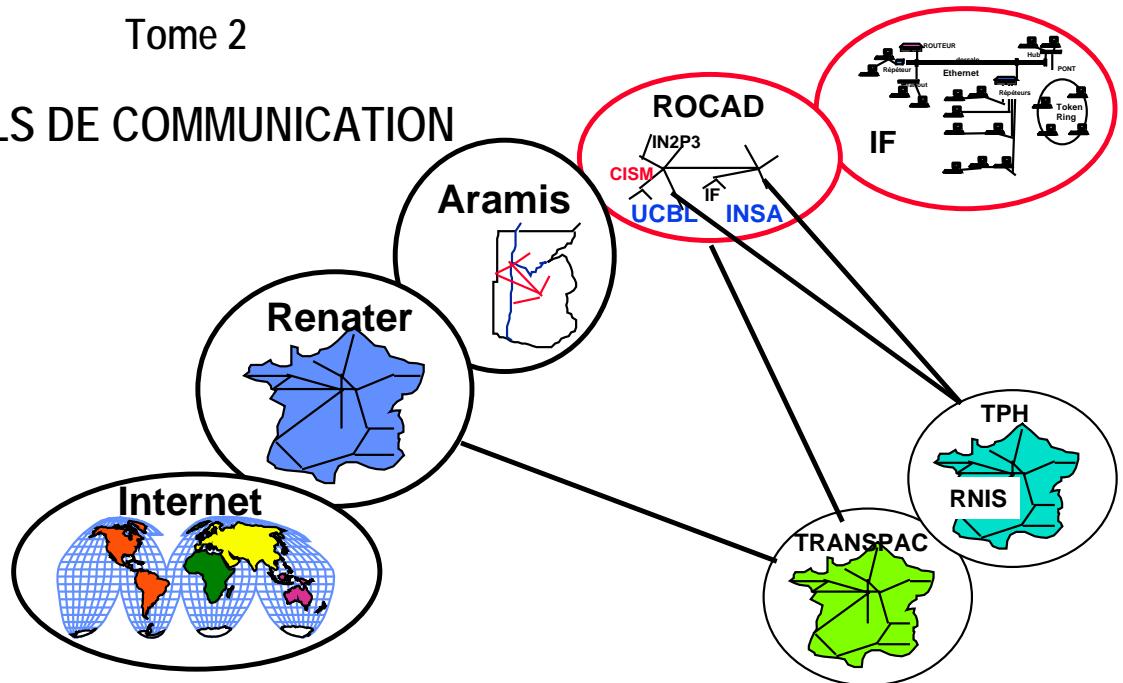
Auteur de la ressource pédagogique :  
**BEUCHOT Gérard**

## DEPARTEMENT INFORMATIQUE

## TELEINFORMATIQUE

Tome 2

LOGICIELS DE COMMUNICATION



1996-1997

G.Beuchot

Nota: Le chapître correspondant à la couche 2/OSI: Liaison de données (protocoles HDLC et BSC) est dans le tome 1: Bases Techniques pour les réseaux.  
La description des protocoles de l'architecture INET (TCP/IP) est aussi dans le tome 1.

## Sommaire

---

<b>1 COUCHE 3/OSI : SERVICE RESEAU AVEC CONNEXION</b>	<b>9</b>
1.1 Présentation	9
1.2 Service Réseau avec Connexion	9
1.2.1 Service fourni.....	9
1.2.2 Phases et primitives associées .....	12
1.2.3 Paramètres associés aux primitives.....	13
1.3 Protocole pour Réseau en Commutation de paquets : X 25	15
1.3.1 Présentation .....	15
1.3.2 Concepts de base : .....	16
1.3.3 Phases d'une connexion .....	18
1.3.4 Structures des NPDU : formats de paquets .....	24
1.3.5 Services complémentaires facultatifs d'usager.....	30
<b>2 ACCES A UN RESEAU EN COMMUTATION DE PAQUETS DEPUIS UN ETTD CARACTERE : PROTOCOLES X28 - X29 - X3</b>	<b>33</b>
2.1 Présentation	33
2.2 Exemple d'utilisation	34
2.3 Le service d'assemblage-désassemblage de paquets (Avis X3) .	35
2.3.1 Service .....	35
2.3.2 Paramètres de l'ADP .....	36
2.3.3 Profils .....	41
2.4 Interface pour accès à un service d'ADP (Avis X28) .	44
2.4.1 Signaux de commande et de service de l'ADP .....	44
2.4.2 Fonctionnement de l'interface . Diagramme d'état .....	48
2.5 Commande de l'ADP par l'ETTD distant (Avis X29) .	49
<b>3 COUCHE 3/OSI : RESEAU SANS CONNEXION</b>	<b>52</b>
3.1 Service fourni.	52
3.2 Service requis	53
3.3 Fonctions traitées par le protocole.	53
3.3.1 Composition / décomposition des IPDU.....	53
3.3.2 Contrôle de la durée de vie de la PDU .....	53
3.3.3 Routage et acheminement.....	53
3.3.4 Segmentation / réassemblage .....	54
3.3.5 Purge.....	54
3.3.6 Détection d'erreurs.....	54
3.4 Structure des PDU.	54
3.4.1 Partie fixe:.....	55
3.4.2 Partie adresse : .....	55
3.4.3 Partie segmentation : .....	55
<b>4 COUCHE 4/OSI : SERVICE TRANSPORT</b>	<b>56</b>
4.1 Introduction	56
4.1.1 Rôle: .....	56
4.1.2 Service requis: .....	56
4.1.3 Service fourni:.....	56
4.2 Services Transport	58

---

4.2.1	Normalisation.....	58
4.2.2	Fonctions.....	58
4.2.3	Modélisation .....	59
4.2.4	Paramètres des interactions .....	61
4.2.5	Négociation.....	61
4.3	Protocole de Transport	61
4.3.1	Introduction.....	61
4.3.2	Fonctions.....	64
4.3.3	Classes et fonctions.....	64
4.3.4	Adressage - Références .....	65
4.3.5	Unités de Données de Protocole ::TPDU.....	66
4.3.6	Automates .....	70
<b>5</b>	<b>COUCHE 5/OSI : SERVICE SESSION</b>	<b>76</b>
5.1	Présentation :	76
5.1.1	Rôle: .....	76
5.1.2	Service requis: .....	76
5.1.3	Services fournis: .....	76
5.1.4	Références .....	77
5.1.5	Définitions spécifiques: .....	77
5.2	Service Session	77
5.2.1	Découpage temporel d'une connexion de Session : .....	77
5.2.2	Activité .....	78
5.2.3	Resynchronisation .....	79
5.2.4	Service de jetons .....	79
5.2.5	Unités fonctionnelles et sous-ensembles .....	80
5.2.6	Qualité de service .....	83
5.2.7	Primitives .....	84
5.2.8	Synchronisation .....	85
5.3	Protocole de Session	87
5.3.1	Fonctions de la couche Session .....	87
5.3.2	SPDU affectées aux unités fonctionnelles.....	89
5.3.3	Structure des SPDU.....	90
5.3.4	Automate .....	93
<b>6</b>	<b>COUCHE 6/OSI : SERVICE PRESENTATION</b>	<b>95</b>
6.1	Rôle	95
6.2	Service requis.	95
6.3	Service fourni.	96
6.4	Eléments du service Présentation.	96
6.4.1	Contexte de présentation.....	96
6.4.2	Services élémentaires et primitives.....	97
6.4.3	Introduction à la compression de données.....	99
6.4.4	Introduction à la cryptographie.....	100
<b>7</b>	<b>COUCHE 6-7/OSI : SYNTAXE ABSTRAITE ASN.1</b>	<b>109</b>
7.1	Structure des PDU	109
7.1.1	Identificateur T : .....	110
7.1.2	Longueur .....	111

7.1.3	Contenu.....	112
7.2	Création des types	112
7.2.1	macro .....	112
7.2.2	module.....	112
7.3	Exemples :	113
7.3.1	Séquence, ensemble, choix, étiqueté, quelconque .....	113
7.3.2	"Message" .....	114
<b>8</b>	<b>COUCHE 7/OSI : APPLICATION</b>	<b>123</b>
8.1	Rôle et Services :	123
8.2	Fonctions des Services Application	125
8.3	Syntaxe abstraite et syntaxe de transfert	126
8.4	Structure de la couche Application	127
8.4.1	Concepts de base.....	127
8.4.2	Objet Association Simple :SAO.....	128
8.4.3	Structure d'une Entité d'Application.....	130
8.4.4	Noms et Fonctions répertoire.....	132
8.4.5	Définition d'une syntaxe abstraite .....	132
<b>9</b>	<b>COUCHE 7/OSI : SERVICES COMMUNS D'APPLICATION</b>	<b>133</b>
9.1	ACSE : Commande d'Association	133
9.1.1	Introduction : .....	133
9.1.2	Service fourni.....	133
9.1.3	Relation avec les autres ASE d'une entité d'Application.....	134
9.1.4	Service requis .....	134
9.1.5	Description succincte des services .....	134
9.1.6	Description succincte du protocole.....	135
9.2	RTSE : Service de Transfert Fiable	136
9.2.1	Introduction.....	136
9.2.2	Description du service.....	137
9.2.3	Services requis .....	138
9.2.4	Protocole RTSE .....	138
9.3	ROSE : Service d'opérations distantes	141
9.3.1	Introduction.....	141
9.3.2	Modèle pour les opérations distantes.....	144
9.3.3	Service fournis .....	145
9.3.4	Eléments de protocole ROSE .....	149
9.4	CCR : Commitment, Concurrency and Recovery	149
9.4.1	Concepts et processus .....	150
9.4.2	Service CCR .....	152
9.4.3	Protocole CCR.....	152
9.5	X500 : Service d'Annuaire	153
9.5.1	Composants de l'annuaire X500 .....	153
9.5.2	Description du service.....	154
9.5.3	Annuaire distribué.....	154
<b>10</b>	<b>MESSAGERIE X400 (COURRIER ELECTRONIQUE)</b>	<b>156</b>
10.1	Présentation	156
10.1.1	Introduction .....	156

10.1.2	Modèle fonctionnel CCITT.....	156
10.1.3	Classes d'agents utilisateurs.....	158
10.1.4	Normalisation.....	158
10.1.5	Adressage .....	158
10.1.6	Systèmes physiques .....	160
10.2	Services.	161
10.2.1	Décomposition.....	161
10.2.2	Opérations distantes : .....	162
10.2.3	Serveur de transfert fiable.....	162
10.3	Protocoles	166
10.3.1	Structures des APDU :.....	166
10.3.2	Service et Protocole (P1) Transfert de Messages : .....	167
10.3.3	Service (IPM) et Protocole (P2) Agent Utilisateur:.....	168
10.3.4	Autres services .....	169

---

## 11 COUCHE 7/OSI : TRANSFERT ET GESTION DE FICHIERS FTAM 180

11.1	Service fourni	180
11.1.1	Généralités .....	180
11.1.2	Fonctions associées au service fichier.....	182
11.2	Services requis	183
11.2.1	ACSE.....	183
11.2.2	Présentation .....	184
11.2.3	Session.....	184
11.3	Modèle de système de fichiers virtuel	184
11.3.1	Fichiers réels et fichiers virtuels : projection.....	184
11.3.2	Caractéristiques d'un fichier virtuel.....	185
11.3.3	Schéma d'un système de fichiers virtuels.....	185
11.3.4	Structure d'accès .....	186
11.3.5	Structures de présentation .....	187
11.3.6	Types de documents.....	188
11.4	Service de transfert de fichiers	188
11.4.1	Phases et Régimes.....	189
11.4.2	Eléments de service.....	191
11.5	exemples	193
11.5.1	Transmission d'un fichier complet à un système distant.....	193
11.5.2	Accès à une base de données.....	194
11.5.3	Serveur de fichiers sur un réseau local .....	195
11.5.4	Administration de fichier.....	196
11.6	Annexe1 : Objets définis en syntaxe ASN.1	197
11.7	Annexe 2 : Modèle de système de fichiers	197
11.7.1	Concepts de base.....	197
11.7.2	Structure d'accès .....	198
11.7.3	Description formelle .....	199
11.7.4	Structure de transfert .....	200
11.7.5	Actions sur un fichier.....	201

---

## 12 COUCHE 7/OSI : MESSAGERIE INDUSTRIELLE MMS 202

12.1	Service requis	202
12.2	Service fourni	202

12.3	Sous-ensembles fonctionnels	203
12.3.1	Modalités .....	203
12.3.2	Gestion de contexte .....	203
12.3.3	Gestion de l'Equipement Virtuel de Production .....	204
12.3.4	Gestion de variables .....	207
12.3.5	Gestion de programmes .....	208
12.3.6	Gestion d'événements .....	208
12.3.7	Gestion de sémaphores .....	209
12.3.8	Gestion de domaines .....	209
12.3.9	Gestion de journal.....	210
12.3.10	Communication opérateur.....	210
12.3.11	Annexe : Gestion de fichiers .....	210
12.4	Eléments de protocole.	211
<b>13</b>	<b>COUCHE 7/OSI : APPEL DE PROCEDURES DISTANTES (RPC)</b>	<b>212</b>
13.1	Le modèle du standard RPC	212
13.1.1	Le concept de RPC ouvert.....	212
13.1.2	Principe d'une application distribuée utilisant le RPC .....	213
13.1.3	Architecture du modèle .....	214
13.1.4	Le service RPC OSI.....	215
13.1.5	Service d'opération distante (ROSE) nécessaire.....	217
13.2	Architectures réelles pour un RPC et exemples d'utilisation	217
13.2.1	Structure modulaire.....	217
13.2.2	Exemple d'utilisation .....	218
13.2.3	Implantation sous Unix sur un service de niveau 4/OSI .....	220
<b>14</b>	<b>COUCHE 7/OSI : PROTOCOLE D'APPAREIL VIRTUEL</b>	<b>222</b>
14.1	Applications interactives et Applications de transfert d'information pour restitutio...	222
14.2	Modèle conceptuel	223
14.2.1	Définitions.....	223
14.2.2	Structure logique et implantation.....	224
14.3	Exemple ancien : Appareil virtuel Architel	225
14.3.1	Structure de document.....	225
14.3.2	Signalisation .....	226
14.3.3	Objets et modèles.....	226
14.3.4	Programme de saisie .....	227
<b>15</b>	<b>COUCHE 7/OSI : SYSTEMES TRANSACTIONNELS REPARTIS OSI-TP</b>	<b>229</b>
15.1	Introduction : le traitement transactionnel	229
15.1.1	Caractéristiques des applications transactionnelles.....	230
15.1.2	La transaction .....	230
15.1.3	Déroulement d'une transaction .....	232
15.1.4	Validation à deux phases.....	234
15.2	Modèles d'échanges entre applications	235
15.2.1	Modèles client-serveur .....	235
2.1.3	Le modèle DCE (Distributed Computing Environment) de l'OSF .....	237
15.2.2	Le modèle Distributed Transaction Processing (DTP) de l'X/OPEN ....	240
15.2.3	Echanges inter-applications par files de messages : modèle MQ.....	243

15.3	La normalisation du traitement transaction réparti : OSITP	245
15.3.1	Présentation de la norme TP (Transaction Processing) .....	245
15.3.2	Terminologie et concepts de base .....	246
15.3.3	Dialogue .....	247
15.3.4	Arbre de transaction.....	247
15.3.5	Services et protocole d'OSITP .....	248
15.4	Les moniteurs transactionnels	250
15.4.1	TUXEDO.....	251
15.4.2	ENCINA.....	254
15.4.3	CICS/6000 .....	256
15.4.4	Moniteur MQM pour mise en œuvre du modèle MQ .....	257
15.5	Conclusion	261
15.6	Annexes :	262
15.6.1	Terminomogie.....	262
15.6.2	Bibliographie .....	263
15.6.3	L'environnement distribué .....	264

---

# 1 COUCHE 3/OSI : SERVICE RESEAU AVEC CONNEXION

## 1.1 Présentation

La couche 3 du Modèle de Référence de l'OSI, couche Réseau, peut fournir deux types de service :

- Un service Réseau avec connexion

utilisé sur les réseaux étendus décrit dans les normes OSI 8348 CCITT X213.

Pour un réseau en commutation de paquets, il est supporté, en particulier, par le protocole  
CCITT X25 = OSI 8208

Pour les réseaux en commutation de circuits on utilise X21.

- Un service Réseau sans connexion utilisé sur les réseaux locaux décrit dans les normes OSI 8473 (ECMA 92)

Les objectifs poursuivis par ces services sont en grande partie semblables. Toutefois l'établissement d'une connexion permet d'assurer une qualité de service donnée, éventuellement négociée, et le contrôle des accès.

Les fonctions d'**administration de réseau** (voir "Administration de réseau" : administration de couche et opérations de couche) font aussi partie de cette couche. Elle sont définies dans la norme OSI 9542

## 1.2 Service Réseau avec Connexion

Le service décrit ci-dessous est celui défini par l'avis CCITT X213

### 1.2.1 Service fourni

#### 1.2.1.1 Caractéristiques générales

"Le service réseau assure le **transfert transparent de données** entre utilisateurs du service de réseau. Il leur **rend invisible** la façon dont les ressources de communication mises en oeuvre sont utilisées pour réaliser ce transfert."

Il assure en particulier :

- L'indépendance par rapport aux supports de transmission sous-jacents (utilisation de sous-réseaux hétérogènes) en déchargeant l'utilisateur de toute préoccupation autre que la qualité de service.
- Le transfert de bout en bout entre utilisateurs du service de réseau en traitant toutes les fonction de **routage**.
- la transparence des informations transférées ; les données utilisateurs sont une suite d'octets.
- Le choix de la qualité de service soit par les paramètres négociés soit par des paramètres préétablis.
- **L'adressage de l'utilisateur** du service de réseau en l'identifiant de manière non ambiguë.

Pour qu'un utilisateur du service de réseau puisse faire la distinction entre plusieurs connexions de réseau reliées à un même point d'accès au service (NSAP), il doit disposer d'un mécanisme **local** d'identification d'extrémité de connexion (NCEP) qui est utilisé par toutes les primitives d'accès au service. Cette identification locale **ne doit pas être confondue** avec les paramètres d'**adresse** utilisés pour la **Connexion de réseau**.

### 1.2.1.2 Fonctions fournies

Le service de réseau offre les fonctions suivantes :

1) Connexion de réseau.

Plusieurs connexions peuvent exister entre un même couple d'utilisateurs.

2) Négociation d'une certaine qualité de service entre deux utilisateurs et le fournisseur du service de réseau.

3) Transfert de données transparent (NSDU). Ces NSDU sont constituées d'un nombre entier d'octets.

4) Contrôle de flux entre utilisateurs

5) Réinitialisation pour remettre la connexion dans un état défini et synchroniser les activités des deux utilisateurs.

6) Libération inconditionnelle et éventuellement destructive d'une connexion par un utilisateur ou le fournisseur.

Le service de réseau peut aussi fournir de manière optionnelle les deux fonctions suivantes :

7) Transfert de données express de taille limitée avec un contrôle de flux indépendant de celui des données normales.

8) Confirmation de la réception des données.

Le service de réseau n'offre qu'une seule classe de service (avec deux fonctions optionnelles).

### 1.2.1.3 Modèle du service de réseau

Chaque connexion du service de réseau peut être représentée par deux files d'attente, de A vers B et B vers A, reliant les NSAP A et B.

Chaque file est de capacité limitée, mais cette limite n'est pas nécessairement fixée ou déterminable.

Les files sont vides avant qu'un utilisateur y ai placé un objet de connexion. Elles le deviennent après introduction d'un objet de libération ou de réinitialisation dans l'une d'elle.

Les objets sont placés dans les files par les utilisateurs du service sous contrôle du fournisseur ou par le fournisseur lui-même. Ils sont retirés sous le contrôle de l'utilisateur destinataire.

Dans une file il est possible de :

- modifier l'ordre de deux objets si le second est défini comme prioritaire (données express par exemple)
- supprimer un objet si l'objet suivant est destructif (objet de libération par exemple)

### 1.2.1.4 Qualité du service réseau

La qualité du service de réseau (QOS) se rapporte à des caractéristiques observées entre ses extrémités. Elle relève du fournisseur du service. Quand une connexion est établie, les utilisateurs du service de réseau doivent avoir la même connaissance et interprétation de la qualité de service.

La QOS est décrite par des paramètres

- négociés en phase de connexion
- prédéfinis par une autre méthode

Après connexion ces paramètres ne peuvent être renégociés.

D'autre part le fournisseur ne peut garantir que ces paramètres soient maintenus.

### Paramètres de performances

Rapidité :

- Délai d'établissement de connexion de réseau
- Débit
- Temps de transit
- Délai de libération de connexion de réseau

Les paramètres "débit" et "temps de transit" peuvent être négociés. Ces valeurs peuvent être différentes pour chaque sens de transmission.

Le temps de transit est le temps écoulé entre une demande de transfert de données et l'indication de transfert de données correspondante lorsque ce transfert est correct.

Le débit est défini pour une suite de  $n$  NSDU ( $n \geq 2$ ) transmis à la cadence maximale. C'est la plus petite des deux valeurs données par le nombre d'octets des  $n - 1$  dernières NSDU divisé par le temps écoulé entre : soit première et dernière "demande de transfert de données", soit première et dernière indication de transfert de données" (pour un transfert correct).

Exactitude, fiabilité

Probabilité d'échec d'établissement de connexion de réseau

Taux d'erreur résiduel

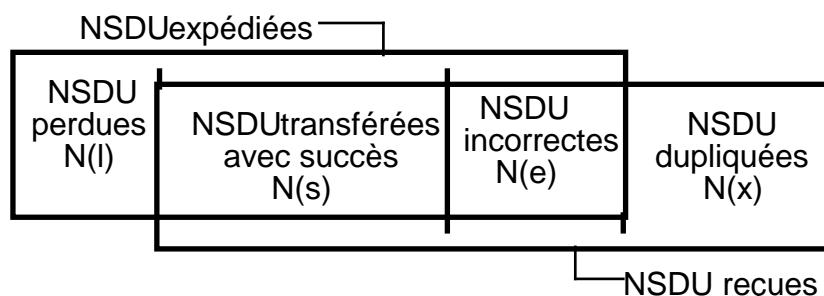
Probabilité de rupture de connexion de réseau

Probabilité d'incident de transfert

Probabilité d'échec de libération de connexion de réseau

Le taux d'erreur résiduel (RER) est le rapport du nombre total de NSDU incorrectes, perdues ou dupliquées au nombre total de NSDU transférées :

$$RER = \frac{N(l) + N(e) + N(x)}{N}$$



### Autres paramètres

Protection de la connexion de réseau

Priorité de la connexion de réseau

Coût maximal acceptable

## 1.2.2 Phases et primitives associées

### 1.2.2.1 Etablissement de connexion de réseau

Cette phase sert à établir une connexion de réseau entre deux utilisateurs s'ils existent et sont connus du fournisseurs.

Les demandes d'établissement simultanées peuvent conduire à une ou deux connexions de réseau ou aucune.

Les primitives suivantes sont utilisées :

Demande de connexion de réseau  
Indication               "  
Réponse à une demande de connexion de réseau  
Confirmation            "

### 1.2.2.2 Transfert de données

Cette phase traite le transfert de données normales et optionnellement express dans les deux sens à l'alternat ou simultanément.

Durant cette phase il peut y avoir réinitialisation de la connexion.

Les primitives suivantes sont utilisées :

Demande de réinitialisation de réseau  
Indication               "  
Réponse à une Demande de réinitialisation de réseau  
Confirmation            "

Demande de transfert de données de réseau  
Indication               "

en option :

Demande de transfert de données express de réseau  
Indication               "  
Demande d'accusé de réception de données de réseau  
Indication               "

### 1.2.2.3 Libération de connexion

La libération est effectuée à la demande d'un utilisateur ou du fournisseur du service de réseau.

Elle peut intervenir à tout moment. Elle ne peut être rejetée.

Dès que la procédure de libération est engagée, la connexion de réseau doit être libérée.  
Ce service n'est pas confirmé.

Les primitives suivantes sont utilisées :

Demande de déconnexion de réseau  
Indication               "

### 1.2.3 Paramètres associés aux primitives

Nous étudierons très succinctement les paramètres associés aux primitives d'utilisation de service de réseau.

### 1.2.3.1 Description de quelques paramètres

Adresse :

Les paramètres dont les valeurs sont des adresses se réfèrent tous à des adresses de NSAP (Point d'accès au service de réseau). La représentation de ces adresses est de longueur variable (d'un maximum de 32 chiffres décimaux). Les adresses de NSAP sont fondées sur le concept de domaines d'adressage hiérarchiques.

Chaque domaine peut être divisé en sous-domaines.

Pour les (sous-)réseaux publics de données l'avis X121 du CCITT fournit un plan de numérotage international. Le numéro international est composé de 5 à 14 chiffres décimaux. Il est parfois nécessaire de lui ajouter un suffixe permettant de désigner un utilisateur raccordé à un sous-réseau privé. Trois chiffres décimaux (ou plus) peuvent être prévus à cette fin.

Les paramètres d'adresse utilisés sont :

Adresse de l'entité appelée

Adresse de l'entité appelante

Adresse en réponse

L'adresse en réponse donne l'adresse du NSAP effectivement connecté. Elle peut être différente de celle de l'entité appelée en cas de réacheminement de l'appel ou d'adressage générique.

Options :

Elles permettent d'utiliser les fonctions "confirmation de réception" et "données express".

Paramètres de QOS :

Ils permettent de négocier le débit et le temps de transit.

L'appelant spécifie une valeur cible et une valeur minimale acceptable. Le fournisseur remplace la valeur cible par une valeur disponible dans l'intervalle [valeur cible - valeur minimale]. L'utilisateur appelé choisit la valeur adopté dans l'intervalle [valeur disponible - valeur minimale].

Paramètres de déconnexion ou réinitialisation :

Origine :

désigne l'origine de la fonction : utilisateur, fournisseur ou "inconnue"

Raison :

Si l'origine est inconnue, ce paramètre est non défini sinon il indique la cause (permanente ou transitoire) du refus de connexion ou de la libération.

Pour une réinitialisation par le fournisseur la cause peut être "non spécifiée" ou encombrement. Si elle est demandée par un utilisateur la raison est simplement "demandée par utilisateur".

### 1.2.3.2 Utilisation des paramètres

#### Primitives de connexion :

Les adresses d'entité appelée et appelante sont utilisées dans les demandes et indications. Les adresses en réponse dans les réponses et confirmations.

Les paramètres d'option ou de QOS peuvent se trouver dans toutes les primitives.

On peut aussi, dans certains cas, placer des données utilisateur dans toutes les primitives.

#### Primitives de transfert de données :

Pour les données normales, outre les données utilisateur, elles peuvent contenir une "demande de confirmation" si cette option est acceptée.

Les primitives de données express ne comporte que des données utilisateur.

#### Primitives de libération ou de réinitialisation :

Les indications de réinitialisation ou de déconnexion peuvent comporter un paramètre "origine" qui précise quel est l'initiateur de cette fonction : utilisateur ou fournisseur.

Les requêtes et les indications comportent un paramètre "raison".

Les demandes et indications de déconnexion peuvent aussi porter des données utilisateurs ou une adresse en retour.

## 1.3 Protocole pour Réseau en Commutation de paquets : X 25

### 1.3.1 Présentation

#### 1.3.1.1 Introduction

Le protocole X25.3 est le sous-ensemble de niveau 3/OSI de la Recommandation X25 du CCITT. Cette recommandation spécifie l'interface entre E.T.T.D. fonctionnant en mode paquet et raccordés à un réseau public de transmission de données par liaisons spécialisées.

Pour un raccordement par le réseau téléphonique il est complété par la Recommandation X32.

La Recommandation X25 couvre les couches 1 à 3 du modèle de Référence de l'OSI.

Au niveau physique il utilise un des protocoles X21 ou X21bis.

Au niveau liaison de données, il prévoit un choix entre plusieurs protocoles : LAP , LAPB, PML (multiliaison). L'utilisation de LAPB, sous-ensemble de HDLC, est recommandée. Les deux autres sont du même type : LAP est une version ancienne non conforme à HDLC et la procédure multiliaison permet l'éclatement sur plusieurs liaisons physiques.

Le sous-ensemble X25.3 a été repris par l'OSI pour le standard 8208.

### **1.3.1.2 Service fourni**

Le service fourni est décrit dans le chapitre " Service de réseau avec connexion".

Le protocole X25.3 permet de transporter des séquences complètes de paquets de données. Ces paquets sont de taille bornée et éventuellement négociable (option). Une séquence complète est une suite de paquets chaînés qui permet de transmettre une NSDU sous une forme fractionnée.

Cette transmission est faite sur un chemin du réseau de télécommunication appelé "Circuit virtuel" ( Voir ci-dessous).

### **1.3.1.3 Service requis**

Le protocole X25.3 utilise un service de Liaison de données avec connexion fourni par le sous-ensemble X25.2.

Toutefois il peut être utilisé avec d'autres services de niveau 2 notamment celui fourni par un protocole de type BSC ou certains protocoles utilisés au niveau 2 sur les réseaux locaux.

## **1.3.2 Concepts de base :**

### **1.3.2.1 Adressage**

Un utilisateur du service de réseau doit connaître l'adresse du NSAP qui lui permet d'accéder au service et l'adresse du NCEP qui désigne l'extrémité du point de la connexion qu'il utilise. En pratique un NCEP correspond à une voie logique. La désignation des NSAP est interne à l' E.T.T.D. et du ressort de son concepteur.

Lorsqu'il utilise un circuit virtuel commuté (CVC), il doit aussi désigner l' E.T.T.D. distant avec lequel il veut communiquer par son adresse absolue, unique sur le réseau.

Cette adresse est mise en correspondance avec l'adresse de NCEP ou numéro de voie logique.

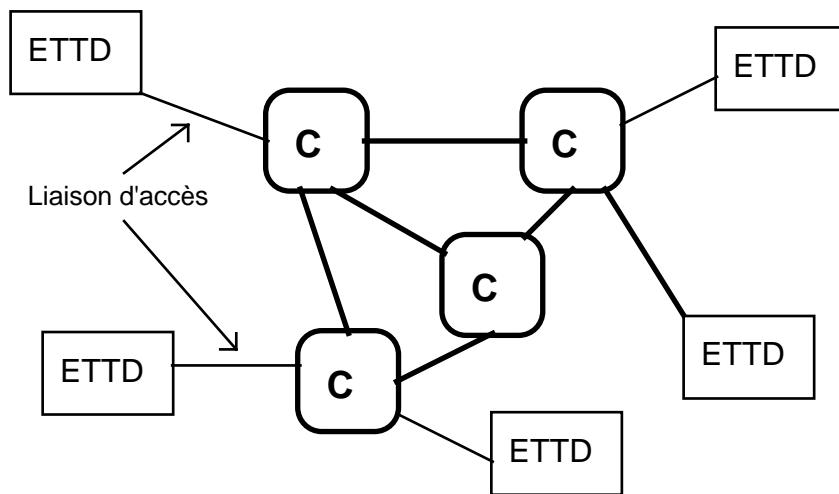
Ainsi, selon l'état de la connexion ou la fonction traitée l'adresse de l'utilisateur distant prend des formes variées.

Dans le protocole X25.3, l'adresse absolue d'un E.T.T.D. est une adresse hiérarchique codée sur 15 chiffres décimaux (12 + 3) au maximum.

Des mécanismes optionnels permettent d'y adjoindre une sous-adresse (voir ci-dessous). D'autre part, une liaison de données peut supporter au maximum 4096 voies logiques (Leur numéro est codé sur 12 bits). Le nombre réel N de voies logiques utilisables est défini à l'abonnement pour un réseau public ou à priori pour un réseau privé.

### 1.3.2.2 Voie logique, Circuit virtuel.

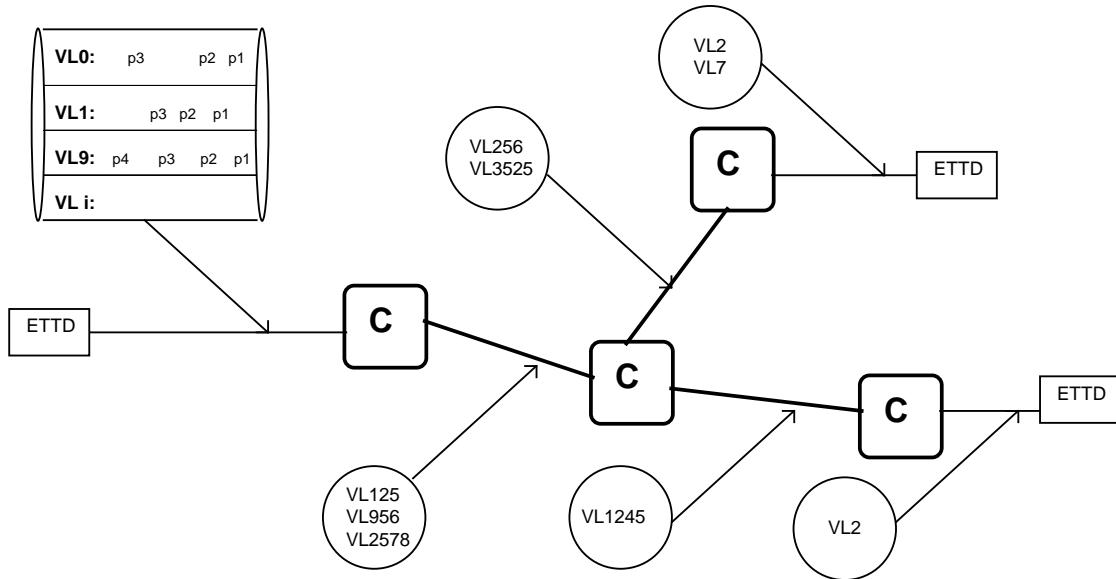
Matériellement, un réseau est constitué de commutateurs de paquets ou d' E.T.T.D. utilisateurs interconnectés par des **liaisons physiques**. Un utilisateur est ainsi connecté au réseau (à un de ses commutateurs) par une liaison d'accès.



Sur chacune de ces liaisons physiques, on établit une **liaison de données** (niveau 2/OSI, X25.2 = HDLC version LAPB) qui permet d'échanger des trames d'information entre ces équipements.

Le service de réseau permet de **multiplexer** plusieurs connexions de réseau sur une liaison de données. Dans un E.T.T.D. ou un commutateur chacune de ces connexions de réseau est appelée **voie logique**. Une liaison de données transporte à la demande, successivement, les PDU correspondant à ces voies logiques. Pour atteindre un E.T.T.D. distant à travers le réseau, une NPDU (appelée aussi "paquet") utilise successivement plusieurs liaisons interconnectées par un ou plusieurs commutateurs.

Dans les réseaux à **circuits virtuels**, tous les paquets transmis lors d'une connexion de réseau suivent le **même chemin** appelé **circuit virtuel**.



Circuits virtuels :      VL0 - VL956 - VL3525 - VL7  
                           VL9 - VL125 - VL256 - VL2  
                           VL1 - VL2578 - VL1245 - VL2

Un circuit virtuel est un assemblage en cascade de **voies logiques**. Ces voies logiques n'ont qu'une existence locale (entre deux noeuds adjacents) et le plus souvent temporaire. En effet, un circuit virtuel peut être établi à la demande ( Circuit virtuel commuté : CVC ) ou une fois pour toute ( Circuit virtuel permanent : CVP ).

L'établissement d'un circuit virtuel revient à définir le chemin qu'emprunteront les paquets et à réserver les ressources (zones de mémoire de stockage, voies logiques) nécessaires à leur acheminement.

### 1.3.3 Phases d'une connexion

#### 1.3.3.1 Etablissement de connexion

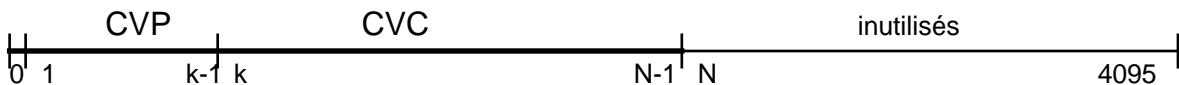
##### 1.3.3.1.1 Utilisation des voies logiques

Lorsqu'un E.T.T.D. initiateur établit une connexion, il désigne le numéro de voie logique qu'il va utiliser. Lorsqu'un E.T.T.D. est accepteur de communication, la voie logique qui le relie au réseau est établie par un commutateur de paquets. Il est souhaitable de minimiser les conflits résultants de l'utilisation simultanée d'un même numéro de voie logique par un E.T.T.D. et un commutateur (pour des connexions distinctes).

Pour cela on utilise les règles suivantes :

- Si l'on utilise N circuits virtuels dont K permanents ( $K \geq 0$ ), les numéros 0 à  $K-1$  sont réservés aux circuits virtuels permanents (préétablis). Les numéros  $K$  à  $N-1$  sont

utilisés à la demande. Les numéros N à 4095 sont inutilisés. Il est recommandé de réserver le numéro 0 aux reprises sur la liaison d'accès (voir ci-dessous).



- Le **réseau** établit toujours ses connexions en utilisant le **plus petit numéro de voie logique disponible**.

L'E.T.T.D. initiateur devra donc toujours utiliser le **plus grand numéro de voie logique disponible**.

Ainsi une contention ne pourra se produire que s'il reste une seule voie logique disponible et qu'il se produit des appels simultanés.

#### 1.3.3.1.2 Etablissement d'une connexion sur un CVC



Remarque : Les mécanismes décrits ci-dessous sont ceux utilisés sur les réseaux publics (par exemple Transpac). Les structures des PDU correspondantes sont données ci-dessous au paragraphe 4.

#### 1.3.3.1.3 Réseau simple

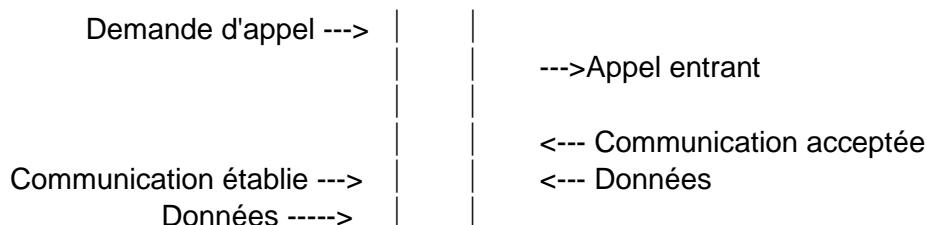
L'E.T.T.D. appelant envoie sur la voie logique I (numéro maximal disponible) un **paquet de demande d'appel** qui contient l'adresse absolue de l'E.T.T.D. distant appelé.

	Appelant	Réseau	Appelé
Longueurs	0 9 1 6 9 0 0 1 0 8 6 0	9 9 1 3 8 2 0 5 0 6 4 0 1 6 9 0 0 1 0 8 6 0	9 0 1 3 8 2 0 5 0 6 4 0

Le commutateur CE, à l'entrée du réseau, contrôle les droits d'accès, affecte les ressources nécessaires à cette voie logique I, ajoute l'adresse absolue de l'E.T.T.D. appelant sur cette voie logique, puis achemine ce paquet d'appel sur la voie logique K d'un chemin disponible dans le réseau. Ce paquet d'appel progresse ainsi dans le réseau jusqu'au commutateur CS auquel est relié l'E.T.T.D. appelé.

Ce commutateur CS établit la liaison avec l'E.T.T.D. appelé sur la voie logique J (plus petit numéro disponible) et lui transmet un **paquet d'appel entrant** qui ne contient, dans son champ d'adresse, que l'adresse absolue de l'E.T.T.D. appelant.

Après avoir analysé l'appel (puis transmis une indication à sa couche supérieure et reçu d'elle une réponse...) l'E.T.T.D. appelé envoie sur la voie logique J un **paquet d'acceptation d'appel** (ou un paquet de libération s'il refuse la connexion).



Ce paquet suit en sens inverse le chemin établi à l'appel (mêmes voies logiques). Il est transmis à l'E.T.T.D. appelant sur la voie logique I comme un **paquet de confirmation d'appel** (ou un paquet de libération si la connexion a été refusée par l'E.T.T.D. appelé ou le réseau).

En cas d'acceptation, le CVC est alors complètement établi.

Si le réseau refuse (droits d'accès, ressources insuffisantes) d'établir la connexion, il envoie une libération au demandeur.

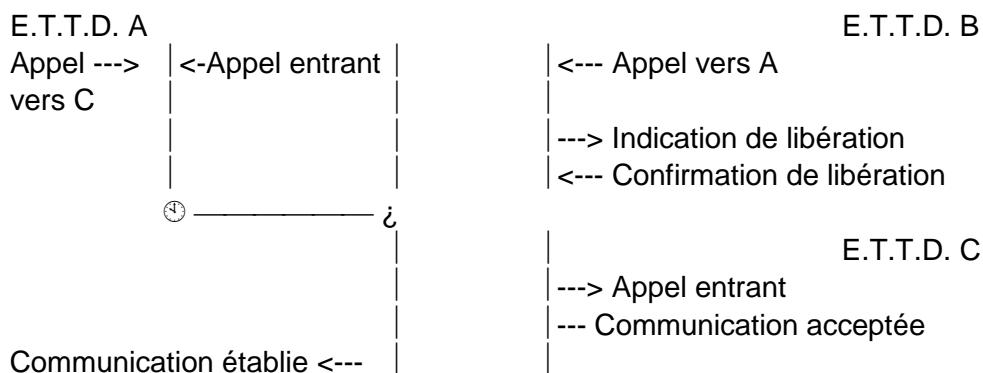
#### 1.3.3.1.4 Utilisation de sous-réseaux locaux

Les systèmes dans les sous-réseaux locaux utilisent l'extension d'adresse X121 : trois derniers chiffres de l'adresse. Le sous-réseau doit indiquer la sous-adresse locale de l'appelant et la sous-adresse de l'appelé sur le sous-réseau distant.

	Appelant	Réseau	Appelé
Longueurs	2 A	B A	B 0
1 2		1 3	
1 6		8 2	
9 0		0 5	
0 1		0 6	
0 8		4 1	
6 8		2 0	
		1 6	
		9 0	
		0 1	
		0 8	
		6 8	

#### Collision de connexion

Lorsqu'une seule voie logique reste disponible à l'interface, une collision de connexions peut se produire si un E.T.T.D. émet une demande d'appel alors que l'E.T.C.D. (réseau) lui transmet une indication d'appel de E.T.T.D. distant. Dans ce cas, l'E.T.C.D. (réseau) **envoie une indication de libération à E.T.T.D. distant et accepte la demande d'appel de E.T.T.D. local**. Celui-ci voit donc son appel acheminé normalement.



### 1.3.3.2 Transfert de données

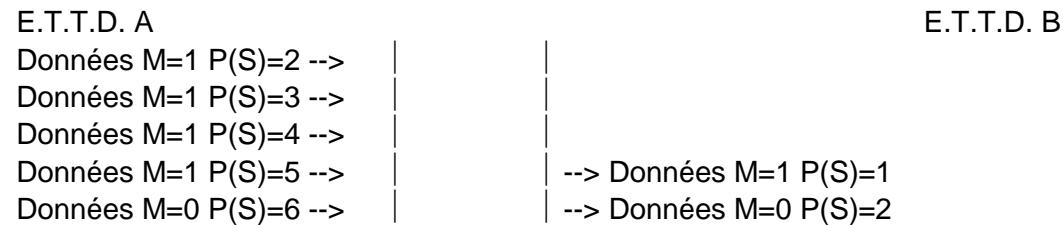
#### 1.3.3.2.1 Données normales

Les données normales de l'utilisateur sont placées dans des paquets de données de taille bornée (par défaut 128 octets ou optionnellement 16,32,64,256,512,1024 ou 2048 octets sur certains réseaux) qui sont séquencés par un numéro de paquet P codé sur 3 bits (modulo 8) ou sur 7 bits (modulo 128) si on utilise la numérotation étendue.

Le protocole X25 permet à l'utilisateur de différencier deux types de données grâce à un indicateur appelé "élément binaire qualificatif" ou "bit Q". Ce mécanisme est facultatif et les deux types de données ont exactement les mêmes propriétés. De plus le séquencement des paquets ne tient pas compte de cet indicateur.

Le protocole X25 réalise le transfert de **séquences complètes de paquets de données**. Ces séquences sont constituées de paquets chaînés ayant tous le même type (qualifié ou non). Ce chaînage utilise un indicateur binaire M : "données à suivre" pour indiquer que le paquet concerné aura un successeur dans la séquence. Le dernier paquet d'une séquence à son indicateur M mis à 0. Il peut être de taille quelconque (non nulle). Les paquets antérieurs ont leur indicateur M à 1. Ils doivent tous être de taille maximale.

Dans l'exemple ci-dessous E.T.T.D. A émet un message de 150 octets dans 4 paquets plein de 32 octets et un paquet final de 22 octets. Le réseau délivre cette séquence sous forme d'un paquet plein de 128 octets et d'un paquet final de 22 octets.



Un indicateur "confirmation de remise" D, optionnel, permet à E.T.T.D. source d'indiquer qu'il souhaite recevoir un accusé de réception de remise de bout en bout. Cet accusé de remise est passé au moyen du numéro de séquence de paquet en réception P(R) (qui indique le numéro du prochain paquet attendu). L'utilisation de cet indicateur ne dispense pas de l'utilisation d'un protocole de niveau supérieur pour effectuer les récupérations après réinitialisation ou libération.

Si cet indicateur D n'est pas utilisé (cas le plus fréquent) la gestion du séquencement est réalisée localement au niveau de la voie logique. S'il est utilisé, le séquencement est respecté de bout en bout.

Ceci conduit à ne jamais émettre de paquet de taille nulle (supprimé par le réseau) et à ne pas utiliser des tailles de paquet différentes sur les voies logiques aux extrémités du circuit virtuel.

### 1.3.3.2.2 Données express : paquets d'interruption

Lorsqu'une connexion de réseau est établie, un flux de données express, indépendant du flux de données normales et du contrôle de flux correspondant peut être utilisé. Ces données sont passées par des **paquets d'interruption** qui doivent être **confirmés**. La taille des données utilisateur est réduite à un **seul octet**. Une source ne peut émettre un nouveau paquet d'interruption tant qu'elle n'a pas reçu le paquet de confirmation du paquet d'interruption qu'elle vient d'émettre. Les paquets d'interruption ne sont pas soumis par le réseau au contrôle de flux et peuvent éventuellement doubler des paquets de données normales.



### 1.3.3.3 Déconnexion. Réinitialisation. Reprise sur la liaison d'accès.

#### 1.3.3.3.1 Déconnexion

A un instant quelconque, une connexion peut être libérée soit par un des E.T.T.D. communiquant soit par le réseau.

Pour cela, un E.T.T.D. émet un paquet de **demande de libération** qui sera transmis à E.T.T.D. distant par une **indication de libération**. Dès que le réseau a libéré les composants réservés par la voie logique concernée (circuit virtuel) il répond par une **confirmation de libération**. Le paquet de demande de libération, en progressant dans le réseau libère ainsi tous les composants. Il **purge** tous les paquets de données (ou autres) qu'il rattrape ou qu'il **croise**. E.T.T.D. distant accueille réception de l'indication de libération par un paquet de **confirmation de libération**.



Le réseau peut libérer une connexion (sur incident) en transmettant aux E.T.T.D. reliées une indication de libération.

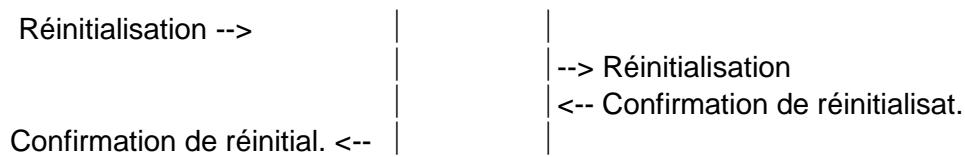
En cas de collision : transmission simultanée à une interface d'une demande et d'une indication de libération sur la même voie logique , on considère que celle-ci est libérée et aucune confirmation n'est émise.

#### 1.3.3.3.2 Réinitialisation

Lorsqu'une connexion est établie, un circuit virtuel (toutes les voies logiques qui le constituent) peut être réinitialisé par un paquet de **demande de réinitialisation** émis par un

des E.T.T.D.. Il est transmis par le réseau à l'autre E.T.T.D. sous la forme d'un paquet **d'indication de réinitialisation**.

Cette indication doit être confirmée par un paquet de **confirmation de réinitialisation**. Cette réinitialisation est destructive et détruit tous les paquets de données rattrapés ou croisés.



Cette réinitialisation place la connexion dans un état correspondant à celui qui suit l'établissement de connexion.

Une collision de réinitialisation se produit à une interface lorsqu'une demande et une indication de réinitialisation sont émises simultanément. Dans ce cas le circuit virtuel est considéré comme réinitialisé et aucune confirmation de réinitialisation n'est transmise.

### 1.3.3.3 Reprise sur la liaison d'accès

En cas d'incident sur la liaison d'accès toutes les voies logiques supportée par celle-ci sont en défaut. On procède donc à une **libération de tous les circuits virtuels commutés** et à un **réinitialisation des circuits virtuels permanents**.

Les paquets de reprise sont émis sur la voie logique 0.

## 1.3.4 Structures des NPDU : formats de paquets

Nous ne décrirons pas en détail les formats de tous les paquets, mais nous donnerons leur structure générale et une description plus ou moins complète de certains d'entre eux.

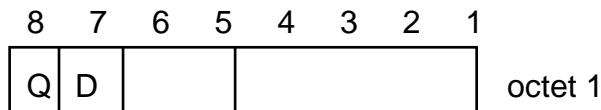
Les NPDU sont composées d'une en-tête (PCI) suivie éventuellement de données utilisateur. Elles sont composées d'un nombre entier d'octets.

### 1.3.4.1 En-tête de paquet

#### 1.3.4.1.1 Identificateur général de format

Les paquets pouvant faire l'objet d'une numérotation séquentielle normale (modulo 8) ou étendue (modulo 128), deux formats de paquets sont actuellement prévus. D'autres pourront être ajoutés. D'autre part nous avons noté que certains paquets pouvaient être "qualifiés" (bit Q) ou l'objet d'une confirmation de remise (bit D).

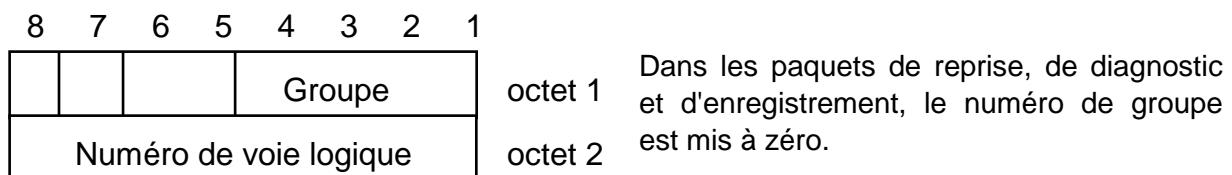
Un champ d'**identification générale de format**, codé sur 4 bits, permet de différencier ces cas : Il porte sur les bits 5 à 8 du premier octet.



- 0 1 : Numérotation modulo 8
- 1 0 : Numérotation modulo 128
- 1 1 : Extension de format

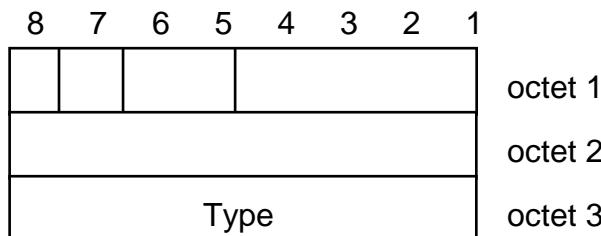
#### 1.3.4.1.2 Numéro de voie logique

Ce numéro est décomposé en deux champs : un numéro de groupe de voies logiques codé sur 4 bits et un numéro de voie proprement dit codé sur 8 bits.



#### 1.3.4.1.3 Identificateur de type de paquet

Le troisième octet code le type des paquets sur un (bit 1 à 0), cinq (bits 1 à 5) ou huit bits. Les paquets de données normales sont identifiés par le bit 1 de cet octet qui est mis à 0.



#### 1.3.4.2 Paquets d'appel

Ces paquets comportent une partie fixe suivie d'une partie optionnelle et éventuellement de données (d'appel) de l'utilisateur.

Pour les paquets de demande d'appel ou d'appel entrant l'identificateur de type vaut  $0B_H$  ; pour les paquets de confirmation et de communication établie, il vaut  $0F_H$ .

Le bit Q est à 0. Les autres bits de l'identificateur de format sont déterminés en fonction des besoins. Le numéro de voie logique est spécifié comme indiqué plus haut.

### 1.3.4.2.1 Adresse

L'adresse des E.T.T.D. est un paramètre **obligatoire** dans les paquets de **demande d'appel** et **d'appel entrant** et **facultatif** dans les paquets de **confirmation d'appel**. S'il existe ce paramètre est codé à partir du 4ème octet.

Le codage des adresses absolues des E.T.T.D. est de type longueur-valeur.

L'octet numéro 4 porte les longueurs (nombre de chiffres), codées sur 4 bits, des adresses des E.T.T.D. appelant et appelé.

Les octets suivants contiennent l'adresse proprement dite codées par une suite de 0 à 15 chiffres **décimaux**. Si le nombre de chiffres est impair il est complété par un 0.

L'octet 5 et les suivants contiennent l'adresse de E.T.T.D. appelé si elle existe puis l'adresse de E.T.T.D. appelant si elle est présente.

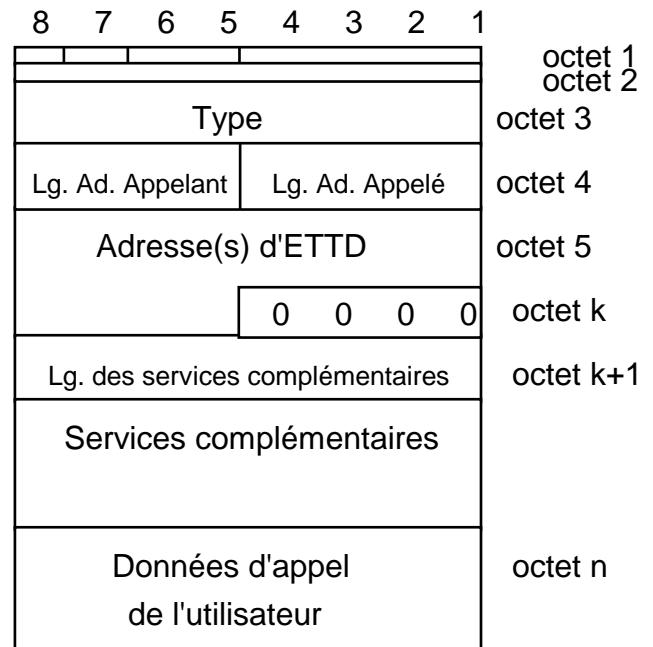
### 1.3.4.2.2 Services complémentaires

Les services complémentaires font l'objet du paragraphe 5. Nous ne donnons ici que le format permettant de les représenter.

Ce champ n'est présent que si un tel service est utilisé.

Il est codé sur un nombre entier d'octets (au maximum 109). Sa longueur, éventuellement 0, est codée sur l'octet qui suit immédiatement l'adresse.

Le codage de chaque service complémentaire est de type type-valeur. Le type est codé sur 1 octet. On distingue 4 catégories de services complémentaires (A,B,C,D) dont les paramètres sont codés respectivement sur 1, 2, 3 et n octets.



Les bits 7 et 8 de l'octet type indique la catégorie donc la longueur de la zone paramètre.

exemples :

Catégorie A :

8	7	6	5	4	3	2	1
0	1	X	X	X	X	X	X
Champ de Paramètres							

Catégorie B :

8	7	6	5	4	3	2	1
0	1	X	X	X	X	X	X
Champ de							
Paramètres							

Catégorie D :

8	7	6	5	4	3	2	1
1	1	X	X	X	X	X	X
Longueur réelle							
Champ de							
Paramètres							

#### 1.3.4.2.3 Données d'appel de l'utilisateur

Ce champ optionnel fait suite aux champs de services complémentaires. Sa longueur maximale est de 16 octets. Si l'on utilise le service complémentaire de "sélection rapide" (voir ci-dessous) elle peut être portée à 128 octets.

Ce champ est codé comme une suite d'octets.

#### 1.3.4.3 Paquets de libération, de réinitialisation et de reprise

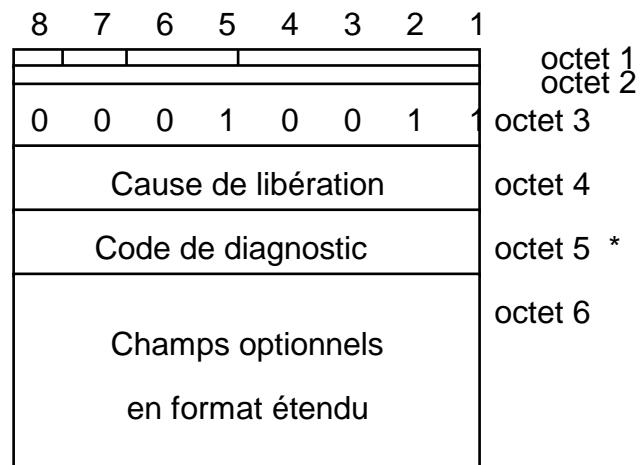
A la suite de l'en-tête standard de 3 octets les paquets de demande et indication de libération comportent des champs "cause" et "diagnostic" codés chacun sur un octet. Leur type est codé 13H. Les paquets de confirmation de libération ne comportent pas de champs "cause" et "diagnostic". Leur type est codé 17H.

Si l'on utilise le format étendu (lié au service de sélection rapide) on peut ajouter des champs facultatifs d'adresse(s), de services complémentaires et de données de libération de même nature que dans les paquets d'appels.

Dans les demandes de libération, le champ "cause de libération" peut prendre la valeur 0 ou une valeur supérieure ou égale à 128 (bit 8 mis à 1).

Les paquets de réinitialisation (demande ou indication, confirmation) ainsi que les paquets de reprise utilisent les mêmes formats que les paquets de libération correspondant en format de base. Ils sont codés respectivement  $1B_h$ ,  $1F_h$ ,  $FB_h$ ,  $FF_h$ .

Format d'une demande de libération.



- Le code de diagnostic n'est pas obligatoire dans les demandes de libération du format de base (pas de champs optionnels à sa suite).

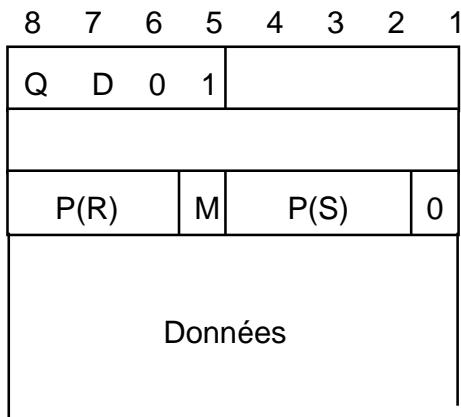
#### 1.3.4.4 Paquets de données normales et d'interruption

##### 1.3.4.4.1 Formats

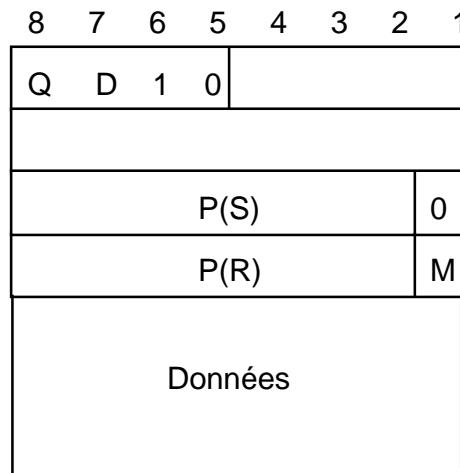
Ces paquets comportent une en-tête sur 3 octets (séquencement normal) ou sur 4 octets (séquencement étendu).

Le bit de poids faible (bit 1) du troisième octet, mis à 0, les identifient. Le bit M : bit 5 du 3ème octet ou bit 1 du 4ème octet (séquencement étendu) permet de chaîner les paquets en séquence complète.

### Séquencement normal



### Séquencement étendu



#### 1.3.4.4.2 Séquencement

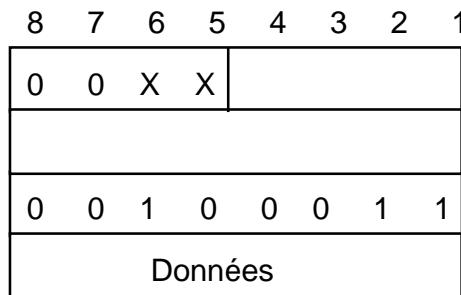
Les paquets de données normales sont séquencés par le champ P(S). Le champ P(R) indique le numéro de paquet **attendu**.

Le système travaille avec une anticipation W appelée "ouverture de fenêtre". Les mécanismes mis en jeu sont identiques à ceux étudiés au niveau 2/OSI pour le protocole HDLC. Si on n'utilise pas le mécanisme d'indication de remise (bit D) la mise à jour de ces champs est réalisée localement par le réseau. On ne peut alors tenir compte de leur valeur pour réaliser un contrôle de bout en bout.

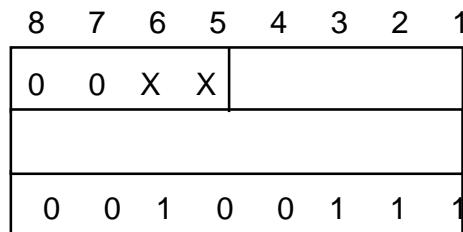
#### 1.3.4.4.3 Paquets d'interruption

Ces paquets permettent d'envoyer des données hors contrôle de flux. Ces données sont codées sur un octet. Un paquet d'interruption doit être confirmé par un paquet de confirmation d'interruption.

### Interruption



### Confirmation d'interruption



### 1.3.4.5 Paquets de supervision

#### 1.3.4.5.1 Paquets de contrôle de flux: RR et RNR. Paquets de rejet REJ

Ces paquets sont codés sur 3 octets en numérotation normale et sur 4 octets en numérotation étendue.

L'identificateur de paquet de RR est codé 01<sub>h</sub>. Celui de RNR est codé 05<sub>h</sub> et le paquet de rejet 09<sub>h</sub>.

exemple : paquet RR

Séquencement normal

8	7	6	5	4	3	2	1
0	0	0	1				
0	0	0	0	0	0	0	1

Séquencement étendu

8	7	6	5	4	3	2	1
0	0	1	0				
0	0	0	0	0	0	0	1
P(R)							0

#### 1.3.4.5.2 Paquets de diagnostic

Après l'en-tête standard de 3 octets, il comporte un champ d'un octet de code de diagnostic suivi d'un nombre entier d'octets d'explication de diagnostic. Ce type est codé F1<sub>h</sub>.

#### 1.3.4.5.3 Paquets d'enregistrement

Les paquets de demande d'enregistrement doivent être confirmés par un paquet de confirmation d'enregistrement. Ces paquets, dont le type est codé respectivement F3<sub>h</sub> et F7<sub>h</sub>, possèdent une en-tête , un champ d'adresse absolue (voir connexion), un champ de longueur d'enregistrement (sur 7 bits, bit de poids fort (bit 8) à 0) enfin un champ d'enregistrement pouvant avoir au maximum 109 octets.

### 1.3.5 Services complémentaires facultatifs d'usager

Ces services, actuellement au nombre de 34, sont négociés à la connexion par l'intermédiaire des "champs de facilité".

Ces services complémentaires sont nombreux et certains d'entre eux peuvent être regroupés en familles que nous décrirons rapidement.

### **1.3.5.1 Enregistrement en ligne des services complémentaires**

Ce service permet à tout moment à un E.T.T.D. de demander la liste des services complémentaires actuellement utilisés ou la mise en œuvre de services complémentaires. Ce service utilise les paquets d'enregistrement. La liste des services utilisés est fournie dans la confirmation d'enregistrement. L'envoi d'un paquet de demande d'enregistrement vide provoque l'envoi des services en cours.

### **1.3.5.2 Interdiction des appels ; voie logique unidirectionnel**

Quatre services complémentaires permettent :

- d'interdire tous les appels à l'arrivée
- d'interdire tous les appels au départ
- de limiter l'usage d'une voie logique aux appels entrants
- de limiter l'usage d'une voie logique aux appels sortants

### **1.3.5.3 Groupes fermés d'usager (GFU)**

Un groupe fermé d'usagers (GFU) est un ensemble d'E.T.T.D. autorisés à communiquer les uns avec les autres et pour lesquels la communication avec des autres E.T.T.D. est interdite. Cette restriction peut être limitée aux appels entrants, aux appels sortants ou être bilatérale.

Les services complémentaires ci-dessous sont relatifs aux groupes fermés d'abonnés :

- GFU avec accès entrant
- GFU avec accès sortant
- Interdiction des appels à l'arrivée dans un GFU
- Interdiction des appels au départ dans un GFU
- Choix du GFU
- Choix du GFU avec accès sortant
- Choix du GFU avec accès entrant
- GFU bilatéral
- Choix de GFU bilatéral

### **1.3.5.4 Taxation**

Plusieurs services complémentaires portent sur la taxation - Taxation à l'arrivé (PCV) Ce service est notamment utilisé lors d'un appel à travers le réseau téléphonique commuté lorsque l'appelant ne fournit pas de numéro d'identification d'usager (NUI). Voir chapitre suivant.

- Acceptation de la taxation à l'arrivée
- Interdiction de taxation locale
- Information de taxation
- Identification de l'usager du réseau

### 1.3.5.5 Sélection rapide

La sélection rapide est un service complémentaire qui permet de transférer des champs de données utilisateur de 128 octets maximum dans des paquets d'appel et/ou des paquets de libération.

Un service complémentaire d'acceptation de sélection rapide peut aussi être demandé pour autoriser le service précédent.

Le service de sélection rapide peut être demandé **avec ou sans restriction de réponse**.

- Avec restriction de réponse, l'E.T.T.D. appelé répond à un appel entrant avec sélection rapide par un paquet de libération qui peut contenir au maximum 128 octets de données.
- Sans restriction de réponse, l'E.T.T.D. appelé répond à un appel entrant avec sélection rapide soit par un paquet de libération soit par un paquet de communication établie qui peuvent porter au maximum 128 octets de données utilisateur. Dans ce dernier cas, il autorise, quand la communication est établie, l'E.T.T.D. et l'E.T.C.D. à émettre un paquet de demande de libération ou d'indication de libération, respectivement, pouvant comporter jusqu'à 128 octets de données.

### 1.3.5.6 Autres services complémentaires

- Numérotation étendue des paquets

Les champs P(S) et P(R) seront codés modulo 128. L'en-tête des paquets de données et de supervision a alors 4 octets.

- Modification de l'élément binaire D
- Retransmission des paquets

Ce service complémentaire permet la mise en oeuvre des paquet de rejet REJ

- Longueur de paquets par défaut non standard
- Dimension de fenêtre par défaut non standard
- Attribution de classes de débit par défaut
- Négociation des paramètres de contrôle de flux
- Négociation de classes de débit
- Choix de l'EPR (Exploitation privée reconnue)

Ce service permet de choisir un réseau de transit d'EPR pour acheminer une communication.

- Groupe de recherche Répartition des appels sur un groupe d'E.T.T.D.
- Réacheminement des appels
- Notification de modification de l'adresse de la ligne demandée
- Notification de réacheminement d'appel
- Sélection et indication du délai de transit

---

## 2 Accès à un réseau en commutation de paquets depuis un ETTD caractère : Protocoles X28 - X29 - X3

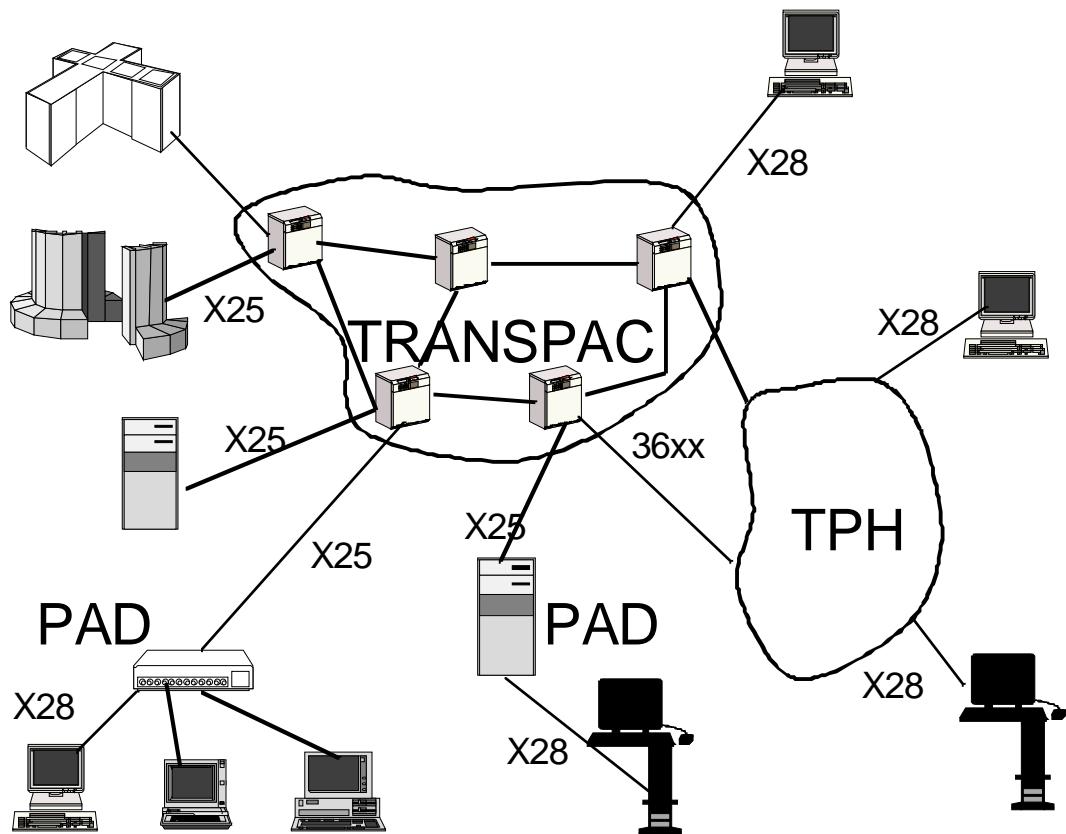
### 2.1 Présentation

Les équipements terminaux de traitement de données qui utilisent directement un service Réseau avec connexion fonctionnant en commutation de paquets sont relativement complexes et coûteux . Pour faciliter l'accès à ce type de réseau et notamment aux réseaux publics de données (comme Transpac en France) , un ensemble de protocoles a été spécifié. Ils permettent de raccorder des terminaux simples : visus, imprimantes, microordinateurs aux réseaux via des **assembleurs- désassembleurs de paquets (ADP ou PAD)** .

Ces composants sont :

- soit des logiciels disponibles sur les commutateurs d'entrées des réseaux en commutation de paquets.
- soit des matériels (qui supportent de tels logiciels) qui permettent de raccorder des ETTD caractères sur un accès (X25) à un réseau en commutation de paquets.
- soit des logiciels sur des miniordinateurs ou des stations de travail ayant un accès en mode X25 à un réseau en commutation de paquets.

Le schéma ci-dessous illustre ces diverses possibilités.



Ce système doit permettre la connexion d'une grande variété d'ETTD ; il doit donc s'adapter à leurs différentes caractéristiques : vitesse, contrôle de parité, écho, caractères de mise en page, longueurs de lignes, etc. Cette adaptation est réalisée en ajustant les valeurs d'un ensemble de **paramètres du PAD** qui caractérisent un type de terminal. Un ensemble cohérent de ces paramètres constituent un "**profil**" d'ETTD ..

## 2.2 Exemple d'utilisation

Le diagramme suivant illustre une communication entre une visu et un ordinateur distant interconnectés via un Réseau en commutation de paquets, par exemple Transpac. La visu est reliée au Réseau de données par l'intermédiaire du Réseau téléphonique commuté.

L'utilisateur, avant de pouvoir échanger des données, doit donc connecter son terminal au Réseau en commutation de paquets (à son sous-ensemble PAD), puis à l'ordinateur à travers ce réseau . Il pourra aussi adapter le PAD à son terminal en modifiant quelques paramètres ou en choisissant un profil particulier.

Dans l'exemple ci-dessous, la vitesse de transmission est déterminée par le choix du point d'entrée du réseau de données, grâce au numéro d'appel téléphonique.

Dans certains cas, le réseau peut déterminer automatiquement cette vitesse de transmission en analysant une chaîne de 2 ou 3 caractères prédéfinis qui doivent être émis dès que l'on passe sur le circuit de données (par exemple H+ ou .+. ou ↴.↓)

Réseau (ETCD)	sens	ETTD	Commentaires
envoi porteuse	← →	3606 Commutation voix-données	appel téléphonique 2400 b/s
reconnaissance de la vitesse	← → →	↔↔ Bienvenue	optionnel
	← →	SET? 1:1,10:72 PAR 1:1,10:72	modification du profil
Demande d'appel X25	← → →	169001886 COM	Appel réseau X25
Paquets de données X25	← →	Hello Coucou	Appel confirmé Caractères émis Caractères reçus
reset	←	Break	Demande de déconnexion
Paquets de données	→	la suite ..	Caractères reçus
Demande de libération	→	LIB DTE 000	Indication de libération
LIB	← →	.	
	←	138002459	Demande d'un nouveau CV

A la connexion, un PAD offre un profil par défaut. En cas d'accès par le réseau téléphonique commuté, il s'agit du "profil imple" normalisé; en cas d'accès par une liaison spécialisée ou sur un PAD local privé, tout profil initial peut être prédéfini .

## 2.3 Le service d'assemblage-désassemblage de paquets (Avis X3) .

### 2.3.1 Service

L'ADP (PAD) effectue un certain nombre de fonctions et présente plusieurs caractéristiques optionnelles . Son fonctionnement dépend des valeurs données à l'ensemble des variables internes, appelées paramètres de l'ADP ; cet ensemble existe indépendamment pour chaque ETTD.

Les fonctions de base de l'ADP comprennent :

- l'assemblage des caractères reçus de l'ETTD local en paquets
- le désassemblage des données utilisateurs des paquets reçus de l'ETTD distant, via le réseau de données
- l'établissement, la libération, la réinitialisation des circuits virtuels sur le réseau
- l'envoi de paquets de données lorsque les conditions adéquates sont réunies : caractère spécial d'envoi ou temporisation
- l'envoi de paquets d'interruption
- le traitement du signal de coupure (Break) de l'ETTD arythmique
- la génération des signaux de service et l'envoi de signaux de commande de l'ADP
- un mécanisme de positionnement et de lecture des paramètres actuels de l'ADP.

En option, l'ADP peut offrir :

- un mécanisme de sélection d'un profil normalisé
- la détection automatique de la vitesse de transmission, du code ou de la parité utilisées
- un mécanisme permettant à un ETTD distant de demander une communication entre l'ETTD arythmique local et un autre ETTD.

### 2.3.2 Paramètres de l'ADP

L'ensemble des paramètres de l'ADP est défini dans les tableaux suivants avec leurs valeurs obligatoires et optionnelles ainsi que les valeurs par défaut correspondant aux profils normalisés .

- Le premier paramètre permet, s'il est positionné à 1, à l'ETTD arythmique de repasser de l'état de transfert de données à l'état de commande en utilisant le caractère DLE (ctrl P) ou un autre caractère que l'on peut définir. Lorsque qu'aucun caractère n'est sélectionné, il est possible de revenir en mode de commande si le paramètre 7 (utilisation du signal de coupure, Break) est positionné à la valeur 8; on ne peut alors plus utiliser ce signal pour une autre fonction. Dans le cas où aucun mécanisme n'est prévu pour repasser en mode de commande (paramètres 1 = 0 et 7 ≠ 8), seul l'ETTD distant peut rompre la communication.
- Le paramètre 2 permet de générer ou non un caractère en écho à l'entrée du PAD. Il est utile, par exemple, pour masquer les mots de passe. Si le système distant génère aussi un caractère en écho les caractères sont doublés à l'écran .... La solution la meilleure consiste à garder, pour contrôle, l'écho sur le PAD (2:1) et de supprimer l'écho distant (par exemple par stty -echo) pour éviter un trafic inutile et coûteux sur le réseau X25.
- Le paramètre 10, débit binaire, n'est accessible qu'en lecture et ne peut être modifié.
- Le contrôle de flux entre ETTD arythmique et réseau peut être assuré dans les deux sens en positionnant les paramètres 5 et 12.
- Le paramètre 15 permet de valider les trois paramètres suivants (16,17,18) d'effacement caractère, de ligne ou d'affichage de la dernière ligne.

- Le ou les caractères d'envoi de données sont définis par le paramètre 3. Si aucun caractère n'est retenu, un délai de temporisation est fixé par le paramètre 4 . L'envoi sur temporisation est préférable lorsque l'ETTD est un microordinateur; il est tout à fait à déconseiller lorsque le message est saisi manuellement. L'unité de temps est le 20ème de seconde.

**Valeurs possibles et ensemble de valeurs combinées des paramètres de l'ADP  
(voir la remarque 1)**

Numéro de référence du paramètre	Description du paramètre	Valeurs pouvant être choisies		Signification du paramètre de l'ADP	Observation
		Obligatoires	Optionnelles (voir la remarque 2)		
1	Rappel de l'ADP par l'utilisation d'un caractère (E)	0 1	32 à 126	Impossible Caractère DLE Possible: en utilisant un caractère graphique défini par l'utilisateur	
2	Renvoi en écho (E)	0 1		Pas de renvoi en écho Renvoi en écho	
3	Choix de un ou plusieurs caractères d'envoi de données (E)	0 2 6 18 126		Pas de caractère(s) d'envoi de données Caractère CR Caractères CR, ESC, BEL, ENQ, ACK Caractères CR, EOT, ETX Tous les caractères des colonnes 0 et 1 et le caractère DEL	Valeur formée par la combinaison (2+4) Valeur formée par la combinaison (2+16) Valeurs formées par la combinaison (2+4+8+16 + 32+64)
4	Choix du délai de temporisation de repos (E)	0 20 255	1 à 19 21 à 254	Valeur du délai de temporisation de repos en 20 <sup>e</sup> de secondes	(voir la remarque 3)
5	Commande des dispositifs auxiliaires (E)	0 1	2	X-FERMÉ (DC1) et X-OUVERT (DC3) non utilisés X-FERMÉ et X-OUVERT utilisés (transfert de données) X-FERMÉ et X-OUVERT utilisés (transfert de données et commande)	
6	Commande des signaux de service d'ADP (E)	0 1	5 8 à 15	Pas de transmission de signaux de service d'ADP à l'ETTD asynchrone Transmission des signaux de service d'ADP dans le format normalisé Transmission des signaux de service d'ADP et du signal de service rapide d'ADP dans le format normalisé Transmission de signaux de service d'ADP dans un format dépendant du réseau	Valeur formée par la combinaison (1+4)

Numéro de référence du paramètre	Description du paramètre	Valeurs pouvant être choisies		Signification du paramètre de l'ADP	Observations
		Obligatoires	Optionnelles (voir la remarque 2)		
7	Choix du fonctionnement de l'ADP lors de la réception d'un signal de coupure en provenance de l'ETTD arythmique (E)	0 2 8 21	1 5	Rien Interruption Réinitialisation Interruption et indication de coupure  Echappement de l'état <i>transfert de données</i> Mise au rebut des données de sortie, interruption et indication de coupure	Valeur formée par la combinaison (1 + 4)  Valeur formée par la combinaison (1 + 4 + 16)
8	Mise au rebut des données de sortie	0 1		Remise normale des données Mise au rebut des données	
9	Remplissage après le retour du chariot (CR) (E)	0 1 à 7	8 à 255	Pas de remplissage après le retour du chariot (voir la remarque 4)  Nombre de caractères de remplissage insérés après le retour du chariot	
10	Retour à la ligne (E)	0 1 à 255		Pas de retour à la ligne  Nombre de caractères graphiques par ligne	
11 (lecture seulement)	Débit binaire de l'ETTD arythmique (E)	0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18	1	110 bit/s 134,5 bit/s 300 bit/s 1 200 bit/s 600 bit/s 75 bit/s 150 bit/s 1 800 bit/s 200 bit/s 100 bit/s 50 bit/s 75/1 200 bit/s 2 400 bit/s 4 800 bit/s 9 600 bit/s 19 200 bit/s 48 000 bit/s 56 000 bit/s 64 000 bit/s	Les valeurs appliquées dans les divers ADP dépendent de la gamme de débits de transmission de données de l'ETTD.  L'attribution de valeurs décimales à tous les débits connus a pour but d'éviter une future révision de la Recommandation
12	Contrôle du flux de l'ADP (E)	0 1		X-FERMÉ (DC1) et X-OUVERT (DC3) non utilisés pour le contrôle de flux  X-FERMÉ (DC1) et X-OUVERT (DC3) utilisés pour le contrôle de flux	

Numéro de référence du paramètre	Description du paramètre	Valeurs pouvant être choisies		Signification du paramètre de l'ADP	Observations
		Obligatoires	Optionnelles (voir la remarque 2)		
13	Insertion de l'interligne après retour du chariot (A)	0 1 4 5 6 7		Pas d'insertion d'interligne  Insérer un interligne après transmission du CR à l'ETTD arythmique  Insérer un interligne après le renvoi en écho du CR à l'ETTD arythmique  Insérer l'interligne après transmission à l'ETTD arythmique et après le renvoi en écho du CR  Insérer un interligne dans le train de données après un CR en provenance de l'ETTD arythmique et après le renvoi en écho d'un CR à l'ETTD arythmique  Insérer un interligne dans le train de données à destination et en provenance de l'ETTD arythmique et après le renvoi en écho d'un CR à l'ETTD arythmique	Combinaison (1+4)  Combinaison (2+4)  Combinaison (1+2+4)  <i>Remarque – ne s'applique qu'à l'état transfert de données</i>
14	Remplissage après l'interligne (A)	0 1 à 7 8 à 255		Pas de remplissage après l'interligne  Nombre de caractères de remplissage inscrits après l'interligne	<i>Remarque – ne s'applique qu'à l'état transfert de données</i>
15 (voir la remarque 5)	Édition (A)	0 1		Non recours à l'édition dans l'état transfert de données  Recours à l'édition dans l'état transfert de données	
16 (voir la remarque 5)	Effacement de caractère (A)	0 à 126 127		Un caractère de l'AI n° 5  Caractère 7/15 (DEL)	
17 (voir la remarque 5)	Effacement de ligne (A)	0 à 23 24 25 à 127		Un caractère de l'AI n° 5  Caractère 1/8 (CAN)  Un caractère de l'AI n° 5	
18 (voir la remarque 5)	Affichage de ligne (A)	18	0 à 17 19 à 127	Un caractère de l'AI n° 5  Caractère 1/2 (DC2)  Un caractère de l'AI n° 5	

Numéro de référence du paramètre	Description du paramètre	Valeurs pouvant être choisies		Signification du paramètre de l'ADP	Observations
		Obligatoires	Optionnelles (voir la remarque 2)		
19 (voir la remarque 5)	Signaux de service d'ADP d'édition (A)	1	0 2 8 32 à 126	Pas de signaux de service d'ADP d'édition  Signaux de service d'ADP d'édition pour terminaux à imprimante  Signaux de service d'ADP d'édition pour terminaux à écran  Signaux de service d'ADP d'édition utilisant un caractère de l'AI n° 5	
20 (voir les remarques 6 et 7)	Gabarit d'écho (A)	0	1 2 4 8 16 32 64 128	Pas de gabarit d'écho (tous les caractères sont renvoyés en écho)  Pas de renvoi en écho de CR Pas de renvoi en écho de LF Pas de renvoi en écho de VT, HT, FF Pas de renvoi en écho de BEL, BS Pas de renvoi en écho de ESC, ENQ Pas de renvoi en écho de ACK, NAK, STX, SOH, EOT, ETB, ETX Pas de renvoi en écho de caractères d'édition désignés par les paramètres 16, 17, 18 Pas de renvoi en écho de tous les autres caractères des colonnes 0 et 1 non mentionnés ci-dessus ainsi que de DEL	Des valeurs peuvent être formées par la combinaison de valeurs de base
21	Traitement de parité (A)	0	1 2 3	Pas de contrôle ou de production de parité  Contrôle de parité Production de parité Contrôle de parité et production de parité	Valeur formée par la combinaison (1 + 2)
22	Attente de page (A)	0 23	1 à 22 24 à 255	Attente de page désactivée  Nombre de caractères de changement de ligne considérés par l'ADP pour la fonction d'attente de page	

E: paramètre essentiel à prévoir au niveau international

A: paramètre additionnel qui peut être proposé sur certains réseaux de données et peut également être mis à disposition au niveau international

### 2.3.3 Profils

Le tableau ci-dessous, emprunté au réseau Transpac, donne quelques exemples de profils pour terminaux divers ou "Entrée Basse Vitesse d'Ordinateur" (EBVO) . Il n'est pas limitatif et sur un PAD privé ou un accès spécialisé à un réseau public, il convient de se définir un profil spécifique de l'ETTD connecté.

Paramètres Profil	1	2	3	4	5	6	7	8	9	10
0	1	1	126	0	1	1	2	0	0	0
1	0	0	0	20	0	0	2	0	0	0
2	0	0	0	10	1	0	21	0	0	0
3	1	0	2	80	0	1	21	0	0	0
4	1	0	2	40	0	1	21	0	4	0
6	1	1	126	0	1	1	21	0	0	0
9	1	0	2	0	0	1	0	0	0	69
11	0	0	0	3	0	0	21	0	0	0
12	1	0	126	0	1	1	2	0	0	0
13	1	0	0	5	1	1	21	0	0	0
14	0	0	0	5	1	1	8	0	0	0
15	1	0	0	5	0	1	21	0	0	0

- 0 Profil simple (défini par X28); écho assuré par le PAD, pas de borrage, pas de délai d'envoi de données.
- 1 Profil transparent (défini par X28). Convient pour les EBVO.
- 2 Profil recommandé pour les EBVO.
- 3 Profil pour terminal communiquant avec un autre ETTD-C, en particulier une EBVO; délai d'envoi de données pour permettre la transmission de données à partir de périphériques annexes; adapté aussi à certains terminaux transmettant par blocs.
- 4 Mêmes caractéristiques que le profil 3 mais avec délai d'envoi de données différent et 4 caractères de bourrage.
- 6 Mêmes caractéristiques que le profil 0 mais avec une procédure différente sur un signal Break.
- 9 Profil utilisé par TRANSPAC pour les accès des terminaux Telex.
- 11 Ce profil peut être utilisé à la place du profil 2 pour une EBVO sans asservissement, lorsqu'un court délai d'envoi de données est requis.
- 12 Identique au profil 0 sauf en ce qui concerne le paramètre 2, pour les terminaux ne nécessitant pas l'écho par le PAD.
- 13 Ce profil convient en particulier pour les terminaux en mode bloc ou message capables d'être asservis par le PAD et n'utilisant pas le DLE en transmission normale.
- 14 Mêmes caractéristiques que le profil 13 mais le (DLE) peut être utilisé (le passage en phase de commande se fait par un Break).
- 15 Mêmes caractéristiques que le profil 13 sans asservissement par le PAD.

**Positionnement des paramètres d'ADP**

Les références et les valeurs des paramètres se rapportent à la Recommandation X.3 (voir la remarque 1)

Numéro de référence du paramètre (voir la remarque 3)	Description du paramètre	Positionnement pour profils normalisés du CCITT (voir la remarque 2)	
		Profil normalisé transparent (voir la remarque 4)	Profil normalisé simple (voir la remarque 4)
1	Rappel de l'ADP au moyen d'un caractère	Mis sur <i>impossible</i> (valeur 0)	Mis sur <i>possible</i> (valeur 1)
2	Renvoi en écho	Mis sur <i>pas de renvoi en echo</i> (valeur 0)	Mis sur <i>renvoi en echo</i> (valeur 1)
3	Choix du signal d'envoi de données	Mis sur <i>pas de signal d'envoi de données</i> (valeur 0)	Mis sur <i>tous les caractères des colonnes 0 et 1 et sur le caractère 7/15 (DEL) de l'Alphabet international n° 5</i> (valeur 126)
4	Choix du délai de temporisation de repos	Mis sur <i>une seconde</i> (valeur 20)	Mis sur <i>pas de délai de temporisation</i> (valeur 0)
5	Commande de dispositifs auxiliaires	Mis sur <i>pas d'utilisation de X-FERMÉ et X-OUVERT</i> (valeur 0)	Mis sur <i>utilisation de X-FERMÉ et X-OUVERT</i> (valeur 1)
6	Commande des signaux de service d'ADP	Mis sur <i>pas d'envoi de signaux de service à l'ETTD arythmique</i> (valeur 0)	Mis sur <i>envoi des signaux de service</i> (valeur 1)
7	Choix du fonctionnement de l'ADP à la réception du signal de coupure provenant de l'ETTD arythmique	Mis sur <i>réinitialisation</i> (valeur 2)	Mis sur <i>réinitialisation</i> (valeur 2)
8	Mise au rebut des données de sortie	Mis sur <i>remise normale des données</i> (valeur 0)	Mis sur <i>remise normale des données</i> (valeur 0)
9	Remplissage après retour du chariot (CR)	Mis sur <i>pas de remplissage après CR</i> (valeur 0) (voir la remarque 5)	Mis sur <i>pas de remplissage après CR</i> (valeur 0) (voir la remarque 5)
10	Retour à la ligne	Mis sur <i>pas de retour à la ligne</i> (valeur 0)	Mis sur <i>pas de retour à la ligne</i> (valeur 0)
11 (en lecture seulement)	Débit binaire de l'ETTD arythmique	Indique le débit de l'ETTD	Indique le débit de l'ETTD
12	Contrôle de flux de l'ADP par l'ETTD arythmique	Mis sur <i>pas d'utilisation de X-FERMÉ et X-OUVERT</i> (valeur 0)	Mis sur <i>utilisation de X-FERMÉ et X-OUVERT</i> (valeur 1)
13 (voir la remarque 6)	Insertion d'interligne après retour du chariot	Mis sur <i>pas d'insertion d'interligne</i> (valeur 0)	Mis sur <i>pas d'insertion d'interligne</i> (valeur 0)
14 (voir la remarque 6)	Remplissage d'interligne (LF)	Mis sur <i>pas de remplissage après LF</i> (valeur 0)	Mis sur <i>pas de remplissage après LF</i> (valeur 0)
15 (voir les remarques 6 et 7)	Édition	Mis sur <i>pas d'édition dans l'état Transfert de données</i> (valeur 0)	Mis sur <i>pas d'édition dans l'état Transfert de données</i> (valeur 0)

Numéro de référence du paramètre (voir la remarque 3)	Description du paramètre	Positionnement pour profils normalisés du CCITT (voir la remarque 2)	
		Profil normalisé transparent (voir la remarque 4)	Profil normalisé simple (voir la remarque 4)
16 (voir la remarque 6)	Effacement de caractère	Mis sur le caractère 7/15 (DEL) (valeur 127)	Mis sur le caractère 7/15 (DEL) (valeur 127)
17 (voir la remarque 6)	Effacement de ligne	Mis sur le caractère 1/8 (CAN) (valeur 24)	Mis sur le caractère 1/8 (CAN) (valeur 24)
18 (voir la remarque 6)	Affichage de ligne	Mis sur le caractère 1/2 (DC2) (valeur 18)	Mis sur le caractère 1/2 (DC2) (valeur 18)
19 (voir les remarques 6 et 7)	Signaux de service d'ADP d'édition	Mis sur signaux de service d'ADP d'édition pour terminaux à imprimante (valeur 1)	Mis sur signaux de service d'ADP d'édition pour terminaux à imprimante (valeur 1)
20 (voir les remarques 6 et 8)	Gabarit d'écho	Mis sur <i>renvoi en echo de tous les caractères</i> (valeur 0)	Mis sur <i>renvoi en echo de tous les caractères</i> (valeur 0)
21 (voir la remarque 6)	Traitement de parité	Mis sur <i>pas de détection ou de production de parité</i> (valeur 0)	Mis sur <i>pas de détection ou de production de parité</i> (valeur 0)
22 (voir la remarque 6)	Attente de page	Mis sur <i>attente de page désactivée</i> (valeur 0)	Mis sur <i>attente de page désactivée</i> (valeur 0)

*Remarque 1* — Tous les paramètres normalisés par le CCITT sont énumérés dans le tableau 1/X.3, y compris ceux des services complémentaires additionnels offerts à l'usager énumérés dans la Recommandation X.2.

*Remarque 2* — Dans le cas de l'accès par un circuit loué, les valeurs appropriées des paramètres de profil sont spécifiées à la souscription de l'abonnement. En cas d'accès à partir de réseaux téléphoniques publics ou de réseaux de données publics à commutation de circuits, la définition d'autres profils normalisés du CCITT doit faire l'objet d'un complément d'étude.

*Remarque 3* — La référence de paramètre 0 n'est pas utilisée pour définir un paramètre d'ADP. La Recommandation X.29 prévoit l'utilisation spécifique de la valeur décimale 0 dans les messages d'ADP pour permettre l'existence de paramètres non définis par le CCITT. Une utilisation similaire de cette valeur dans la Recommandation X.28 nécessite un complément d'étude.

*Remarque 4* — Les procédures permettant à l'ETTD arythmique de choisir un *profil normalisé transparent* ou un *profil normalisé simple* sont actuellement définies par le recours au signal de *demande de service* ou au signal de *commande d'ADP de choix de profil normalisé*.

*Remarque 5* — Il n'y aura pas de remplissage, sauf que les signaux de service d'ADP contiendront un certain nombre de caractères de remplissage, en fonction du débit binaire de l'ETTD arythmique.

*Remarque 6* — Paramètre permettant dans certains pays d'obtenir des services complémentaires additionnels d'usager pour le service international et le service national (voir la Recommandation X.3). La mise en œuvre de ce paramètre dans un ADP doit être décidée à l'échelon national mais, dans le cas où ce paramètre est mis en œuvre, ce tableau donne les valeurs à utiliser lorsqu'un *profil normalisé* est choisi.

*Remarque 7* — Les fonctions d'édition s'appliquent pendant l'état *commande d'ADP* quelle que soit la valeur du paramètre 15. Les valeurs par défaut, ou les valeurs pouvant être choisies, des paramètres 16, 17, 18 et 19 s'appliquent à ces fonctions.

*Remarque 8* — Ce paramètre ne s'applique pas si le paramètre 2 est mis à zéro.

## 2.4 Interface pour accès à un service d'ADP (Avis X28) .

Cet interface permet de relier un ETTD arythmique à un service d'assemblage-désassemblage de paquets.

Il décrit, en particulier, une procédure d'établissement et de libération du circuit de données utilisant un modem (V21, V22 ou V23) connecté par une jonction V24/V28 et suivant l'Avis X20bis .

### 2.4.1 Signaux de commande et de service de l'ADP .

L'ETTD envoie des commandes à l'ADP sous forme de chaînes de caractères (et éventuellement d'un signal de coupure : Break) . Elle en reçoit des indications aussi par des chaînes de caractères

Si, pour les données utilisateurs tous les caractères de l'Alphabet International numéro 5 (sauf ceux retenus comme spéciaux par le profil actuel) peuvent être utilisés, pour passer des commandes un alphabet restreint est permis . Il n'utilise que les caractères suivants :

Majuscules : A..Z

Chiffres : 1 ... 9

Signes : : , ? +

Mise en page : espace, retour chariot et saut de lignes

Contrôle de flux : Xon (DC1) Xoff (DC3)

Divers : Nul (bourrage) et effacement (DEL)

La fin de commande est déterminée par l'envoi du caractère "retour chariot" ou du caractère " + " .

Certains ADP privés admettent des commandes en minuscule.

Les commandes qui peuvent être émises par l'ETTD et les indications qu'il peut recevoir (signaux de service) sont résumées dans les deux tableaux ci-dessous .

On notera en particulier la commande de libération : CLR . Dans certains réseaux la commande LIB peut être utilisée (par exemple dans Transpac) .

La lecture de tous les paramètres de l'ADP est faite par PAR . On peut n'en lire qu'un ou plusieurs en précisant les numéros :

PAR 1,6,12 par exemple .

Ces paramètres peuvent être positionnés tous ensemble en forçant un profil par PROF x . Un ou plusieurs peuvent l'être par SET m:x,n:y,p:z ou SET? m:x,n:y . Dans ce cas les valeurs prises sont renvoyées en écho .

### Signaux de commande d'ADP

Format du signal de commande d'ADP	Fonction	Signal de service envoyé en réponse (voir la remarque)
STAT	Demander une information d'état relative à une communication virtuelle établie avec l'ETTD	FREE ou ENGAGED
CLR	Libérer une communication virtuelle	CLR CONF ou CLR ERR (en cas d'erreur de procédure locale)
PAR? Liste des références des paramètres	Demander les valeurs actuelles des paramètres spécifiés	PAR (liste des références des paramètres avec leurs valeurs actuelles ou INV)
SET? Liste des références des paramètres et valeurs correspondantes	Demander la modification ou le positionnement des valeurs actuelles des paramètres spécifiés et demander les valeurs actuelles des paramètres spécifiés	PAR (liste des références des paramètres avec leurs valeurs actuelles ou INV)
PROF (identificateur)	Donner aux paramètres de l'ADP un ensemble normalisé de valeurs	Accusé de réception
RESET	Réinitialiser la communication virtuelle	Accusé de réception
INT	Transmettre un paquet d'interruption	Accusé de réception
SET Liste des paramètres avec les valeurs demandées	Positionner ou modifier les valeurs des paramètres	Accusé de réception ou PAR (liste des références des paramètres non valides suivis de INV)
Signal de commande d'ADP de sélection	Établir une communication virtuelle	Accusé de réception

Remarque — Les signaux de service d'ADP ne sont pas envoyés lorsque le paramètre 6 est mis à la valeur 0.

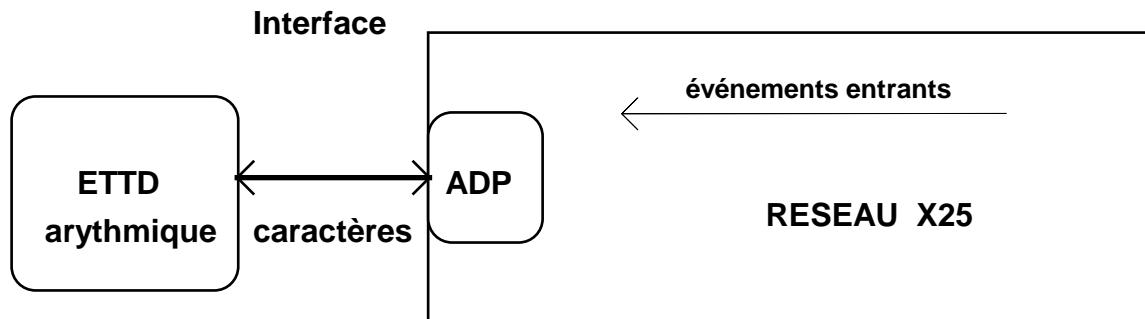
Format normalisé du signal de service d'ADP		Explication
RESET	DTE 1, 2 ou 3 caractères qui représentent la valeur ERR décimale du code de diagnostic (voir la remarque 1) NC RPE	Indique que l'ETTD distant a réinitialisé la communication virtuelle Indique que la communication virtuelle a été réinitialisée en raison d'une erreur de procédure locale Indique que la communication virtuelle a été réinitialisée en raison de l'encombrement du réseau Indique que la communication virtuelle a été réinitialisée en raison d'une erreur de procédure distante
CLR	Voir le tableau A-3/X.28	Indication de libération
CLR	CONF	Confirmation de libération
Voir la remarque 7	-	Indication de communication établie
	Les caractères à envoyer dépendent du réseau	Signal de service d'ADP d'identification d'ADP
ERR		Indique qu'un signal de commande d'ADP est erroné
Voir la remarque 2		Indication d'appel entrant
XXX		Indication d'exécution de la fonction d'effacement de ligne pour terminaux à imprimante (voir les remarques 3 et 4)
\		Indication d'exécution de la fonction d'effacement de caractère pour terminaux à imprimante
PAGE		Indique qu'une condition d'attente de page est intervenue
BS SP BS		Indication d'exécution de la fonction d'effacement de caractère pour terminaux vidéo (voir la remarque 4)
ENGAGED		Réponse au signal de commande d'ADP d'être quand une communication a été établie
FREE		Réponse au signal de commande d'ADP d'être quand une communication n'est pas établie
PAR	Valeur décimale de paramètre: valeur de paramètre, INV, ou liste de paramètres non valables	Réponse au signal de commande d'ADP de position et lecture et au signal de commande d'ADP de position si un paramètre au moins n'est pas valable
*		Signal de service d'ADP d'incitation
Caractère de mise en page		Signal de service d'ADP d'accuse de réception
TRANSFERT Services complémentaires d'adresse d'ETTD et/ou de données d'usager	Voir les remarques 5 et 6	Indique qu'une nouvelle sélection de l'ETTD appelé par l'ADP est en cours

**Signaux de service d'ADP d'indication de libération**

Signal de service d'ADP d'indication de libération	Symbolic mnémotechnique normalisée possible (voir la remarque)	Explication (voir la Recommandation X.96)
Ligne d'abonné occupée	OCC	L'ETCD décelle que l'ETTD appelé est occupé par d'autres communications et qu'il n'est pas en mesure d'accepter l'appel entrant
Encombrement du réseau	NC	Il existe dans le réseau une situation telle que: 1) encombrement momentané du réseau; 2) dérangement momentané du réseau
Demande de service complémentaire non valable	INV	Le service complémentaire demandé par l'ETTD appelant n'est pas valable
Interdiction d'accès	NA	L'ETTD appelant n'est pas autorisé à établir une communication avec le numéro demandé (par exemple, en cas d'incompatibilité due aux groupes fermés d'usagers)
Erreur de procédure locale	ERR	Une erreur de procédure causée par l'ETTD est décelée par l'ADP. Exemple: format incorrect
Erreur de procédure distante	RPE	Une erreur de procédure causée par l'ETTD est décelée par l'ETCD à l'interface ETTD/ETCD distante
Numéro non accessible	NP	L'adresse de l'ETTD appelé ne figure pas dans le plan de numérotation ou n'est affecté à aucun ETTD
Dérangement	DER	La ligne d'abonné demandée est en dérangement
Libération par l'ADP	PAD	La communication a été libérée par l'ADP local en réponse à une invitation à libérer émise par l'ETTD éloigné
Libération par l'ETTD	DTE	L'ETTD éloigné a libéré la communication
Acceptation de la taxation à l'arrivée non souscrite par abonnement	RNA	L'ETTD appelé n'a pas souscrit par abonnement à l'acceptation de la taxation à l'arrivée

## 2.4.2 Fonctionnement de l'interface . Diagramme d'état .

L'exemple donné ci-dessous illustre le fonctionnement de cet interface



A l'initialisation , il est placé en "**mode de commande**" :

l'ETTD peut passer à l'ADP une série de commandes, en particulier pour ajuster le profil .

La **dernière commande** de cette phase est l'envoi de l'adresse réseau de l'ETTD distant . Cette commande commence par un chiffre ; elle peut éventuellement comporter des lettres pour les services complémentaires (voir X25) .

Après l'établissement du circuit virtuel qui suit la réception de cette adresse, l'ADP passe en "**mode de transfert**" . Tous les caractères émis sont alors transmis ou utilisés pour la gestion du terminal (voir paramètres) à l'exception du caractère, défini par le paramètre 1 (ou 7), qui permet de repasser en mode de commande; il s'agit le plus souvent du caractère DLE (ctrl P , 10h)

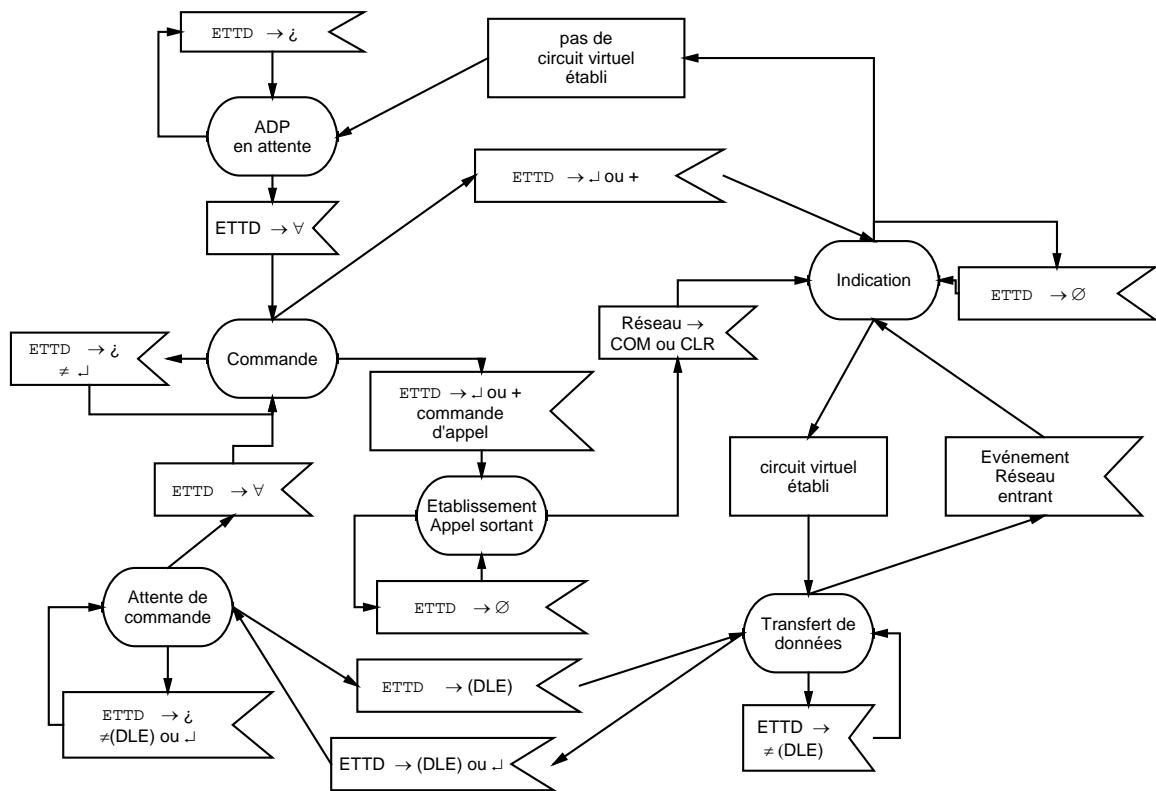
A l'émission de ce caractère spécial, l'interface repasse provisoirement en mode de commande, permettant l'envoi d'**une seule** commande terminée par "retour chariot" ou "+". Ce mécanisme permet, en particulier, de libérer la connexion. Il permet aussi de modifier des paramètres en cours de transfert, par exemple de supprimer ou de rétablir l'écho local.

L'automate d'états simplifié ci-dessous illustre le fonctionnement de cet interface ; il ignore les cas d'erreur, les expirations de temporisation ou les commandes issues de l'ETTD distant (voir paragraphe 5 ci-après) .

Cet automate est actif lorsque le niveau physique est établi .

Notations :

- ¿ : caractère non significatif
- ∀ : caractère significatif
- ↓ : retour chariot
- ∅ : caractère quelconque



Nota : Durant la phase d'indication, l'ADP émet un message pour l'ETTD (indication de ligne active à l'initialisation, indication due à une commande de l'ETTD distant ( voir § 5), réponse à une commande . La fin de cette phase se termine par l'envoi de la séquence retou chariot-saut de ligne .

## 2.5 Commande de l'ADP par l'ETTD distant (Avis X29).

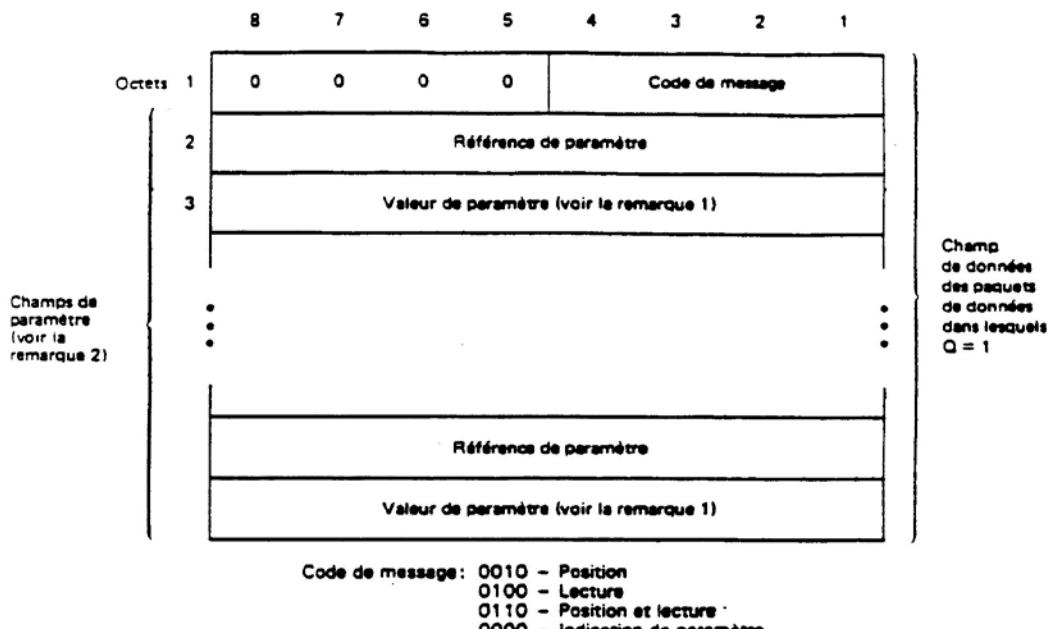
L'ADP peut non seulement interagir avec l'ETTD local mais aussi avec l'ETTD distant, si le protocole défini par l'Avis X29 est mis en oeuvre. Cet ETTD est habituellement un ETTD "paquet"; ce peut aussi être un autre ADP.

Ce protocole permet essentiellement de lire et de positionner les paramètres de l'ADP . Pour ce faire, l'ADP reçoit et émet des paquets de données qualifiés (bit Q = 1). Il peut aussi être utilisé pour inviter l'ADP à réinitialiser ou à libérer la communication.

Les tableaux ci-dessous donnent le format des paquets ainsi que le type et le codage des messages échangés par l'ADP pour traiter ces fonctions.

L'implantation de ce protocole n'est pas obligatoire; elle est toutefois très utile, en particulier pour faire ajuster le profil de l'ADP par l'ordinateur avec lequel l'ETTD arythmique dialogue. Dans ce cas, l'utilisateur transmet le type de son terminal à cet ordinateur ; celui-ci peut simultanément adapter sa vue du terminal (TERMCAP sous UNIX par exemple) et le profil du PAD, déchargeant ainsi l'utilisateur de cette tâche parfois délicate.

Ce mécanisme est aussi utilisé pour supprimer l'écho local lors de la transmission de séquences confidentielles (mots de passe) . Il permet aussi à l'ordinateur d'empêcher l'utilisateur de passer en mode de commande ...



Code de message: 0010 - Position  
 0100 - Lecture  
 0110 - Position et lecture  
 0000 - Indication de paramètre

Remarque 1 - Ces octets sont composés de 0 dans les messages d'ADP de lecture.

Remarque 2 - Champs de paramètre qui ne sont pas nécessairement présents (voir le tableau 1/X.29).

#### Type et codage de l'octet 1 des messages d'ADP

Type	Code de message				
	Eléments binaires	4	3	2	1
Message d'ADP de position . . . . .		0	0	1	0
Message d'ADP de lecture . . . . .		0	1	0	0
Message d'ADP de position et lecture . . . . .		0	1	1	0
Message d'ADP d'indication des paramètres . . . . .		0	0	0	0
Message d'ADP d'invitation à libérer . . . . .		0	0	0	1
Message d'ADP d'indication de coupure . . . . .		0	0	1	1
Message d'ADP de nouvelle sélection . . . . .		0	1	1	1
Message d'ADP d'erreur . . . . .		0	1	0	1

Remarque - La possibilité d'étendre le champ de code de message fera l'objet d'une étude ultérieure.



---

### 3 Couche 3/OSI : RESEAU SANS CONNEXION

#### C.I.P. Connectionless Internet Protocol

Ce standard définit un protocole de niveau 3/OSI, sous-couche supérieure SNICP (Subnetwork Independant Convergence Protocol), interréseaux locaux (ou étendus éventuellement) sans connexion.

Ce protocole suit la norme OSI\_8473 ou ECMA 92.

#### 3.1 Service fourni.

Cette sous-couche permet l'interconnexion de réseaux locaux sur un même site ou sur des sites distants via un réseau étendu (par exemple à commutation de paquets X.25).

Remarque : Ainsi il peut parfois être placé entre un protocole de niveau 4 et un protocole de niveau 3 (X25) avec connexion ce qui peut être paradoxal ...

Service sans connexion, il doit être très tolérant :

- aux fautes des sous-réseaux interconnectés
- aux défauts dans les ponts et autres sous-couches au niveau 3

Il permet de réaliser des fonctions :

- d'adressage
- de routage
- de composition / décomposition des IPDU
- de segmentation / réassemblage
- de surveillance des PDU pour purger celles dont la durée de vie est expirée.

Les seules primitives nécessaires à l'interface avec la couche 4 sont :

N\_UNITDATA.request  
N\_UNITDATA.indication  
                        avec pour paramètres  
les adresses source et destination  
la qualité de service  
les données utilisateur

Si la taille de la NSDU peut atteindre en théorie 65535 octets, pour l'interconnexion de réseaux locaux, elle est limitée à 1024 octets moins l'en-tête de l'IPDU.

La qualité de service comprend :

- le délai de transit
- la protection d'accès

- les déterminants de coûts
- la durée de vie maximale d'une NSDU

## 3.2 Service requis

Il est accessible par les interactions :

`SN_UNITDATA.request`  
`SN_UNITDATA.indication`

avec les mêmes paramètres que pour les interactions avec la couche supérieure.

Ceci suppose une **connaissance à priori de ce service** qui dépend de son implantation spécifique. Les fonctions utilisées sont dépendantes du protocole utilisé (avec ou sans connexion, réseau local ou étendu).

## 3.3 Fonctions traitées par le protocole.

### 3.3.1 Composition / décomposition des IPDU.

Cette fonction traite la PDU à émettre ou recevoir compte tenu de l'identification du protocole réseau des adresses de la segmentation éventuelle des données utilisateurs des paramètres fixes ou optionnels

### 3.3.2 Contrôle de la durée de vie de la PDU

A partir des informations de durée de vie qui accompagne la PDU (dans son en-tête) celle-ci est soit détruite soit reroutée.

### 3.3.3 Routage et acheminement

A partir de l'analyse de l'adresse et de la qualité de service (protection d'accès, délais de transit), cette fonction détermine :

- si la PDU est arrivée à destination
- si le domaine de destination est celui du sous-réseau local mais sur une autre station
- le sous-réseau qui permet d'atteindre la destination finale
- l'adresse à placer pour assurer l'acheminement dans le pont

### 3.3.4 Segmentation / réassemblage

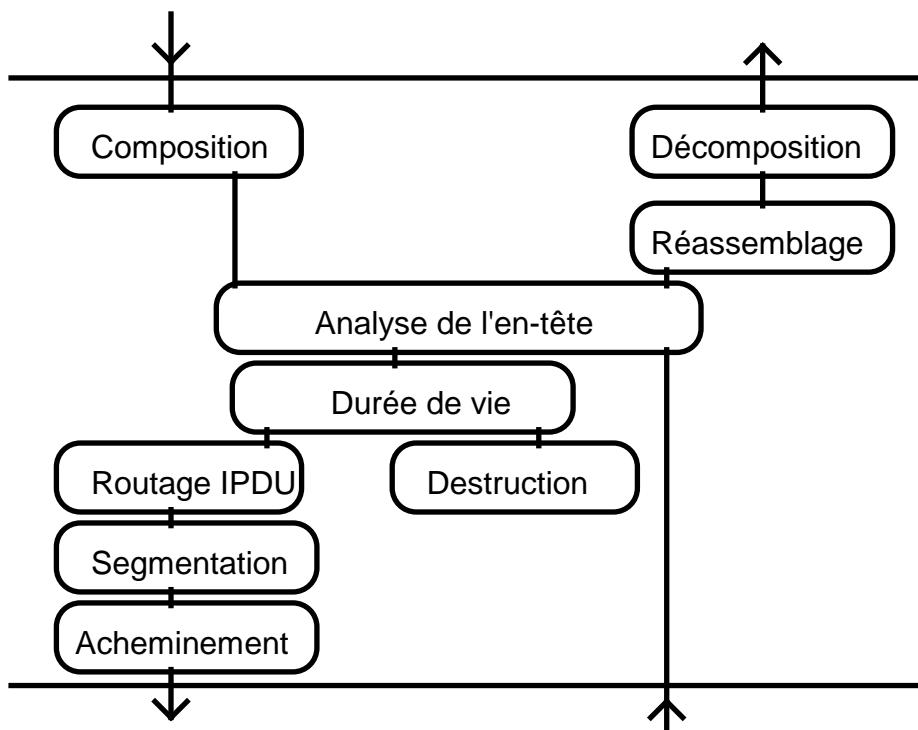
Cette fonction assure la fragmentation des IPDU en fonction des tailles de données des sous-réseaux et place dans ces IPDU segmentés les mêmes adresses source et destination et les mêmes identificateurs de données. (bits "données à suivre" et "segmentation permise")

### 3.3.5 Purge

Les PDU dont

- l'en-tête n'est pas analysable
- la durée de vie est expirée
- des segments manquent à l'expiration de la durée de vie, la taille est excessive et qui ne sont pas segmentables sont détruits. Il en est de même en cas de congestion du réseau.

Ces activités sont traitées dans l'ordre défini par le schéma ci-dessous :



### 3.3.6 Détection d'erreurs

Cette fonction est optionnelle.

## 3.4 Structure des PDU.

Les PDU sont composées d'une en-tête obligatoire et facultativement de données utilisateur.

L'en-tête (PCI) comporte 3 champs obligatoires et 2 champs optionnels :

- Identificateur de couche réseau
- partie fixe
- adresses
- partie segmentation (optionnelle)
- options (facultatives)

Pour le protocole décrit l'identificateur vaut 81 hexa.

### 3.4.1 Partie fixe:

Elle comporte 8 octets et porte les fonctionnalités de base du protocole : longueur de l'en-tête, version de protocole, **durée de vie** de la PDU, type de la PDU, fanions de segmentation (segment à suivre, non-segmentable), longueur du segment (PCI et données sur 2 octets) et 2 octets réservés pour le contrôle d'erreurs.

Champs	Octets
Indicateur de longueur	1
Identificateur de version	2
Durée de vie de la PDU	3
SP      MS      Type	4
Longueur de segment	5 - 6
Réserve pour détection d'erreurs	7 - 8

Deux types sont connus : Données et Erreur

### 3.4.2 Partie adresse :

Elle a une longueur dépendant du codage des adresses source et destination. Pour chaque adresse la longueur du champ est indiquée en tête.

### 3.4.3 Partie segmentation :

Pour une PDU segmentée, elle donne un identificateur de PDU, la position relative du segment et la longueur totale de la PDU (avant segmentation). Ces trois champs sont codés sur 2 octets.

Les paramètres optionnels sont codés en TLV (type, longueur, valeur).

---

## 4 Couche 4/OSI : Service TRANSPORT

### 4.1 Introduction

#### 4.1.1 Rôle:

Assurer entre des entités de Session un **Transfert de données transparent**

- en les déchargeant des détails de l'execution de ce transfert
- de manière fiable
- à un bon rapport qualité/prix, c'est-à-dire en optimisant l'utilisation du service Réseau
- avec une qualité de service définie.

Fournir à ces entités de Session un service de bout en bout.

Ainsi la couche Transport, utilisant les services des niveaux inférieurs, fournit un service de transmission "idéal" et optimisé.

#### 4.1.2 Service requis:

Service de niveau 3/OSI dans le cas d'un réseau étendu ou de niveau 2/OSI dans le cas d'une liaison point-à-point ou d'un Réseau local simple.

Si on dispose de toutes les fonctionnalités fournies par la couche Transport, ce service de niveau 2 ou 3 ne nécessite aucune qualité de service particulière.

#### 4.1.3 Service fourni:

- Etablissement de connexion avec une *qualité de service négociée*.
- Transfert de données :
- normales
- express

fiable et optimisé:

- Choix du meilleur service Réseau et multiplexage.

La taille des Unités de Données normales du Service de Transport (TSDU) peut être quelconque.

Libération de connexion.

Pour pouvoir fournir un service donné à partir de service de niveau inférieur quelconque, l'utilisateur peut choisir entre différents protocoles et, pour un protocole donné, entre différentes classes de service:

- Protocole de Transport avec connexion, classes 0 à 4.
- Protocole de transport sans connexion
- Protocole de Transport de masse point-à-point ou multipoint pour liaisons par satellites.

Cette multiplicité de classes permet de

- tirer le meilleur parti d'une connexion de Réseau
- assurer à un coût minimal une qualité de service donnée. Il est à noter toutefois que certaines fonctions ne sont pas assurées par toutes les classes.

Le choix d'une classe est réalisé d'après 2 ensembles de critères:

- besoins exprimés par l'entité de Session
- type et qualité de service de la connexion de Réseau.

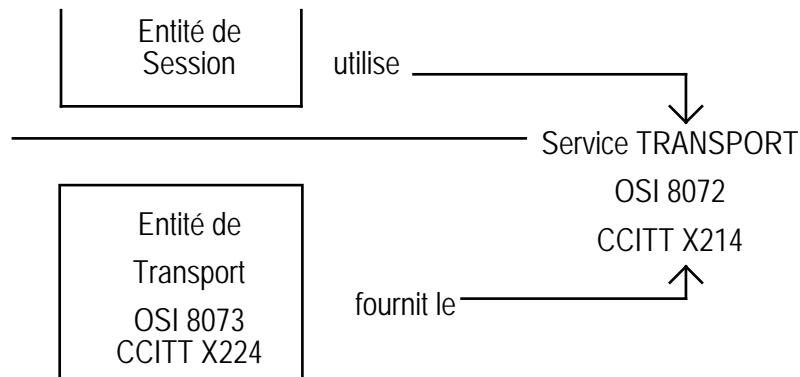
On distingue trois types de service Réseau:

- A : taux d'erreurs résiduelles et taux acceptables
- B : taux acceptable et taux d'incidents signalés
- C : taux d'erreurs résiduelles inacceptable pour des applications courantes.

Les classes 0 et 2 sont adaptées à un service Réseau de type A , les classes 1 et 3 à un service de type B. Un service Réseau de type C nécessite un service de Transport de classe 4 ( C'est souvent le cas pour un Réseau local).

## 4.2 Services Transport

### 4.2.1 Normalisation



### 4.2.2 Fonctions

Le Service Transport assure :

- Le choix de la qualité de service  
æ optimisation au moindre coût de l'utilisation des ressources de communication disponibles
- L'indépendance vis à vis de la qualité du Service Réseau, notamment en fonction des supports de communication
- La signification de bout en bout
- La transparence des informations transférées  
ie pas de restrictions au contenu, format, codage, des données utilisateurs qui sont considérés comme une suite d'octets.
- L'adressage de l'utilisateur du Service Transport donc, implicitement, de l'utilisateur du service Application.

Pour obtenir ces objectifs le Service Transport comporte une dizaines de fonctions de base dont deux seulement sont assurées par toutes les classes (transfert de données normales, connexion) et d'autres,optionnelles parfois, sont négociées à la connexion. Le tableau ci-dessous donnent la répartition de ces fonctions.Une liste plus détaillée est donnée plus loin.

Fonctions	Classes	0	1	2	3	4
Connexion (Etablissement et Refus)		x	x	x	x	x
Données normales (association avec une connexion)		x	x	x	x	x
Segmentation/réassemblage		x	x	x	x	x
Déconnexion explicite normale			x	x	x	x
Déconnexion explicite sur anomalie	x			x		
Données express			o	o	x	x
Numérotation des TPDU			x	c	c	c
Numérotation étendue de TPDU				o	o	o
Concaténation/Séparation		x	x	x	x	x
Mémorisation jusqu'à acquittement			c		x	x
Contrôle de flux explicite				c	x	x
Multiplexage				x	x	x
Gel des références		x			x	x
Reprise sur erreur / Resynchronisation			x		x	x
Redémarrage après coupure Réseau			x		x	
Retransmission sur temporisation						x
Contrôle d'inactivité						x
Correction des erreurs (retransmission)						x
Autodétection des erreurs						o
Eclatement						o
Reséquencement (après éclatement)						o

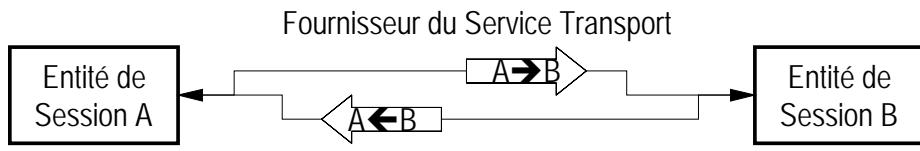
Pour certains réseaux (satellites par exemple) et qualités de service demandés (Transfert de masse) d'autres services Transport pourront être définis.

On peut ainsi dérouler la séquence typique suivante:

- Etablissement d'une connexion avec négociation d'une certaine qualité de service (Un même couple d'utilisateurs finaux peut utiliser plusieurs connexions de Transport)
- Transfert transparent d'un nombre entier d'octets de données normales avec **contrôle de flux**
- Transfert (optionnel), avec contrôle de flux **indépendant**, de données express
- Libération inconditionnelle (éventuellement destructive) de la connexion.

#### 4.2.3 Modélisation

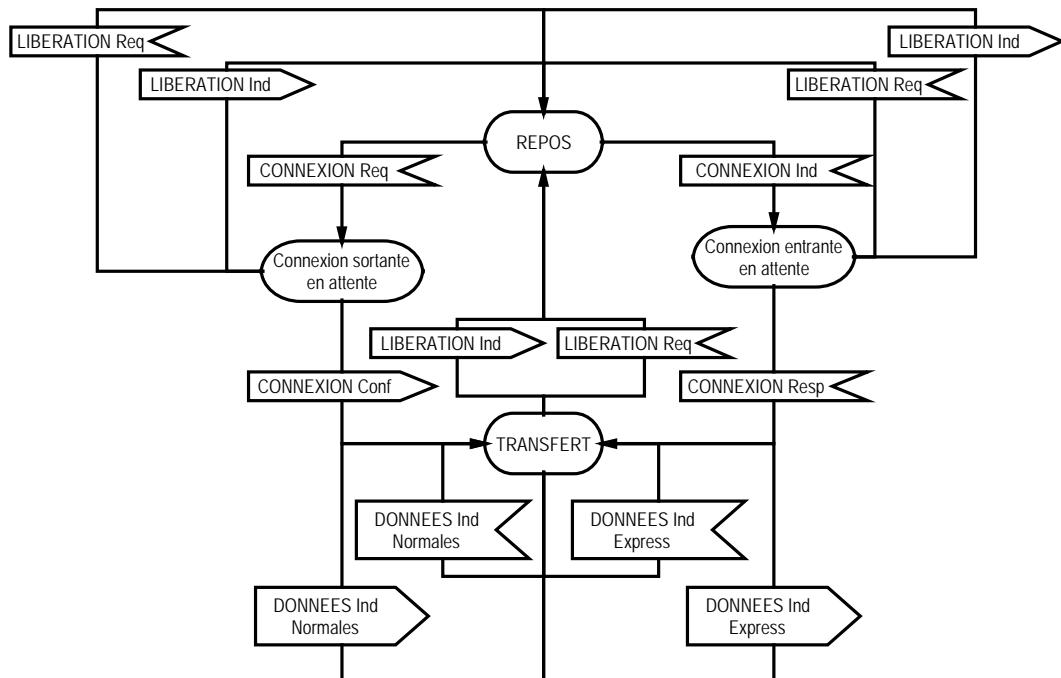
Le service Transport peut être modélisé par une double file d'attente avec les priorités



Les objets suivant sont gérés dans les files avec les priorités indiquées.

- objets relatifs à la déconnexion ie prioritaires (la terminaison peut être destructive)
- TSDU express ie priorité sur
  - fin de TSDU
  - données normales
- données normales
  - fin de TSDU
- objets relatifs à la connexion ie pas de priorité

Le service Transport peut aussi être modélisé par un automate d'états finis



#### 4.2.4 Paramètres des interactions

Hormis les données utilisateur normales ou express qui peuvent être passées dans pratiquement toutes les interactions (par valeur ou par pointeur.) les paramètres suivants sont fournis dans les requêtes ou indications de connexion.

- Classe demandée, classe de repli
  - Longueur des TPDU

Ces deux paramètres peuvent être figés à l'implantation

- ### • Adresses

Dans tous les cas il s'agit d'une adresse de TSAP (point d'accès au Service)

- appelante
  - appelée
  - en réponse normalement adresse appelée

æ éventuellement spécifique sur appel générique

- Option données express  
**Négociée** soit Service choisi ou non choisi
  - Qualité de service  
Débit, temps de réponse, etc

Dans les indications de déconnexion on aura la cause de cette déconnexion.

## 4.2.5 Négociation

Les paramètres négociables : classe de repli, longueur de TPDU supérieure à 128 octets, option sont traités de la manière suivante:

- L'option est demandée ou non par l'appelant
  - Si elle est non demandée, l'appelé ne peut pas la demander
  - Si elle est demandée l'appelé peut l'accepter ou la refuser

## 4.3 Protocole de Transport

Nous ne traiterons que le protocole de Transport OSI (8073) avec connexion ou son équivalent CCITT (X224)

### 4.3.1 Introduction

## Spécification

## Définition du Service Transport objectifs

© [G. BEUCHOT], [2001], INSA de Lyon, tous droits réservés.

## du protocole

fait référence à des

## moyens attendus ↴

## Définition du Service Réseau

#### **4.3.1.1 Buts:**

## Optimiser les coûts d'exploitation

### Améliorer la qualité de service

- débit
  - taux d'erreurs acceptable
  - impératifs d'intégrité des données
  - impératifs de fiabilité

Rq: Etant donné sa taille, l'implantation d'un tel protocole ne peut encore pas être totalement validée; cependant elle doit exécuter correctement le protocole dans un ensemble de circonstances représentatives.

#### 4.3.1.2 Définitions:

## Générales

- Concaténation/Séparation  
Mise en correspondance de plusieurs TPDU avec un NSDU (niveau N-1,Réseau)
  - Segmentation/réassemblage  
Mise en correspondance de un TSDU avec plusieurs TPDU
  - Multiplexage/démultiplexage  
Prise en charge de plusieurs N-Connexions (ici Transport) sur une (N-1)-Connexion (ici Réseau)  
ie mise en correspondance des adresses par une machine "mapping"
  - Eclatement/recombinaison  
Utilisation de plusieurs (N-1)-Connexions (Réseau) pour assurer une N-Connexion (Transport).
  - Contrôle de flux  
Asservissement de la source par le collecteur de données  
Ceci correspond à la notion de **Crédit**

## Spécifiques

- Classe Sous-ensemble du protocole
    - préférée
    - de repli
    - proposée Demandée par l'appelant
    - proposée Classe préférée ou de repli

- adoptée Choisie par l'appelé
- Référence Désignation **TEMPORAIRE** (pendant toute la durée d'une connexion) d'une des deux entités communiquant sur une connexion
  - utilisée
  - gelée existe mais non utilisable
  - non affectée

## 4.3.2 Fonctions

### 4.3.2.1 Etablissement de connexion

- choix du meilleur service Réseau
  - choix de multiplexage
  - choix d'une taille optimale de TPDU
  - choix des fonctions opérationnelles pour le transfert de données
  - transfert de données express
  - contrôle de flux explicite
  - récupération sur erreur
  - etc
- 
- mise en correspondance des adresses Transport et Réseau
  - identification de la connexion
  - (éventuellement) données utilisateur

### 4.3.2.2 Libération de connexion

### 4.3.2.3 Transfert de données

- segmentation/réassemblage
- concaténation/réassemblage
- contrôle de flux
- Identification de connexion
- données express
- délimiteur de TSDU
- selon la classe éclatement/recombinaison

## 4.3.3 Classes et fonctions

### 4.3.3.1 Réseau de type A:

- Classe 0
  - connexion
  - données normales
  - (libération par le Réseau)
- Classe 2
  - connexion
  - données normales
  - **multiplexage**
  - numérotation des TPDU
  - option:

- contrôle de flux
- données express
- numérotation étendue
- o concaténation
- o libération

#### 4.3.3.2 Réseau de type B:

- Classe 1
  - o connexion
  - o données normales
  - o numérotation des TPDU
  - o reprise sur défaut Réseau
  - o option : données express
  - o gel des références
  - o libération
- Classe 3
  - o **Toutes les fonctions de la classe 1**
  - o multiplexage
  - o contrôle de flux
  - o option : numérotation étendue

#### 4.3.3.3 Réseau de type C:

- Classe 4
  - o **Toutes les fonctions de la classe 3**
  - o autodétection d'erreurs
  - o contrôle d'inactivité
  - o retransmission sur temporisation
  - o éclatement

### 4.3.4 Adressage - Références

Le Protocole de Transport réalise une mise en correspondance des adresses entre  
adresse Réseau  
adresse Transport

Souvent l'adresse Transport sera construite en utilisant comme base l'adresse (absolue, par exemple X121) Réseau. L'adresse absolue identifiant un point d'accès au service Transport (TSAP) pourra être transmise (facultativement) dans les TPDU de connexion.

Pendant la durée d'une connexion de Transport, on utilise, sauf dans les TPDU de données en classe 0, des adresses temporaires codées sur 16 bits: les **Références**. Celles-ci sont échangées en phase de connexion comme indiqué ci-dessous:

Destinataire	Source
0	référence A
émis dans CR	

Destinataire	Source
0	référence A
reçu dans CR	

Destinataire	Source
référence A	référence B
reçu dans CC	

Destinataire	Source
référence A	référence B
émis dans CC	

Les TPDU de connexion (CR,CC) et de déconnexion contiennent les deux références (référence destinataire à 0 dans CR). Les autres TPDU ne contiennent que la référence destinataire.

### 4.3.5 Unités de Données de Protocole ::TPDU

#### 4.3.5.1 Structure

Toutes les TPDU ont une même structure indiquée ci-dessous:

En-tête (PCI)		Corps (à partir de la SDU)	
LI	partie fixe obligatoire	partie fixe optionnelle	données utilisateur optionnelles

LI : Longueur de l'en-tête (non compris LI) codée sur un octet. Maximum 254

Partie fixe :

Premier champ TYPE codé sur 4 bits

Les autres champs dépendent du type et sont explicités pour chaque TPDU par la suite.

On trouvera:

Référence destinataire	2 Ø
Référence source	2 Ø
Crédit initial	4 bits
Classe	4 bits !
Options	4 bits ! 1 Ø
Cause de déconnexion	1 Ø
Numéro de TPDU	7 bits !
Fin de TSDU (EOT)	1 bits ! 1 Ø
Numéro de séquence en réponse	7 bits - 1 Ø

Rq: en format étendu les numéros sont sur 31 bits

Partie variable :

Champs dans un ordre quelconque de structure TLV (Type,Longueur,Valeur)

- Type codé sur 1 octet (6 bits utilisés)
- Longueur du champ valeur codée sur un octet
- Valeur : chaîne d'octets

#### 4.3.5.2 Liste des TPDU

• Requête de Connexion	CR	toutes classes
• Confirmation de Connexion	CC	"
• Requête de Déconnexion	DR	sauf classe 0
• Confirmation de Déconnexion	DC	"
• Transfert de données normales	DT	toutes classes
• Transfert de données express	ED	sauf classe 0
• Acquittement de données normales	AK	"
• Acquittement de données express	EA	"
• Rejet	RJ	classes 1 et 3
• Erreur	ER	toutes classes
• Identificateur de protocole	PI	

#### 4.3.5.3 TPDU CR : Demande de connexion

C'est, avec CC, le TPDU le plus complexe  
partie fixe:

Type CR - E0h

Crédit initial 0 en classes 0,1 sinon < 15

Référence destination 0

Référence source choisie par entité émettrice

Classe bit 5 à 8, valeur 0 à 4

Options: bits 0 à 3 0 pas d'option

1 contrôle de flux explicite (cl 2)

2 format étendu

partie variable:

pour toutes les classes:

TSAP-ID appelant identificateurs de points

TSAP-ID appelé d'accès au service

Taille de TPDU  $\log_2(L) \rightarrow$  128 à 2k ou 8k octets sauf pour classe 0

ou pour classes indiquées:

Numéro de version

Paramètre de sécurité

Total de contrôle ---> Classe 4 seulement

Options additionnelles

bit 1 : données express

bit 3 : acquittement en classe 1

Classe de repli

Délai maximal d'accusé de réception  
Débits (moyen,maximal,minimal) souhaités  
Taux d'erreurs résiduel souhaité  
Temps de transit maximal acceptable (pour un TPDU de 128 octets, en millisecondes)  
Priorité  
Délai de réaffectation classes 1 et 3

Données utilisateur:  
optionnelles, 32 octets maximum

#### 4.3.5.4 TPDU CC: Confirmation de connexion (Type D0h)

mêmes paramètre que pour le TPDU CR

#### 4.3.5.5 TPDU DR : Demande de déconnexion

Partie fixe:

Type : 80h  
Référence Destinataire  
Référence Source  
Cause      1 octet  
0      cause inconnue  
1      encombrement TSAP  
2      Session non connectée  
3      Adresse inconnue  
128     Déconnexion normale

Partie variable:

Diagnostic  
Total de contrôle en classe 4

Données utilisateur:  
optionnelles, maximum 64 octets

#### 4.3.5.6 TPDU DC: Confirmation de déconnexion

Partie fixe:

Type : C0h  
Référence Destinataire  
Référence Source

Partie variable:

Total de contrôle en classe 4

#### 4.3.5.7 TPDU DT: Données normales

Classes 0,1

Partie fixe:

Type - F0h

Numéro de TPDU

Fin de TSDU bit 8 (= 1 dans dernier TPDU)

Données utilisateur:

Champ de taille maximale:

128,256,512,1024,2048 octets en classe 0

128,256,512,1k,2k,4k,8k octets en classe 1

Classes 2 à 4

Partie fixe:

Type - 0Fh

Référence Destinataire

Numéro de TPDU

Fin de TSDU bit 8 (= 1 dans dernier TPDU)

Partie variable:

Total de contrôle en classe 4

Données utilisateur:

Taille maximale : 128,256,512,1k,2k,4k ou 8k octets

#### **4.3.5.8 TPDU ED : Données express**

Type : 10h paramètres : Voir TPDU DT

Données utilisateur 1 à 32 octets

#### **4.3.5.9 TPDU AK : Acquittement de données normales**

Partie fixe:

Type : 60h

Crédit

Référence destinataire

YRTU-NR Numéro de TPDU attendu

Partie variable:

Total de contrôle en classe 4

#### **4.3.5.10 TPDU EA : Acquittement de données express**

Type : 20h . Paramètres voir TPDU AK

Pas de Crédit ( champ toujours à 0)

#### **4.3.5.11 TPDU RJ : Rejet**

En classes 1 et 3

Partie fixe:

Type - 50h

Crédit

Référence destinataire  
YRTU-NR      Numéro de TPDU attendu

#### 4.3.5.12 TPDU ER : Erreur de TPDU

Partie fixe:

Cause

- 0 non spécifiée
- 1 paramètre non valide
- 2 TPDU non valide
- 3 valeur de paramètre non valide

Partie variable:

- TPDU non valide
- Total de contrôle en classe 4

#### 4.3.5.13 TPDU PI : Identificateur de protocole

Partie fixe:

Type	- 01h
PRO-ID	identificateur
	1 OSI
	2 Teletex

Share Partage de protocoles sur le connexion Réseau

Rq: Les autres codes sont réservés, sauf 40h laissé libre (et utilisable pour des protocoles constructeurs spécifiques qui restent ainsi compatibles avec le Transport OSI).

### 4.3.6 Automates

#### 4.3.6.1 Automate simplifié pour classes 0 et 2

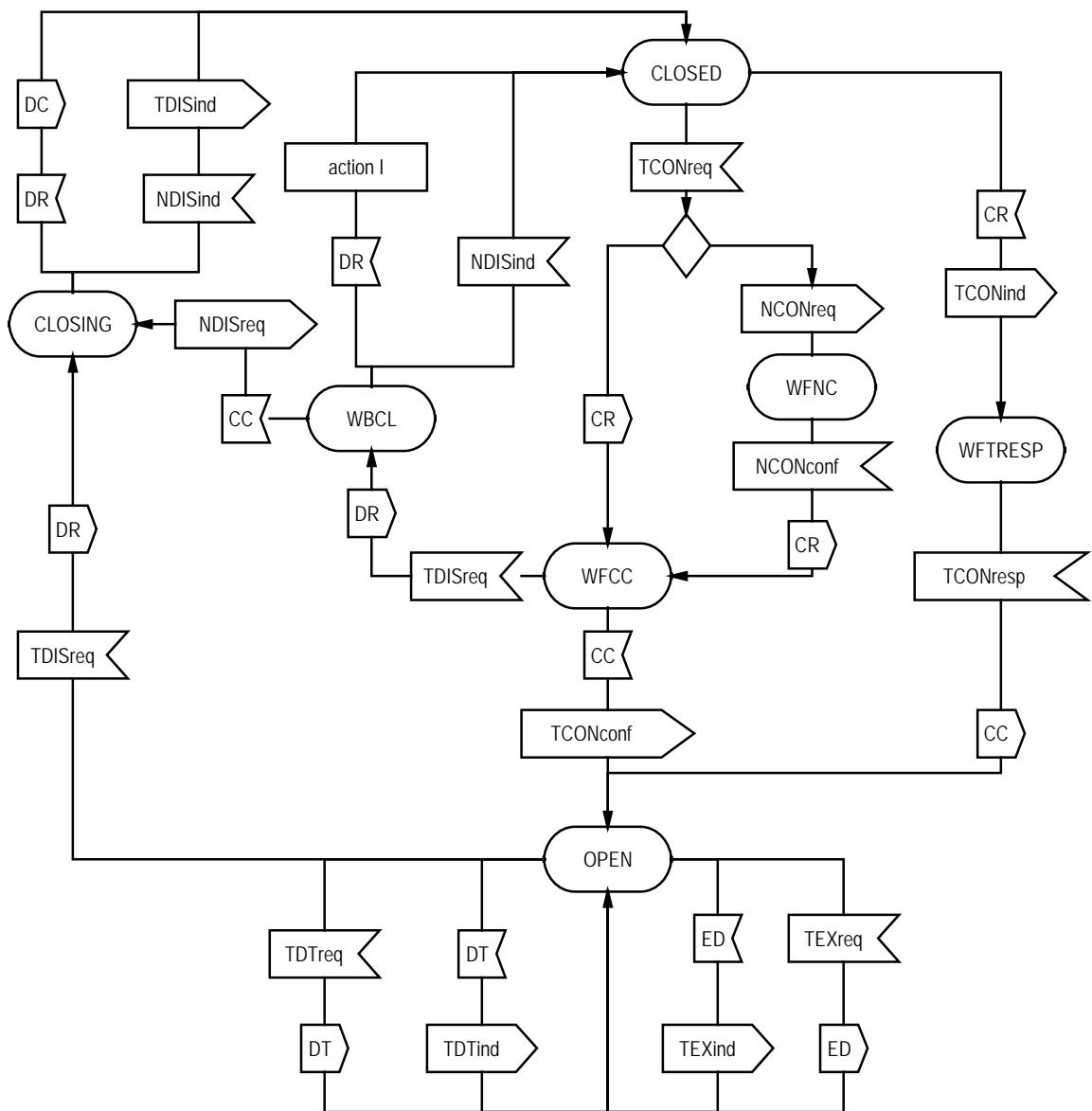
Événements

Rq : Les TPDU sont des événements sortants (NDATAreq) ou entrants (NDATAind)

Prédicats : seul le prédictat p2 "pas de connexion Réseau disponible" est pris en compte.

Action 1 : libérer la connexion Réseau si non utilisée

Entrants	Sortants	TPDU
TCONreq	TCONind	CR
TCONresp	TCONconf	CC
TDTreq	TDTind	DT
TEXreq	TEXind	ED
TDISreq	TDISind	DR
NDISind	NDISreq	DC
NCONconf	NCONreq	AK
NRSTind	NRSTresp	EA
		ER



#### 4.3.6.2 Automate de Transport OSI classes 0/2: Notations

Etats	1	WFNC	attente de connexion Réseau	2
	2	WFCC	attente de confirmation de connexion Réseau	
	3	WBCL	attente avant libération	
	4	OPEN	connexion de Transport ouverte	
	5	CLOSING	libération en cours	2
	6	WFTRESP	attente de réponse de connexion	
	0	CLOSED	Fermé; Repos	

Evénements entrants	cr ca dr er lr nc ri li	TCONreq TCONresp TDTreq TEXreq TDISreq NCONconf NRSTind NDISind	Requête de connexion de Transport Réponse de connexion de Transport Requête d'envoi de données Normales Requête d'envoi de données Express Requête de déconnexion Confirmation de connexion Réseau Indication de réinitialisation Réseau (sur anomalie) Indication de déconnexion Réseau	2
TPDU Evénements entrants ou sortants	CR CC DR DC AK EA DT ED ER	NDTreq ou NDTind	Demande de connexion Confirmation de connexion Demande de déconnexion Confirmation de déconnexion Accusé de réception de données Normales Accusé de réception de données Express Données Normales Données Express Erreur (anomalie)	2 2 2 2 2
Evénements entrants	ci cc di ei id rl rc	TCONind TCONconf TDTind TEXind TDISind NDISreq NCONreq	Indication de connexion de Transport Confirmation de connexion de Transport Indication d'envoi de données Normales Indication d'envoi de données Express Indication de déconnexion Requête de déconnexion Réseau Requête de connexion Réseau	III IV
Actions spécifiques (complexes)	I II III IV V		Libérer la connexion Réseau si elle n'est pas utilisée Traitement d'une erreur sur TPDU Transfert de données normales Transfert de données express Réponse NRESET émise sur la connexion réseau	
Prédicats	p0 p1 p2 p3 p4 p5 p6 p7 p8		Requête de connexion inacceptable Demande de connexion (CR) inacceptable Pas de connexion de Réseau disponible Connexion de Réseau disponible ouverte Connexion de Réseau disponible en cours d'ouverture Classe 0 choisie Confirmation de connexion inacceptable Classe 2 choisie Confirmation de connexion acceptable	

Remarques:

2 en classe 2 seulement

III voir action spécifique III

IV voir action spécifique IV

#### 4.3.6.3 Table de transition de l'Automate:

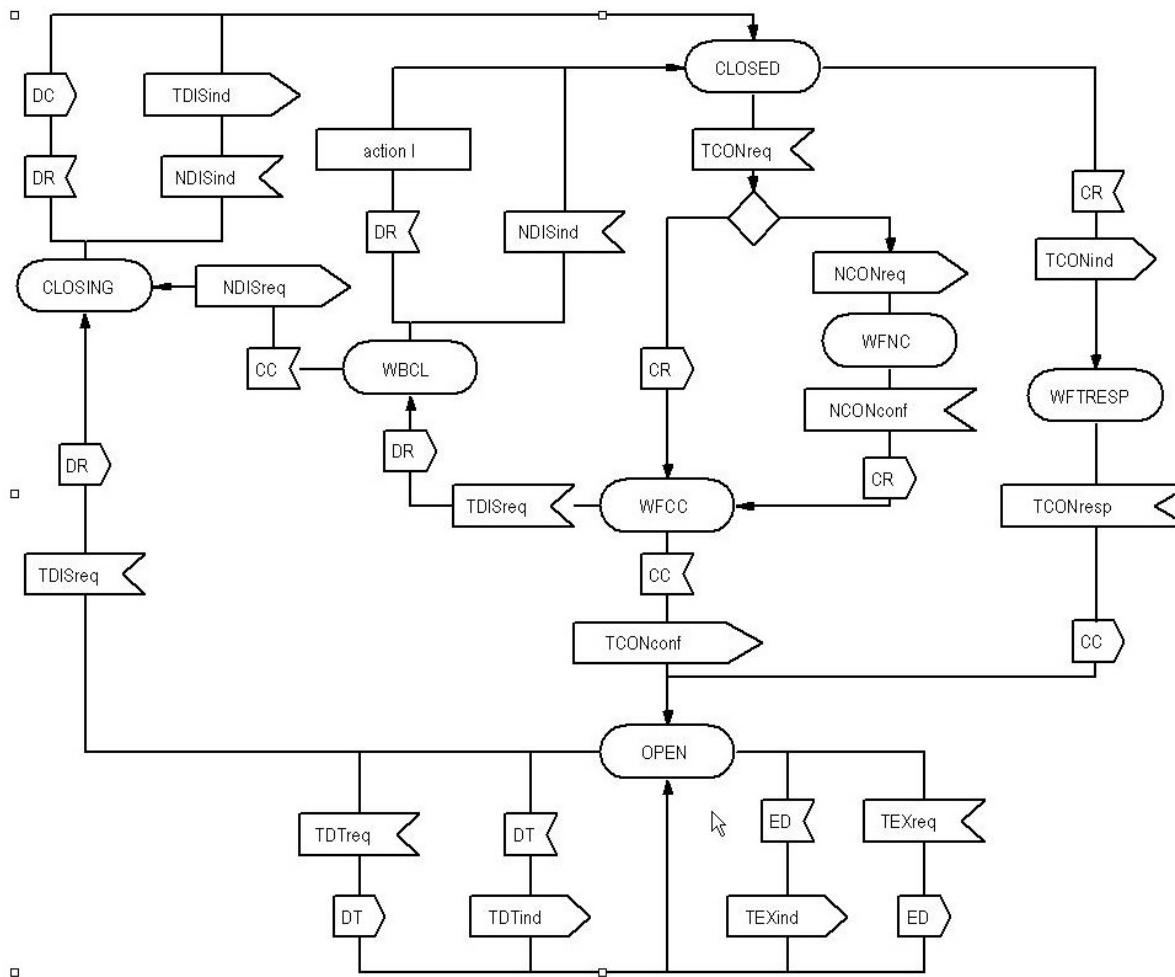
Etat Entrée	WFNC 1	WFCC 2	WBCL 3	OPEN 4	CLOSING 5	WFTRESP 6	CLOSED 0
cr							p0:id -0 p2:rc -1 p3:CR -2 p4: -1
ca							CC -4
dr				III -4			
er				p7: IV -4			
lr	I 0	- p5:rl -0 p7: -0		p5: rl -0		DR -0	
nc	CR 2	-					
ri		I;V;id -0	I;V -0	I;V;id -0	I;V -0	I;V;id -0	
li	id -0	id -0	-0	id -0	-0	id -0	
CR				p9: -4	p9: -5	p9: -6	p1:DR -0 p1:ci -6
DR		I;id -0	I -0	p5: nota p7:DC;id -0	I -0		
DC	nota : n'existe pas en classe O p7: I -0						
CC		p8:cc -4 P5&P6: id;rl -0 P7&P6: id;DR -5	p5:rl -0 p7:DR -5				DR -0
AK				p7:III -4 p5:nota	-5		-0
EA				p7:III -4 p5:nota	-5		-0
ED				p7:III -4 p5:nota	-5		-0
DT				III -4			-0
ER		I;id -0	I -0	II	II		-0

nota: si cette TPDU est reçue, elle doit être traitée comme erreur

#### 4.3.6.4 Graphe de l'automate:

Automate simplifié du Transport ISO classe 2

Etats de l'automate
CLOSED = repos
WFNC = attente réseau
WFCC = attente confirmation transport
WBCL = attente libération transport
OPEN = connecté
CLOSING = libération en cours
WFTRESP = attente réponse de connexion



## ANNEXE : Algorithme de Contrôle d'erreurs en classe 4

Le champ de contrôle placé en fin de PCI est composé de deux octets que nous noterons X et Y .

Si  $i$  est le numéro d'un octet de la TPDU,  $a_i$  la valeur de cet octet et  $L$  la longueur en octets de la TPDU, les valeurs de X et Y sont telles que :

$$\sum_{i=1}^L a_i \equiv 0 \pmod{255}$$

$$\sum_{i=1}^L i \cdot a_i \equiv 0 \pmod{255}$$

On utilise en émission et réception deux variables intermédiaires  $C_0$  et  $C_1$  . On note aussi n la position du premier octet de contrôle (X) dans la TPDU (numéro d'octet) .

### à l'émission :

Initialiser à 0 : X, Y,  $C_0$  et  $C_1$

Pour chaque octet  $i$  de 1 à  $L$  :

ajouter  $a_i$  à  $C_0$  ajouter  $C_0$  à  $C_1$

Calculer  $X = -C_1 + (L - n) * C_0$        $Y = C_1 - (L - n + 1) * C_0$

Placer X et Y dans les octets  $n$  et  $n + 1$

### à la réception :

Initialiser à 0 :  $C_0$  et  $C_1$

Pour chaque octet  $i$  de 1 à  $L$  :

ajouter  $a_i$  à  $C_0$  ajouter  $C_0$  à  $C_1$

Si  $C_0$  et  $C_1$  ne sont pas nuls la TPDU est incorrecte

Cet algorithme calcule :

$$C_1 \sum_{i=1}^L (L-i+1) \cdot a_i \pmod{255}$$

qui doit être égal à zéro puisque :

$$\sum_{i=1}^L (L-i+1) \cdot a_i = (L+1) \sum_{i=1}^L a_i - \sum_{i=1}^L i \cdot a_i \pmod{255}$$

---

## 5 Couche 5/OSI : Service SESSION

### 5.1 Présentation :

#### 5.1.1 Rôle:

Fournir aux entités de Présentation les moyens pour **ORGANISER** et **SYNCHRONISER** leur dialogue et **GERER** leurs échanges de données

Au cours de l'existence d'une connexion de Session , les services de Session **maintiennent l'état du dialogue** entre utilisateurs, même en cas de perte de données par le service Transport.

#### 5.1.2 Service requis:

Service de niveau 4/OSI fourni par une connexion de Transport. Il n'y a pas de multiplexage de plusieurs entités de Session sur une connexion de Transport et souvent on utilise la même adresse pour les entités de Session et de Transport.

#### 5.1.3 Services fournis:

Entre deux entités de Présentation on peut établir plusieurs connexions de Session

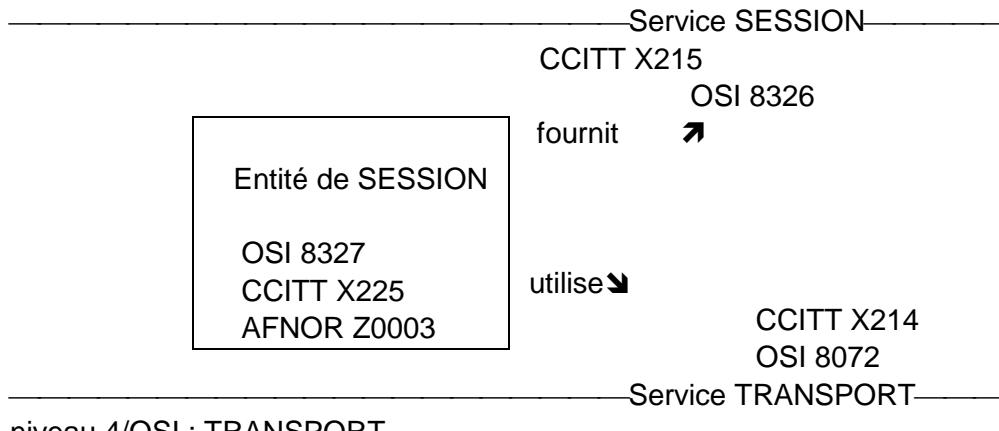
- simultanées
- ou - consécutives

Les services suivant peuvent être fournis:

- Etablissement de connexion
- Libération de connexion
- Echange de données
  - Normales
  - Express
  - Typées
- Mise en quarantaine
- Gestion de l'interaction
  - unilatérale
  - bilatérale \* simultanée (TWS)
  - \* à l'alternat (TWA)
- Synchronisation (Découpage temporel)
- Resynchronisation (reprises)
- Gestion d'Activité
- Rapport d'anomalies

### 5.1.4 Références

niveau 6/OSI : PRESENTATION



niveau 4/OSI : TRANSPORT

### 5.1.5 Définitions spécifiques:

*Jeton:* Attribut d'une connexion de Session attribué dynamiquement à un utilisateur du Service Session lui permettant de faire usage de certains services

*Mise en quarantaine:* Facilité de ne pas mettre à disposition de l'utilisateur destinataire un nombre entier d'Unités de données de service Session (SSDU) avant libération par l'utilisateur expéditeur

*Synchronisation:* Facilité permettant à des entités utilisatrices

- de définir et identifier des points de synchronisation
- de remettre une connexion de session dans un état prédéfini
- de convenir d'un point de resynchronisation

*Activité:* Unité logique de Travail pouvant s'étendre sur une partie, la totalité d'une ou plusieurs connexions de Session.

## 5.2 Service Session

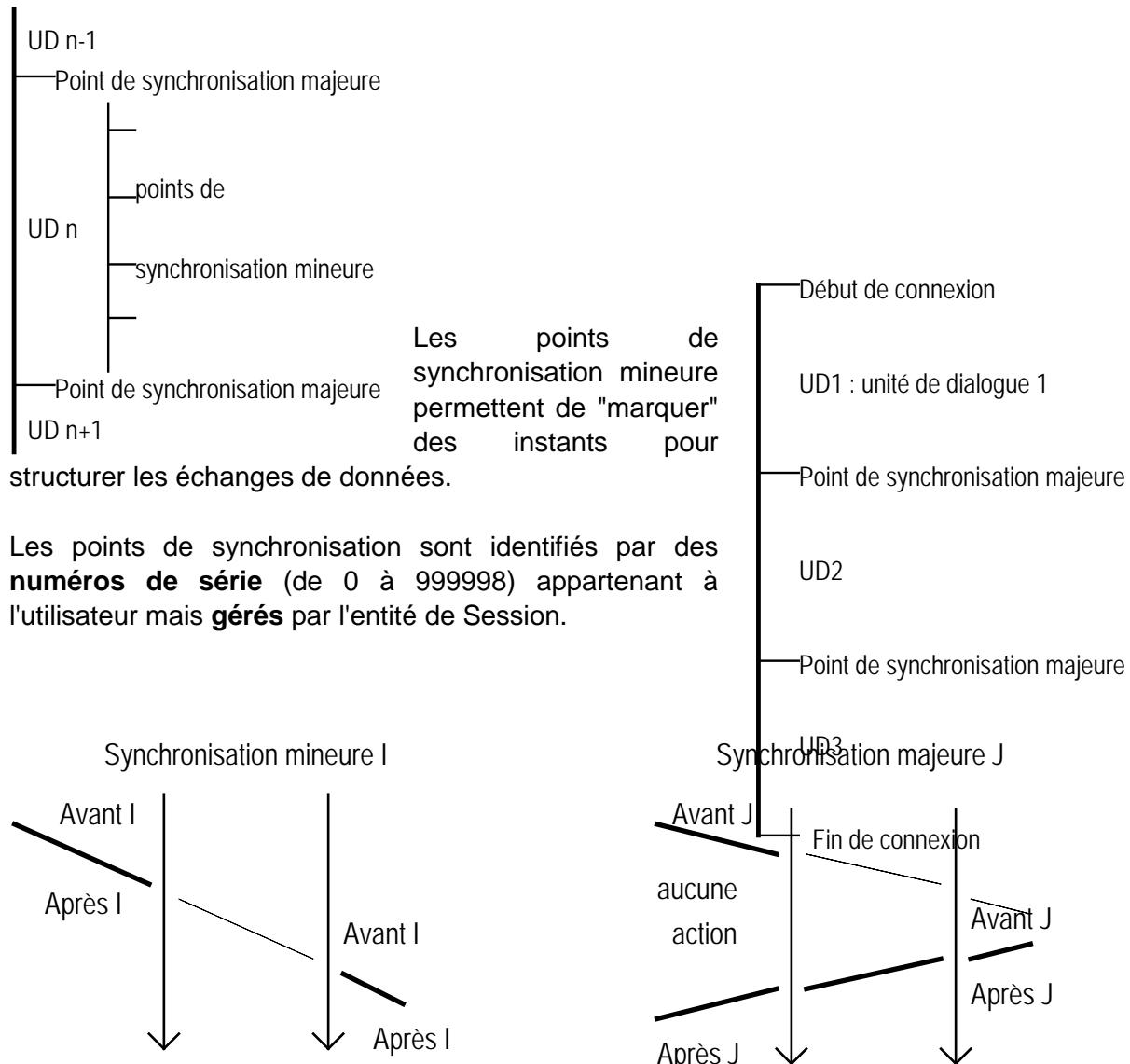
### 5.2.1 Découpage temporel d'une connexion de Session :

Unités de dialogue et points de synchronisation

La durée d'une connexion de Session peut être découpée en UNITES de DIALOGUE par des **Points de synchronisation majeure**.

Tous les éléments de communication échangés dans une unité de dialogue sont COMPLETEMENT SEPARES de ceux qui la précèdent ou la suivent.

Une connexion ou une unité de dialogue peut être structurée par des **points de synchronisation mineure**. Ces points peuvent CONFIRMER EXPLICITEMENT ou NON que cette confirmation explicite ait été demandée ou non. Même si une confirmation a été demandée celle-ci reste optionnelle et n'est pas forcément émise (sauf choix d'implantation spécifique).

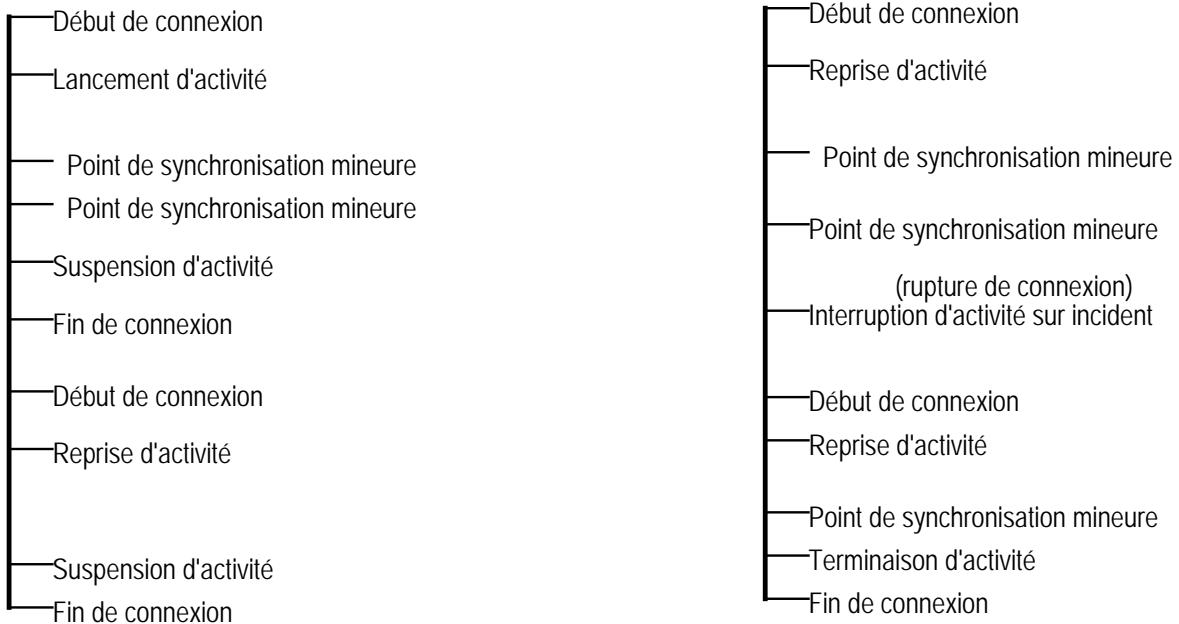


Pour agir, le SERVICE DE RESYNCHRONISATION utilise les points de synchronisation pour définir l'endroit où reprendre un échange de données.

### 5.2.2 Activité

Une ACTIVITE correspond à une unité logique de Travail

- Elle consiste en une ou plusieurs unités de dialogue
- Elle peut s'étendre sur
  - une partie d'une durée de connexion
  - une durée de connexion
  - plusieurs connexions



Ce concept sert à découper des travaux longs et fournit une forme de resynchronisation.

### 5.2.3 Resynchronisation

La resynchronisation remet la connexion de Session dans un **état défini**.

- Elle peut être lancée par une ou l'autre des entités de Session
- Elle élimine les données non remises (**purge**)
- Elle réinitialise les jetons (voir ci-dessous) et les numéros de points de synchronisation.

#### 3 OPTIONS :

- Abandon  
nouveau numéro de série
- Redémarrage  
reprise sur synchronisation mineure  
Numéro supérieur au dernier numéro de synchronisation majeure confirmé et inférieur ou égal au dernier numéro de synchronisation mineure.
- Choix utilisateur  
reprise sur numéro quelconque choisi par l'utilisateur.

### 5.2.4 Service de jetons

**4 jetons** sont définis:

- Jeton de données (droit de parole)

- Jeton de terminaison (droit de deconnexion)
- Jeton de synchronisation mineure
- Jeton de synchronisation majeure ou d'activité

Un jeton peut prendre **4 états** :

- Disponible
- |  |  |
|--|--|
| 1 ↪ attribué   | à un utilisateur   |
| 0 ↪ non attribué   | à l'autre utilisateur<br>qui ne peut utiliser le service<br>mais peut acquérir le jeton                              |
|  | <ul style="list-style-type: none"> <li>• Indisponible pour les 2 utilisateurs (synchronisation, activité)</li> </ul> |
| 3 ↪ attribué implicitement aux 2 utilisateurs (données ,terminaison) | <ul style="list-style-type: none"> <li>• Attribué implicitement ou explicitement</li> </ul>                          |
| 2 ↪  |  |

L'utilisation d'un service est limitée par une combinaison des états des 4 jetons.

### **échange de jetons**

Il peut être réalisé à l'initiative de l'entité qui détient les jetons par un "*don de jeton*" (service non confirmé : requête - indication).

L'entité qui ne détient pas les jetons peut aussi faire une "*demande de jeton*" (service non-confirmed). L'entité détentrice peut (mais ne doit pas forcément) réaliser un "don de jeton".

### **5.2.5 Unités fonctionnelles et sous-ensembles**

Une **unité fonctionnelle** est un regroupement logique de services de base

→ Il y a 19 services élémentaires

Les unités fonctionnelles sont définies pour négocier l'usage de ces services durant la **phase de connexion** et pour définir des **sous-ensembles cohérents** du service Session. Un **sous-ensemble** est une combinaison d'unités fonctionnelles comprenant :

- **le noyau**
- un jeu d'autres unités fonctionnelles suivant certaines conditions restrictives, exclusion mutuelle ou implication, pour assurer la cohérence :
 

duplex	↔/→ demi-duplex
info de capacité	→ activité
signalisation d'anomalie	→ demi-duplex

Rq: dans Architel activité ↔/→ données express

Il existe actuellement 3 sous-ensembles prédéfinis:

- BCS " combinaison de base"
- BSS " synchronisé de base"
- BAS " activité de base"

dont nous verrons ci-dessous la composition.



Unités fonctionnelles	Services
Noyau	Connexion Déconnexion Transfert de données Coupure
Déconnexion négociée	Echange de jetons Refus de déconnexion
Demi-duplex	Echange de jetons
Duplex	pas de service supplémentaire /noyau
Données express	Transfert de données express
Données typées	Transfert de données typées
Données de capacité	Echange de données de capacité
Synchronisation mineure	Echange de jetons Pose de points de synchronisation mineure
Synchronisation majeure	Echange de jetons Pose de points de synchronisation majeure
Resynchronisation	Resynchronisation
Signalisation d'anomalies	Anomalies fournisseur+utilisateurs
Gestion d'activité	Lancement d'activité Interruption d'activité (suspension) Reprise d'activité Abandon d'activité Terminaison d'activité Echange de jetons Passation du contrôle

### Composition des sous-ensembles:

Rq: Trois sous-ensembles sont définis actuellement. D'autres pourront être définis. Un sous-ensemble adapté à une application peut toujours être défini à partir des unités fonctionnelles existantes.

- BSC : " combinaison de base "
  - ➔ transfert de données normales et terminaison normale
    - noyau
    - duplex ou semi-duplex (choix)
- BSS : " synchronisé de base "
  - ➔ transfert de données normales et typées, terminaison négociée synchronisations majeure et mineure, resynchronisation
    - noyau
    - terminaison négociée
    - semi-duplex
    - données typées
    - synchronisation majeure
    - synchronisation mineure
    - resynchronisation
- BAS : " activité de base "

→ transfert de données normales et typées, terminaison normale, synchronisation mineure, activité, anomalies

- noyau
- semi-duplex
- données typées
- échange d'information de capacités
- synchronisation mineure
- gestion d'activité
- signalisation d'anomalies

Rq:échange d'information de capacités : si l'unité fonctionnelle "gestion d'activité" est utilisée et si aucune activité n'est en cours, ce service permet l'échange, confirmé, de données utilisateur en quantité limitée.

### **Unités fonctionnelles et jetons.**

Unité fonctionnelle	Jetons
Terminaison négociée	terminaison
Semi-duplex	données
Synchronisation mineure	Synchronisation mineure
Synchronisation majeure	Synchronisation majeure - activité
Gestion d'activité	Synchronisation majeure - activité

#### **5.2.6 Qualité de service**

- Paramètres négociables
  - protection de la connexion
  - priorité
  - taux d'erreurs résiduel
  - débit dans chaque sens
  - temps de transit dans chaque sens
  - transfert avec optimisation du dialogue concaténation en mode étendu (plusieurs SSDU de catégorie 2 dans un SPDU; voir protocole Session)
  - contrôle étendu utilisation de la coupure, interruption ou abandon d'activité, resynchronisation. En cas d'engorgement du flux normal **nécessité du flux de données express de transport**; ceci est fourni si on utilise l'unité fonctionnelle données express.
- paramètres prédéfinis ou négociés par les couches supérieures
  - délais d'établissement de connexion
  - probabilité de défaut de connexion
    - transfert
    - déconnexion
  - délai de déconnexion
  - "élasticité" de la connexion

### 5.2.7 Primitives

**Notations:**

- D : Demande, Requête
- I : Indication
- R : Réponse
- C : Confirmation

**liste:**

Primitive	Type	Paramètres principaux
Connexion	DIRC	identificateur, adresses, ...
Données (normales)	DI	donnés utilisateur
Données express	DI	donnés utilisateur
Données typées	DI	donnés utilisateur
Transfert d'information de capacité	DIRC	donnés utilisateur
Cession de jetons	DI	jetons
Demande de jetons	DI	jetons, donnés utilisateur
Passation de contrôle	DI	tous les jetons
Pose de points de synchro. mineure	DIRC	type, numéro de série, donnés utilisateur
Pose de points de synchro. majeure	DIRC	type, numéro de série, donnés utilisateur
Resynchronisation	DIRC	
Signalisation d'anomalies	DI	raison, données
Lancement d'activité	DI	identificateur, données utilisateur
Reprise d'activité	DI	identificateur, numéro de série
Interruption d'activité	DIRC	raison
Abandon d'activité	DIRC	raison
Terminaison d'activité	DIRC	numéro de série, donnés utilisateur
Terminaison de Session	DIRC	résultat, données utilisateur
Coupure par l'utilisateur	DI	donnés
Coupure par le fournisseur	I	raison

### priorité des primitives:

- données express
  - resynchronisation
  - interruption d'activité
  - abandon d'activité
  - coupure utilisateur
- ] peuvent être remises avant les autres primitives

➔ sinon fifo...

Rq: L'usage de certaines primitives est lié à la possession des jetons correspondants.

### 5.2.8 Synchronisation

Le mécanisme de synchronisation consiste à poser des repères, les **points de synchronisation**, au cours du dialogue pour s'affranchir de l'influence des temps de transmission entre entités communiquantes et pouvoir reprendre ce dialogue de manière fiable en cas d'incident.

#### Gestion des numéros de série des points de synchronisation

Les points de synchronisation sont caractérisés par un **numéro de série** codé en alphabet numéro 5 sur 6 octets et compris entre 0 et 999998. Le numéro 999999 est utilisé de manière interne pour satisfaire aux algorithmes mais jamais transmis.

- variables:

V(M) Prochain numéro à utiliser

V(A) Plus petit numéro affecté dont la confirmation est attendue. Si

V(A) = V(M) pas d'attente de confirmation

V(R) Plus petit numéro de série su lequel la reprise en resynchronisation est permise

Vsc Variable logique

si faux: pas de réponse à donner à une demande de pose de point de synchronisation mineure

si vrai: répondre à une demande de pose de point de synchronisation mineure si  $V(A) < V(M)$

- initialisation:

à la connexion  $Vsc = \text{faux}$

$V(A) = V(M) = \text{numéro initial}$

au lancement d'activité  $V(A) = V(M) = V(R) = 1$

$Vsc = \text{faux}$

à la reprise d'activité  $V(R) = 1$

$Vsc = \text{inchangé}$

$V(A) = V(M) = \text{numéro de série de point de synchronisation fourni par l'utilisateur} + 1$

#### Pose de points de synchronisation mineure

- Une confirmation explicite n'est pas exigée par la couche Session et peut donc ne pas être émise même si la demande en est faite par l'utilisateur
- Une confirmation explicite peut toujours être fournie même si elle n'a pas été demandée
- Les règles d'accusé sont définies entre utilisateurs  
Sur une demande de pose de point de synchronisation mineure

- numéro de série associé  $\leftarrow V(M)$
- $V(R)$  inchangé
- Si  $V_{sc} = \text{vrai}$   $V(A) \leftarrow V(M)$   
 $V_{sc} \leftarrow \text{faux}$
- envoi SPDU
- $V(M) \leftarrow V(M) + 1$

Sur réception d'une pose de point de synchronisation

- notification d'une indication pose de pt. de sync.
- Si  $V_{sc} = \text{faux}$   $V(A) \leftarrow V(M)$   
 $V_{sc} \leftarrow \text{faux}$
- $V(M) \leftarrow V(M) + 1$
- $V(R) \leftarrow$  inchangé

Sur une réponse à une demande de pose de point de synchronisation

- $V_{sc}$  doit être vrai et  $V(A) - 1 < \text{numéro fourni par utilisateur} < V(M)$
- envoi d'un accusé avec numéro fourni
- $V(A) \leftarrow \text{numéro fourni} + 1$
- $V(M), V(R), V_{sc}$  inchangés

Sur réception d'un accusé de réception de point de synchronisation

- $V_{sc}$  doit être faux et  $V(A) - 1 < \text{numéro de série} < V(M)$
- notification d'une confirmation de pt. de synchro.
- $V(A) \leftarrow \text{numéro de série} + 1$
- $V(M), V(R), V_{sc}$  inchangés

### Pose de points de synchronisation majeure

Le mécanisme est le même que pour la pose de points de synchronisation mineure , mais

- **tous les points sont confirmés explicitement**
- donc , sur une réponse ou une confirmation le **numéro de série associé est toujours égal à  $V(M) - 1$**
- Sur émission ou réception d'un accusé  $V(R) \leftarrow V(A)$

## Terminaison d'activité et pose de point de synchronisation

Une terminaison d'activité est équivalente à une pose de point de synchronisation.

### Resynchronisation

Les valeurs des numéros de série associés dépendent de

l'option	redémarrage	choisie
	abandon	
	choix utilisateur	

$V(R)$  fournit une valeur minimale au numéro de série associé à la reprise pour les options redémarrage ou choix utilisateur. Si l'option abandon est choisie, ce numéro sera pris égal à  $V(M)$  de la machine expéditrice.

### Préparation

Si le service "données **express** de Transport" est fourni au niveau 4/OSI des SPDU "**préparation**" sont émises par ce flux de Transport express avant les SPDU de synchronisation **majeure** (pose ou accusé) et de resynchronisation pour notifier l'arrivée imminente de ces SPDU par le flux normal et que certaines SPDU peuvent être mises au rebut.

## 5.3 Protocole de Session

### 5.3.1 Fonctions de la couche Session

- A partir des services fournis par la couche Transport, ces fonctions doivent assurer le service Session.

Elles concernent :

- la gestion du dialogue
- le synchronisation des flux de données et leur resynchronisation.

- La durée d'une connexion de Session est divisée en **trois phases** :

#### Etablissement de connexion

- mise en correspondance des adresses Session et Transport.
- choix des paramètres de Q.O.S. requis
- négociation des paramètres de Session
- éventuellement - transfert des SSAP
  - distinction entre Sessions
  - transfert limité de données

#### Transfert de données

- données normales    - duplex
  - semi-duplex
- données typées (non soumises à jeton)
- données express
- gestion de jetons
- signalisation d'anomalies
- pose de points de synchronisation    - mineure
  - majeure
- resynchronisation
- gestion d'activité    - initialisation
  - reprise
- échange d'informations de capacité (confirmées)

## **Déconnexion**

- terminaison normale - négociée
  - non négociée
- coupure - fournisseur
  - utilisateur
  - données(éventuellement)

### 5.3.2 SPDUs affectées aux unités fonctionnelles

Il existe 34 SPDUs différentes dont quelques unes sont affectées à plusieurs unités fonctionnelles :

Unités fonctionnelles	Code	SPDU
Noyau	CN AC RF FN DN AB AA DT	demande de connexion acceptation de connexion refus de connexion terminaison de connexion demande de déconnexion demande de coupure acceptation de coupure transfert de données normales
Terminaison négociée	NF GT* PT*	non terminé don de jeton demande de jeton
Semi-duplex	GT* PT*	don de jeton demande de jeton
Duplex		pas de SPDU supplémentaire
Données express	EX	données express
Données typées	TD	données typées
Echange d'informations de capacité	CD CDA	donnés de capacité acquittement de données de capacité
Synchronisation mineure	MIP MIA GT* PT*	point de synchronisation mineure acquittement de point de synchronisation mineure don de jeton demande de jeton
Synchronisation majeure	MAP MAA GT* PT*	point de synchronisation majeure acquittement de point de synchronisation majeure don de jeton demande de jeton
Resynchronisation	RS RA PR*	Demande de resynchronisation Acquittement de resynchronisation préparation de resynchronisation
Signalisation d'anomalies	ER ED	Signalisation d'anomalies Informations d'anomalies

Gestion d'activité	AS	lancement d'activité
	AR	reprise d'activité
	AI	interruption d'activité
	AIA	acquittement d'interruption d'activité
	AD	abandon d'activité
	ADA	acquittement d'abandon d'activité
	AE	terminaison d'activité
	AEA	acquittement de terminaison d'activité
	PR*	préparation (resynchronisation)
	GT*	don de jeton
	PT*	demande de jeton
	GTC	cession des jetons
	GTA	acquittement de cession des jetons

**remarques:**

- une implantation doit pouvoir soit
  - émettre CN et recevoir AC ou RF
  - recevoir CN et émettre AC ou RF
  - émettre et recevoir les deux types de SPDU et recevoir AA (et agir "correctement" dans ce cas)
- PR n'est utilisable que si le Transport de données express existe

### 5.3.3 Structure des SPDU

On utilise des structures T.L.V. imbriquées

**SPDU:**

SI	LI	Champ de paramètres	Données utilisateur
----	----	---------------------	---------------------

SI = 1  $\emptyset$  type de SPDU

LI = 1 ou 3  $\emptyset$  longueur du champ de paramètres

si  $l < 255 \emptyset$  longueur (hors SI et LI)

si  $254 < l < 65536$  premier octet = 255

2ème,3ème = longueur réelle

Champ de paramètres: ensemble d'unités de

- PGI : groupe de paramètres
- PI : paramètres

## PGI:

PGI	LI	Champ de paramètre(s)
-----	----	-----------------------

PGI = 1  $\emptyset$  identificateur de groupe  
 LI = 1 ou 3  $\emptyset$  longueur du champ de paramètre(s)  
 champ : une valeur de paramètre (- PI )  
       une ou plusieurs unité de PI

## PI:

PI	LI	valeur du paramètre si non vide
----	----	---------------------------------

PI = 1  $\emptyset$  identificateur de paramètre  
 LI = 1 ou 3  $\emptyset$  longueur de la valeur paramètre  
 valeur = si LI = 0 le champ valeur est vide

## exemples:

SI	LI
----	----

LI=0

pas de paramètre

SI	LI	PI	LI	PV
----	----	----	----	----

LI=3 =1

paramètre de 1 octet

SI	LI	champ d'information de l'utilisateur
----	----	--------------------------------------

SPDU données le plus simple

SI	LI	PGI	LI	PV
----	----	-----	----	----

PGI sans PI

SI	LI	PI	LI	PV	PI	LI	PV
----	----	----	----	----	----	----	----

Li=8 =1 =3

2 paramètres

SI	LI	PGI	LI	PI	LI	PV	PI	LI	PV
----	----	-----	----	----	----	----	----	----	----

LI=10 =8 =1 =3

PGI de 2 paramètres

## Concaténation des SPDU sur un TSDU

Une TSDU peut contenir de 1 à 4 SPDU concaténées. Les SPDU sont réparties en **3 catégories** permettant de construire **6 structures par concaténation**.

En concaténation de base on a une seule SPDU de catégorie 2

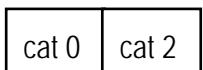


SPDU catégorie 0 isolée



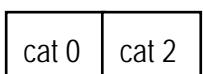
SPDU non concaténable, toujours isolée

### **Les SPDUs de catégorie 2 sont toujours concaténées**

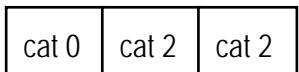


Concaténation de base

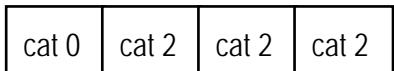
Rq: SPDUs de catégorie 2 traitée avant SPDUs de catégorie 0



Concaténation étendue



Concaténation étendue



Concaténation étendue

Rq: Les SPDUs de catégorie 2 sont traitées avec des priorités liées à leur type indiquées ci-dessous

### **Répartition:**

Catégorie 0	GT, PT (jeton) / en-tête de catégorie 2
Catégorie 1	CN, AC, RF, FN, DN, AB, AA (connexion-déconnexion) NF / GTC, GTA / PR / EX / TD (données express et typées)
Catégorie 2	DT (données normales) / CD, CDA (informations de capacité) MIP, MIA / MAP, MAA / RS, RA (synchronisation, resynchro) AS, AR, AI, AIA, AD, ADA, AE, AEA (gestion d'activité) ER, ED (signalisation d'anomalies)

### **Priorité des SPDUs:**

- Les SPDUs de catégorie 2 sont traitées avant les SPDUs de catégorie 0 (en-tête /demande ou don de jeton)
- Les SPDUs de catégorie 2 concaténés sont traités dans l'ordre suivant:
  - 1: SPDUs lancement ou reprise d'activité  
Rq: Ces SPDUs sont **toujours placées en tête**
  - 2: SPDUs données  
Rq: Ces SPDUs sont **toujours placées en queue**
  - 3: SPDUs pose de point de synchro majeure ou mineure, acquittement de point de synchro, terminaison d'activité, acquittement de terminaison d'activité

### Taille des unités de données, segmentation :

- La taille des SSDU et SPDU peut être quelconque
- La taille des TSDU est négociée entre entités de Session.  
Si valeur 0 → taille illimitée

*segmentation:*

- Si la taille est illimitée (caractère de fin) **pas de segmentation**
- Sinon ,la segmentation est **négociée** avec la taille de TSDU.
  - si non adoptée une SSDU ↔ une seul SPDU
  - si adopté **pour le sens de transmission considérée:**

Une SSDU données normales ou typées est réparti dans une ou plusieurs SPDU

### 5.3.4 Automate

Le nombre élevé d'événements entrants, d'états, de prédictats rend l'automate d'une entité de Session complexe . La structuration en unités fonctionnelles favorise son découpage en 9 sous-automates.

- événements entrants:
  - 31 issus de l'interface haut (6/OSI,utilisateur)
  - 3 + 34 SPDU issus de la couche Transport
  - 1 Horloge chien de garde
- 29 états
- 72 prédictats
- événements sortants
  - 30 vers interface haut (6/OSI,utilisateur)
  - 3 + 34 SPDU vers la couche Transport
  - 31 actions spécifiques

Sous-automates:

- connexion
- transfert de données
- synchronisation
- resynchronisation
- interruption et rupture d'activité
- lancement ou reprise d'activité et échange de données de capacité
- gestion des jetons et signalisation d'anomalies
- deconnexion
- coupure (abandon)

Une implantation réelle ne comporte qu'une partie des sous-ensembles standards ou un sous-ensemble spécifique et ne comprend donc qu'une partie de ces sous-automates

Par exemple MAP 2.1 n'ayant que l'unité fonctionnelle noyau n'utilise que les sous-automates connexion, déconnexion, coupure et transfert de données.



---

## 6 Couche 6/OSI : Service PRESENTATION

### 6.1 Rôle

Permettre à des applications de communiquer en échangeant des données **structurées** dans le cadre d'un dialogue ordonné.

Son rôle est identique à celui du langage dans la communication entre deux personnes: pour se comprendre elles doivent utiliser la même langue c'est à dire une grammaire et un vocabulaire communs.

Ce vocabulaire est constitué par les objets que la couche Présentation va manipuler.

La grammaire correspond aux fonctions s'appliquant aux objets définis par le vocabulaire.

**Dans le cadre de l'architecture OSI, la présentation s'intéresse à la syntaxe des données échangées.**

L'application, elle, se charge de la sémantique des données.

La difficulté de définir **des structures de données universelles** ou au moins couvrant un grand nombre d'applications à conduit à créer une **syntaxe de Transfert : X409** (Syntaxe abstraite) permettant de décrire n'importe quel type de données aussi bien au niveau Présentation qu'au niveau Application.

Une phase de **négociation** permet aux deux entités communiquantes de choisir la syntaxe qui sera utilisée par la suite. **La représentation en ligne du choix de cette syntaxe et du transfert des données conformément à cette syntaxe constitue le protocole de présentation.**

### 6.2 Service requis.

Pour créer un dialogue ordonné la couche Présentation utilise le **service Session**. Dans ce cadre elle fait transiter des commandes issues de l'applications en traitant les paramètres qui sont de son ressort.

## 6.3 Service fourni.

Il est décrit dans le projet de norme ISO/DIS 8822 :

Le transfert d'unités de données de protocole Application (APDU) entre entités d'Application. Pour coder les APDU les entités d'Application utilisent une **yntaxe abstraite** commune. Les données à transférer sont passées à la couche Présentation dans les paramètres de données utilisateur des primitives de Présentation.

La transformation des données codées selon une syntaxe abstraite depuis ou vers une **yntaxe de Transfert**, acceptable mutuellement par les entités de Présentation, qui assure l'intégrité des données à transmettre.

L'encryptage et la compression de données sont caractéristiques de syntaxes de transfert particulières.

La négociation d'une **yntaxe de transfert** utilisable. De cette négociation résulte l'association d'une syntaxe abstraite et d'une syntaxe de transfert compatible ; cette association constitue un **contexte de présentation**. Vu du service utilisateur de la Présentation (Application) ce contexte de présentation représente un usage spécifique d'une syntaxe abstraite.

## 6.4 Eléments du service Présentation.

### 6.4.1 Contexte de présentation.

#### 6.4.1.1 Définition d'un contexte de présentation.

Le service de Présentation fournit des facilités pour la **définition d'un contexte de présentation** adapté aux besoins du transfert d'information. Parfois il est nécessaire de créer plusieurs contextes de présentation pour satisfaire à tous les besoins d'une connexion.

Pour définir ces contextes on dispose de deux services élémentaires :

P-CONNECT

P-ALTER-CONTEXT

Ce second service permet aussi de supprimer les contextes devenus inutiles.

Tout contexte de présentation nouvellement créé est ajouté à l'ensemble des contextes disponibles pour utilisation immédiate par n'importe quelle connexion.

Lorsque l'ensemble des contextes définis est vide, il doit être possible de transférer les données utilisateur. Pour cela on utilise un contexte par défaut toujours présent. Ce contexte par défaut est aussi requis pour le transfert de données express dans le service.

P-EXPEDITED-DATA.

#### **6.4.1.2     Gestion de l'ensemble des contextes de présentation.**

Ce service est fourni en option par le protocole de présentation ISO/DIS8822. Il met en oeuvre deux unités fonctionnelles :

Gestion de contexte et Restauration de contexte.

L'unité fonctionnelle Gestion de contexte est mise en oeuvre à la réception d'une primitive P-ALTER-CONTEXT. Pour être sûr que les contextes des deux entités paires sont identiques, ce service est confirmé.

Des précautions doivent être prises lors d'une modification d'un contexte en cas de Resynchronisation, d'Interruption ou de Reprise d'Activité. En particulier, si la gestion d'Activité est utilisée on doit posséder le Jeton " Synchro Majeure/Activité" pour exécuter la primitive P-ALTER-CONTEXT.

L'unité fonctionnelle Restauration du contexte permet de rétablir un contexte en cours d'utilisation à un Point de Synchronisation en cas de Resynchronisation ou de Reprise d'Activité. Les références des différents contextes utilisés successivement doivent donc être mémorisées. En cas de fin ou d'abandon d'Activité, l'ensemble des contextes en cours est abandonné et un ensemble de contextes hors activité est rétabli. Sur une reprise d'activité l'ensemble des contextes lié à l'activité est rétabli.

### **6.4.2     Services élémentaires et primitives**

#### **6.4.2.1    Etablissement de connexion**

Ce service permet :

de sélectionner les unités fonctionnelles à utiliser  
d'établir un ensemble de contextes initial

de définir les caractéristiques du Service Session à utiliser  
d'établir la syntaxe abstraite utilisée dans le contexte par défaut.

#### **6.4.2.2 Terminaison de connexion**

Terminaison ordonnée et **non destructive** de la connexion de présentation

ou

Terminaison destructive de la connexion

#### **6.4.2.3 Gestion de contexte**

Définition des contextes et identification locale.

Suppression des contextes

#### **6.4.2.4 Transfert d'information**

Données normales gérées par jeton

Données normales sans contrôle

Information de capacité

Données express

#### **6.4.2.5 Gestion du dialogue**

Cette facilité permet de mettre en oeuvre les fonctions de Gestion des jetons, de Synchronisation, de Resynchronisation et de gestion d'Activité. Elle est en correspondance directe avec le service Session et se contente de lui retransmettre les requêtes de l'Application en leur ajoutant les paramètres qu'elle détient.

#### **6.4.2.6 Primitives**

Etablissement

P-CONNECT confirmée

Terminaison

P-RELEASE confirmée

P-U-ABORT non confirmée utilisateur

P-P-ABORT non confirmée fournisseur

Gestion de contexte

P-ALTER-CONTEXT confirmée

Transfert de données

P-TYPED-DATA non confirmée pas de jeton

P-DATA non confirmée soumis à jeton

P-EXPEDITED-DATA non confirmée

P-CAPABILITY-DATA confirmée

Gestion du dialogue

P-U-EXCEPTION-REPORT non confirmée

P-P-EXCEPTION-REPORT	initiée par fournisseur
P-TOKEN-GIVE	non confirmée
P-TOKEN-PLEASE	non confirmée
P-CONTROL-GIVE	non confirmée
P-SYNC-MINOR	confirmation optionnelle
P-SYNC-MAJOR	confirmée
P-RESYNCHRONISE	confirmée
P-ACTIVITY-START	non confirmée
P-ACTIVITY-RESUME	non confirmée
P-ACTIVITY-END	confirmée
P-ACTIVITY-INTERRUPT	confirmée
P-ACTIVITY-DISCARD	confirmée

A titre indicatif les paramètres de la primitive P-CONNECT-request sont indiqués ci-dessous :

Adresse présentation appelante

Adresse présentation appelée

Contextes multiples

Si paramètre absent un seul contexte dans l'ensemble des contextes.

Liste de définition du contexte de Présentation

pour chaque contexte : identificateur et nom de la syntaxe abstraite correspondante.

Nom du contexte par défaut

Qualité de service

Besoins de Présentation

Besoins de Session

Numéro de série initial de point de synchronisation

Assignation initiale des jetons

Identificateur de connexion de Session

Données utilisateur

#### 6.4.3 Introduction à la compression de données

La compréssion de données est une fonction qui est du ressort de la couche présentation. Toutefois, les quelques éléments ci-dessous ne correspondent pas à un projet de norme OSI mais fournissent quelques idées sur le sujet.

La compréssion de données permet de réduire la quantité de données utilisateur à transmettre **sans diminuer la quantité d'information du message**.

On distingue trois niveaux de compression de données.

- **Compression de premier niveau :**

Elle consiste à éliminer du message les caractères non significatifs par exemple des caractères "espace" en fin d'articles de taille fixe

- **Compression de second niveau :**

Elle consiste à coder de manière spécifique et condensée les séquences de caractères identiques, par exemple chaines de \* ou de - ou d'espaces. Dans un texte codé en alphabet numéro 5, on peut utiliser un caractère spécial indiquant la répétition suivi du facteur de répétition et du caractère répété.

Cette compression peut aussi porter sur des chaines de bits comportant beaucoup plus de "0" que de "1" (ou l'inverse). On peut alors ne transmettre que le nombre de "0" séparant deux "1" consécutifs avec une numération adéquate..

- **Compression de troisième niveau :**

Elle utilise les propriétés statistiques du message à transmettre et un codage à code de longueur variable adapté à la nature de l'information. Cette compression est fondée sur la redondance des messages qu'ils portent un message "littéraire" ou des données physiques (par exemple un électrocardiogramme).

La fréquence des symboles est mesurée à l'avance et un code optimal calculé pour représenter le message. Ce code ou son identificateur est transmis au collecteur de données.

En transmission, étant donné une probabilité d'erreur de transmission parfois non négligeable on peut utiliser des variantes des codes d'Huffman , par exemple des codes d'Huffman-Shannon-Fano autosynchronisants permettant de retrouver un décodage cohérent même après une erreur de transmission résiduelle.

#### **6.4.4 Introduction à la cryptographie.**

La protection des données peut se faire à différents niveaux; la protection la plus efficace se fait de bout en bout et s'effectue au niveau Présentation. Les notions ci-dessous ne sont pas liées aux projets de normes OSI en ce domaine.

##### **6.4.4.1 Menaces . Types de protection**

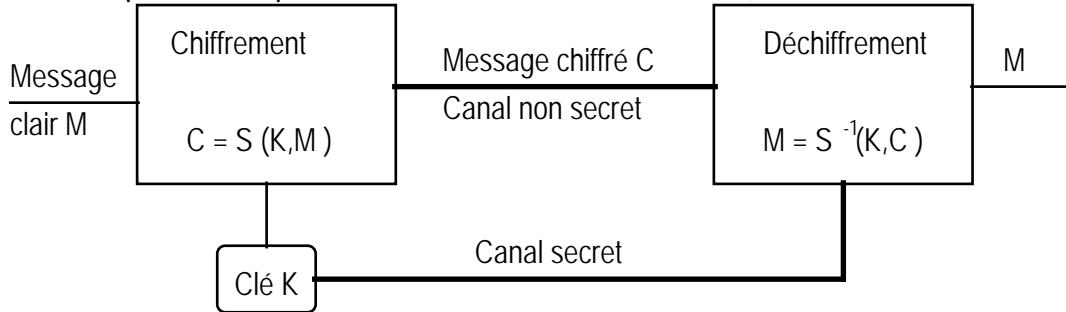
On peut classifier les menaces selon : l'endroit menacé, noeud ou maille du réseau ou le type d'intrusion : passive ou active.

Les intrusions passives consistent à se brancher en parallèle sur la chaîne de transmission et à enregistrer les données qui circulent.

Lors d'une intrusion active, l'intrus non seulement écoute les communications, mais tente aussi de modifier les échanges en créant, modifiant, supprimant ou retardant tout ou partie des messages.

Les mailles d'un réseau semblent plus fragiles que les noeuds. Cependant les intrusions à un noeud d'un réseau peuvent être beaucoup plus graves .

La protection peut donc porter sur les liaisons de données, elle se fait alors au niveau



physique, ou de bout en bout pour protéger toute la chaîne de communication.

Pour se protéger contre les intrusions passives un **chiffrement** des données est suffisant. Contre les intrusions actives il est nécessaire de contrôler l'accès aux données ou la source des messages transmis. On procéde alors à une **authentification de signature**.

**remarque :** Le chiffrement d'un message consiste à transformer un message clair en un message chiffré (secret) à l'aide d'un code. La récupération de ce message à partir du code et du message secret est faite par une opération de déchiffrement. Le décryptage du message secret permet de retrouver le message clair à partir du seul message secret (sans connaître le code).

#### 6.4.4.2 Méthodes de cryptages

Un système de cryptage peut être sûr à cause du coût ou de la durée des calculs nécessaires pour le décoder ou inconditionnellement sûr s'il peut résister à toute analyse du message crypté, même si l'on connaît le message clair correspondant.

Le chiffrement peut se faire au vol, sur des blocs de taille indéfinie ou au moins très long, bloc par bloc. La taille des blocs doit être suffisante pour interdire un décryptage en un temps raisonnable par simple analyse de toutes les combinaisons. On travaillera souvent sur des blocs de 64 bits.

##### Cryptosystèmes à clé unique (privée)

Dans ces systèmes les messages sont chiffrés avant d'être transmis sur un canal non secret et la clé de chiffrement est transmise sur un canal secret.

La fonction de chiffrement  $S$  est inversible telle que :

$$S^{-1}(K, C) = S^{-1}(S(K, M)) = M$$

La fragilité d'un tel système réside dans le partage de la clé entre source et collecteur.

Les méthodes classiques utilisent généralement une combinaison de cryptage par transposition et de cryptage par substitution . Dans le premier cas les symboles sont permutés et souvent regroupés en mots de taille fixe. Dans le second les symboles sont remplacés par d'autres symboles selon la clé de codage ( cf. code de César ...). Ces méthodes sont adaptées à un codage manuel ou par automates rigides électroniques ou électromécaniques.

Exemples :

Le système *DES* : *Data encryption standard* développé par IBM et adopté par l'administration américaine suit ces principes.

Le chiffrement porte sur des blocs d'information de 64 bits et utilise une clé à 56 bits (plus 8 bits de contrôle) . Entre une transposition initiale et une transposition finale on réalise 16 itérations d'une fonction mélant transposition et substitution selon l'algorithme suivant : si  $L_i$  et  $R_i$  désignent les demi-blocs gauche et droit à l'itération  $i$ , on calcule

$$\begin{aligned}L_i &= R_{i-1} \\R_i &= L_{i-1} + f(R_{i-1}, K_i) \text{ modulo } 2\end{aligned}$$

où  $K_i = KS(i, \text{clé})$   
avec  $KS$  : fonction "Key Schedule"

Toute la puissance du codage réside en fait dans cette **fonction non-linéaire** qui est réalisée par une table.

Pour cela le champ  $R_{i-1}$  est étendu de 32 à 48 bits par duplication de la moitié de ses bits. La clé  $K_i$  est ajoutée, puis le champ obtenu est réduit grâce à une table (publique) qui fait correspondre des champs de 4 bits aux 8 champs de 6 bits obtenus ci-dessus. Cette table est très facile à cabler.

Ce système travaille avec des blocs de 64 bits mais en fait sur des demi-mots. Il est simple à mettre en oeuvre sur des ordinateurs ayant des entiers codés sur 32 bits mais peu rapide.

Cet algorithme est donc beaucoup plus facile à implanter dans un circuit intégré qu'à programmer. Ces circuits existent mais sont soumis à des restrictions de diffusion (interdiction d'exportation, autorisation administrative pour l'achat auprès des fabricants nationaux).

Le déchiffrement utilise le même algorithme.

Ce système présente  $2^{56} \approx 7,2 \cdot 10^{16}$  clés possibles. Sa distance d'unicité n'est que de 17,5. Théoriquement pour casser ce code il suffit de 18 caractères du texte clair et crypté mais un temps de calcul très long.

Les **seuls codes réellement indéchiffrables** sont les codes de VERNAM qui consistent à ajouter un séquence aléatoire au texte clair.

Si cette **clé aléatoire est utilisée une seule fois** le code est impossible à traduire. La clé est retrouvée immédiatement (par un simple ou exclusif) si on possède simultanément les tests clair et chiffré. Ceci implique donc que la clé soit à usage unique et renouvelée pour chaque chiffrement.

On peut aussi utiliser des **codes cycliques** avec des algorithmes de multiplication et de division. Un ou plusieurs polynomes générateurs constituent la clé de chiffrement. Un polynome de degré élevé permet de travailler sur des blocs très longs. L'encryptage par multiplication de polynome ajoute N bits au message clair; l'encryptage par division supprime 16 bits. On pourra donc garder sa taille au message en appliquant successivement les deux algorithmes. On réalise ainsi une substitution sur des blocs très très longs ( de plusieurs milliers de bits). Des transpositions peuvent être ajoutées avant ou après les opérations de multiplication ou de division.

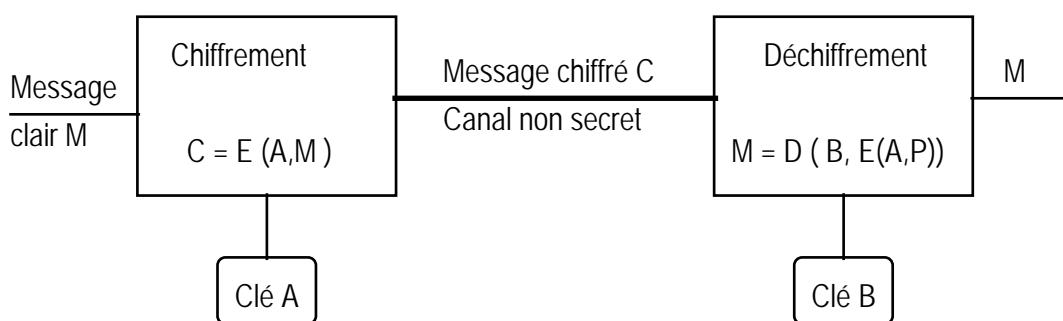
### Cryptosystèmes à clé publique

Pour éviter les dangers introduits par le partage des clés de cryptage on utilise deux clés différentes pour le chiffrement et le déchiffrement. La clé de chiffrement est publique; la clé de déchiffrement reste secrète et n'appartient qu'au collecteur du message.

Le principe de ce système est que, d'une part un intrus est incapable de retrouver un message clair même s'il connaît la clé publique et le message codé, et que d'autre part, la source du message est incapable de retrouver la clé de déchiffrement même connaissant la clé publique, le message clair et le message codé (décryptage en un temps excessivement long, par balayage systématique par exemple). Seul le détenteur de la clé privée peut restituer toutes les composantes du système.

Pour un sens de transfert, on part d'une clé K et d'un couple d'algorithmes  $E(K,M)$  et  $D(K,C)$ . E est l'inverse de D . Les conditions suivantes doivent être satisfaites :

- pour tout K , il est possible de trouver un couple d'inverses  $E(K)$  et  $D(K)$  à partir de K. E et D sont (facilement) calculables
- il est impossible de trouver D à partir de E à cause de la durée ou du coût des calculs



A et B sont déterminés par le collecteur à partir d'une clé de base K.

L'algorithme E et la clé A sont publics. Ils sont fournis par le collecteur du message à tous ses correspondants. Ils sont calculés en fonction de D et de la clé secrète K.

### **Exemple 1 : système Rivest - Shamir - Adleman (RSA)**

Ce système est fondé sur la décomposition en facteurs premiers de grands nombres. Sa puissance tient au fait que pour déterminer la clé secrète D, un intrus doit pouvoir factoriser un nombre n supérieur à  $10^{200}$ . D'après Rivest cette tâche nécessiterait um million d'années avec le meilleur algorithme connu sur une machine traitant un million d'instructions par seconde.

Pour établir les clés publique et secrète on part d'un nombre aléatoire E pris dans un intervalle défini ci-dessous et de deux (ou plusieurs) nombres **premiers** distincts p et q très grands ( $> 10^{100}$ ).

La clé publique est constituée du couple (E, n) avec  $n = \prod_j p_j$

Soit  $\Phi(n)$  l'indicateur d'Euler de n .

$\Phi(n)$  est le nombre d'entiers premiers avec n qui n'entrent pas dans sa factorisation .

Par exemple

$$14 = 2 * 7 \quad \Phi(14) = 6 \text{ (à savoir } 1, 3, 5, 9, 11, 13\text{)}$$

Si n n'a que des facteurs  $p_j \quad \Phi(n) = \prod_j p_j - 1$

Ainsi  $10 = 2 * 5$  et  $\Phi(10) = 1 * 4 = 4$

$$9 = 1 \text{ modulo } 4$$

$$2^9 \text{ modulo } 10 = 2^1 \text{ modulo } 10 = 2$$

comme on peut le vérifier :

$$2^9 \text{ modulo } 10 = 512 \text{ modulo } 10 = 2$$

L'**encryptage** utilise la clé E, nombre aléatoire compris entre 3 et  $\Phi(n)$  . Il est réalisé de la manière suivante :

Le message clair est codé en une séquence de chiffres (non nécessairement binaires) et cette séquence segmentée en blocs  $B_i$  tels que

$$0 \leq B_i < n .$$

Pour chaque bloc  $B_i$  on réalise la substitution :

$$C_i = B_i^E \text{ modulo } n .$$

$C_i$  est transmis sur le canal non secret.

Pour le **déchiffrement** on détermine la clé secrète D telle que :

$$E * D \text{ modulo } \Phi(n) = 1 .$$

Le calcul de D est identique à celui de  $z=w^{-10}$  dans le système précédent.

Remarque : E et n sont publics et servent à calculer la clé secrète D par l'intermédiaire de  $\Phi(n)$ . Il convient donc de choisir n suffisamment complexe pour qu'il ne soit pas possible de trouver  $\Phi(n)$  et en déduire D.

Le message clair est donné par le calcul de

$$B_i = C_i^D \text{ modulo } n$$

Justification :

$$C_i = B_i^E \text{ modulo } n$$

$$C_i^D = B_i^{ED} \text{ modulo } n = B_i^1 \text{ modulo } n = B_i$$

### Applications :

L'exemple ci-dessous est donné à titre indicatif et ne correspond pas à un choix de clés judicieux. En effet le terme n est le produit de facteurs trop petits; ceci permet de calculer (trop) simplement la clé de décryptage connaissant E et n.

Calcul des clés :

$$p = 31 \quad q = 47 \quad \text{soit } n = 1457$$

$$\Phi(n) = 30 * 46 = 1380$$

clé publique :

$$E = 889 \in [3, 1380]$$

clé secrète :

$$D = E^{-1} \text{ modulo } 1380 = 1009$$

$$\text{soit } 889 * 1009 \text{ modulo } 1380 = 1$$

bloc numérique à coder :

$$B = 234$$

chiffrement :

$$C = B^E \text{ modulo } n = 234^{889} \text{ modulo } 1457 = 892$$

déchiffrement :

$$B = C^D \text{ modulo } n = 892^{1009} \text{ modulo } 1457 = 234$$

autre exemple :

si  $p_j = 2, 3, 11, 17$

$$\begin{aligned}n &= 2 * 3 * 11 * 17 = 1122 \\ \Phi(n) &= 1 * 2 * 10 * 16 = 320\end{aligned}$$

si  $E = 47$

$$D = E^{-1} \text{ modulo } \Phi(n) = 143$$

alors  $234^{47} \text{ modulo } 1122 = 174$   
et  $174^{143} \text{ modulo } 1122 = 234$

### **Exemple 2 : système Merkle-Diffie-Hellman**

Le message clair est segmenté en blocs de taille  $n$  correspondant à la longueur des clés A ou B.  $n$  doit être assez grand pour que connaissant uniquement la clé publique, on ne puisse retrouver le message original à partir du message crypté qu'en un temps très très long.

Soit  $B = [b_1, b_2, \dots, b_n]$  la clé privée du destinataire  
 $A = [a_1, a_2, \dots, a_n]$  la clé publique de la source  
 $M = [x_1, x_2, \dots, x_n]$  un bloc du message clair

Les coefficients  $a_i$  et  $b_i$  sont des entiers naturels et les variables  $x_i$  les bits du message clair

Les coefficients  $b_i$  sont choisis de manière aléatoire avec la condition ci-dessous nécessaire pour déchiffrer simplement C. On impose aux coefficients  $b_i$  la propriété:

$$b_i > \sum_{j=0}^{i-1} b_j$$

La clé de chiffrement A, fournie aux correspondants, est calculée à partir de deux entiers naturels  $w$  et  $m$  premiers entre eux et de B.  $w$  et  $m$  sont secrets .

On a  $a_i = b_i * w \text{ modulo } m$

Pour déchiffrer le message C on doit aussi calculer une "gâche"  $z = w^{-1}$  telle que  $z * w \text{ modulo } m = 1$ .

Le message crypté est  $C = AM = a_1x_1 + a_2x_2 + \dots + a_nx_n$   
C'est un entier naturel.

Au déchiffrement on calcule :

$Bx = C * z \text{ modulo } m = AM * z \text{ modulo } m$   
puis on utilise l'algorithme d'empilement suivant :  
pour  $i = n$  jusqu'à 1

$$\text{si } b_i > C - \sum_{j=i+1}^n x_j b_j \quad \text{alors } x_i = 0 \quad \text{sinon } x_i = 1.$$

Rq : la somme est supposée nulle si  $i = n$

Justification :

$$\begin{aligned} Ax &= \sum_{i=1}^n a_i x_i \\ Axz \text{ modulo } m &= \sum_{i=1}^n a_i x_i z \text{ modulo } m \\ &= \sum_{i=1}^n (a_i z) x_i \text{ modulo } m \\ &= \sum_{i=1}^n b_i x_i \\ &= Bx \end{aligned}$$

Il suffit donc de calculer  $Axz$  pour retomber sur un algorithme simple de décodage.

### Application simple :

On suppose que le message est décomposé en blocs de 8 bits (beaucoup trop courts en réalité ...)

Clé de déchiffrement secrète :

$$w = 889 \quad m = 1457 \quad B = [3, 7, 12, 23, 47, 95, 189, 377]$$

Calcul de la "gâche" secrète :

$$z = 1398 \text{ tel que } 1 = 889 * 1398 \text{ modulo } 1457$$

Calcul de la clé publique :

$$a_i = 889 * b_i$$

$$\text{soit } A = [1210, 395, 469, 49, 987, 1406, 466, 43]$$

Premier bloc du message à coder :

"M" soit 01001101

Premier bloc du message chiffré :

$$C = 395 + 987 + 1406 + 43 = 2831$$

Bloc déchiffré :

$$Bx = 2831 * 1398 \text{ modulo } 1457 = 526$$

Décodage par empilement

soit  $x_i = 0$  si  $b_i > B - \sum_{j=i+1}^n x_j b_j$

377 < 526	$x_i = 1$
189 > 526 - 377 = 149	= 0
95 < 149	= 1
47 > 149 - 95 = 56	= 1
23 > 56 - 47 = 7	= 0
12 > 7	= 0
7 ≤ 7	= 1
3 > 7 - 7 = 0	= 0

On retrouve bien le bloc original

---

## 7 Couche 6-7/OSI : SYNTAXE ABSTRAITE ASN.1

La syntaxe ASN.1 a été définie dans le cadre de la couche **7/OSI** pour fournir une représentation commune standard aux PDU spécifiques des diverses applications . Dans ce cadre elle est utilisée comme syntaxe abstraite .

Elle peut aussi être utilisée dans le cadre de la couche **6/OSI** (Présentation) pour donner une description commune des informations à transmettre tenant compte de différents types de terminaux ( alphabet n°5 , vidéotex, télétex, facsimilé,etc.) ou des structures de fichiers . Elle est alors utilisée comme syntaxe de transfert.

L'OSI a repris et légèrement étendu sous le nom de syntaxe ASN.1: Abstract Syntax Numero 1, standards OSI IS/8824 et 8825, la Recommendation X409 du CCITT, créée en 1984 pour coder les PDU de l'Application de messagerie interpersonnelle. Dans le livre bleu (1990) cette recommandation est devenue X208 et X209 .... L'ancienne version doit être désignée par X409-84.

### 7.1 Structure des PDU

Toutes les PDU sont codées sous forme de **structures d'items imbriqués** , le champ valeur de chaque item pouvant être lui-même un item. Tous ces **items** ont une structure de type **T.L.V.: Type,Longueur,Valeur**.

T: Identificateur codé sur 1 ou plusieurs octets

L: Longueur codée sur 1 ou de 2 à 127 octets

V:Champ valeur pouvant atteindre une longueur de 2 puissance 1008 octets.

Ce champ peut contenir uniquement des données utilisateur ou être un autre item.

### 7.1.1 Identificateur T :

Champ	Nature	Code
bits 8-7	classe universel application contexte privé	00 01 10 11
bit 6	forme <b>primitif</b> <b>constructeur</b>	0 1
bits 5 à 1	code valeur extension	0 à 11110 11111
octets suivants	si extension valeur extension	0 à 254 255

Dans la classe "universel" la syntaxe ASN.1 fournit des types de base "**prédéfinis**" et des types construits "**définis**" souvent rencontrés. La liste ci-dessous, créée pour les applications de messagerie interpersonnels n'est pas limitative et sera complétée au fur et à mesure de la spécification de nouvelles applications.

Le type "étiqueté" sert à créer des types spécifiques à une application ou une partie d'application (contexte).

Tous les codes sont donnés en hexadécimal. Les types "chaîne" peuvent être **primitifs** (codes ci-dessous) ou **constructeurs** (bit 6 à 1; ex: 03 -> 23). Un type constructeur correspond à un type dont le champ "valeur" contient un ou plusieurs autres types. Les types séquence, ensemble, choix sont constructeurs

(exemple : séquence -> code ID = 10h , code réel =30h)

**types prédefinis:**

code	types prédefinis	commentaires
00		Avec longueur 0 fin de contenu
01	booléen	vrai (FF) ou faux (00)
02	entier	Codé binaire en complément à 2
03	chaîne binaire	
04	chaîne d'octets	
05	vide	longueur 00
06	identificateur d'objet	
07	descripteur d'objet	
08	externe	
09	réel	
10	énuméré	<b>type créé :</b> explicite ou <b>implicite</b> code : valeur fixée à la création
	étiqueté	parmi plusieurs possibilités; lié ou non lié code : type choisi + valeur
	choix	permet de coder n'importe quelle valeur explicitée par ailleurs ....
	quelconque	

code	types définis	commentaires
16	Séquence	suite <b>ordonnée</b> d'éléments
17	Ensemble	ensemble <b>non ordonné</b> de membres
18	Chaîne numérique	
19	Chaîne imprimable	
20	Chaîne Télétex	
21	Chaîne Vidéotex	
22	Chaîne AI5 (T100)	
23	heure généralisée	date et heure légale
24	heure UTC	temps universel ; terminé par Z
25	Chaîne de caractères graphiques	
26	Chaîne de caractères visibles (T61)	Terminaux Télétex
27	Chaînes de caractères générale	

*Nota : d'autres codes de type universel peuvent encore être ajoutés.*

## 7.1.2 Longueur

Elle peut coder un champ valeur de longueur indéfinie, courte ou longue.

- indéfinie : code 80h; caractère de fin d'item : EOC= 00
- courte :- code 0 à 7Fh; champ valeur de 0 à 127 octets
- longue - code 81h à FFh
  - 1er octet = longueur de la longueur - 1 (0 à 127 octets)
  - octet suivant = longueur du champ valeur

### 7.1.3 Contenu

Le champ V peut être **primitif ou constructeur**. Dans ce cas il possède une structure d'items imbriqués.

## 7.2 Crédation des types

### 7.2.1 macro

Il est possible de définir une **notation non standard** au moyen d'une **macro**. Cette notation est spécifiée dans le corps de macro qui est précédé par "begin" et suivi de "end". Chaque nom de macro est écrit en majuscules.

Exemple (d'autres structures sont permises):

```
ATTRIBUT MACRO ::=  
BEGIN  
TYPE NOTATION ::= "LIST"<Type::= SEQUENCE OF Chaîne-  
IA5>/vide<Type::= Chaîne-IA5>  
VALUE NOTATION ::= valeur(VALUE Type)  
END
```

Utilisation:

```
destinataire ATTRIBUT LIST ::= {"NOM = J.Dupont", "AGE=42"}  
ou  
destinataire ATTRIBUT ::= "NOM = J.Dupont"
```

### 7.2.2 module

Un **module** sert à regrouper des définitions apparentées de types, valeurs et macros ,par exemple, celles d'une spécification particulière de protocole. Un corps de module est aussi encadré par "BEGIN-END".

Chaque nom de module commence par une majuscule.

Exemple:

```
Couleur DEFINITION ::=  
BEGIN  
Couleur-primaire ::= INTEGER{rouge(0),jaune(1),bleu(2)}  
Couleur-Par-Défaut Couleur-Primaire ::= jaune  
END
```

## 7.3 Exemples :

### 7.3.1 Séquence, ensemble, choix, étiqueté, quelconque

Justificatifs ::= SEQUENCE {

nom-utilisateur Chaîne-IA5,  
mot-Passe Chaîne-IA5,  
numéro-Compte INTEGER }

Attributs-Fichier ::= SET { (ensemble)  
propriétaire [0] IMPLICIT Nom-Utilisateur,  
taille-Contenu-En-Octets [1] IMPLICIT INTEGER,  
[2] IMPLICIT Contrôle\_Accès.}

Identificateur-Fichier ::= CHOICE {  
nom\_Relatif [0] IMPLICIT Chaîne-IA5,  
nom\_Absolu [1] IMPLICIT Chaîne-IA5,  
numéro\_Série [2] IMPLICIT INTEGER }

Nom\_Fichier ::= [APPLICATION 8] IMPLICIT SEQUENCE {  
nom\_Répertoire Chaîne-IA5,  
nom\_Fichier Chaîne-IA5 }

Contenu\_Fichier ::= ANY (quelconque)

### 7.3.2 "Message"

L'exemple ci-dessous, tiré de la Recommandation ASN.1, illustre l'utilisation de cette syntaxe. Il donne successivement une description informelle de l'enregistrement à représenter, une description formelle de sa structure puis de la valeur de cet enregistrement, enfin une représentation "commentée" de la valeur de cet enregistrement après codage dans la syntaxe ASN.1.

#### Exercice: Codage ASN.1 d'une structure d'accès

Dans les spécifications FTAM est défini un champ "Context" de type CONTEXT-SPECIFIC. Le troisième champ défini de ce type (valeur 2) est décrit par:

identificateur A2 ([CONTEXT-SPECIFIC 2])

longueur 03

contenu

    identificateur 02 ([UNIVERSAL 2] -> INTEGER)

    longueur 01

    contenu 06

On veut coder le type "AccesStructureType" de valeur

```
{ structureType      hierarchical,  
    maxDepth         16 }
```

par une structure universelle "ensemble" (SET champ code 17 ou 11H) contenant deux champs "Context", pour structureType et maxDepth, dont les valeurs de type sont 0 et 1. Hierarchical sera codé par un entier de valeur 2.

Donner la description ASN.1 de cette partie de FPDU et la chaîne d'octets effectivement transmise (codée en hexadécimal)

**Réponse :**

les TLV pour structureType et maxDepth sont de type constructeur "Context"

maxDepth sera codé sur un TLV de type INTEGER de valeur 16

le paramètre hierarchical sera aussi codé par un INTEGER de valeur 2.

On place dans un SET deux champs CONTEXT-SPECIFIC construits sur le modèle ci-dessus, soit :

identificateur 31 ([UNIVERSAL 11H -> SET]) 0 0 1 10001

longueur 0A

contenu

    identificateur A0 ([CONTEXT-SPECIFIC 0] -> StructureType)

    longueur 03

    contenu

        identificateur 02 ([UNIVERSAL 2] -> INTEGER)

        longueur 01

contenu 02 (hierarchical = 2)  
identificateur A1 ([CONTEXT-SPECIFIC 1] -> maxDepth)  
longueur 03  
contenu  
    identificateur 02 ([UNIVERSAL 2] -> INTEGER)  
    longueur 01  
    contenu 10 (16)

SOIT 31 0A A0 03 02 01 02 A1 03 02 01 10

## Exemples

L'utilisation de la notation et de la représentation standard définies dans cette Recommandation est illustrée par un exemple simple d'enregistrement «salarié» fictif.

### II.1 Description informelle de l'enregistrement «salarié»

La structure de l'enregistrement «salarié» est décrite ci-dessous, avec sa valeur pour un individu particulier.

Nom: John P. Smith  
Fonction: Director  
Matricule: 51  
Date d'engagement: 17 September 1971  
Nom du conjoint: Mary T. Smith  
Nombre d'enfants: 2

Renseignements sur enfant:  
Nom: Ralph T. Smith  
Date de naissance: 11 November 1957

Renseignements sur enfant:  
Nom: Susan B. Jones  
Date de naissance: 17 July 1959

### II.2 Description fonctionnelle de la structure de l'enregistrement

La structure de chaque enregistrement «salarié» est décrite formellement à l'aide de la notation standard des types de données.

Enregistrement-salarié ::= [APPLICATION 0] IMPLICIT SET |

    Nom,  
    fonction [0] Chaîne-IAS,  
    Matricule,  
    date-Engagement [1] Date,  
    nom-Conjoint [2] Nom,  
    [3] IMPLICIT SEQUENCE OF Renseignements-Enfant DEFAULT |||

Renseignements-Enfant ::= SET |  
    Nom,  
    date-Naissance [0] Date;

Nom ::= [APPLICATION 1] IMPLICIT SEQUENCE |  
    prénom Chaîne-IAS,  
    initiale Chaîne-IAS,  
    nom-Famille Chaîne-IAS;

Matricule ::= [APPLICATION 2] IMPLICIT INTEGER

Date ::= [APPLICATION 3] IMPLICIT Chaîne-IAS -- YYYYMMDD

### II.3 Description formelle de la valeur d'enregistrement

La valeur de l'enregistrement "salarié" relatif à John Smith est décrite formellement ci-dessous à l'aide de la notation standard des valeurs de données.

! prénom "John", initiale "P", nom-Famille "Smith",  
fonction "Director",  
51,  
date-Engagement "19710917",  
nom-Conjoint (prénom "Mary", initiale "T", nom-Famille "Smith"),  
  
    (prénom "Ralph", initiale "T", nom-Famille "Smith",  
    "19571111"),  
  
    (prénom "Suzan", initiale "B", nom-Famille "Jones",  
    "19590717");

#### II.4 Représentation de la valeur de l'enregistrement

La représentation au niveau des octets de l'enregistrement précédent est représentée ci-dessous. Les valeurs des identificateurs, longueurs et contenus des entiers sont indiquées en hexadécimal, à raison de deux chiffres hexadécimaux par octet. Les valeurs des contenus des chaînes d'octets sont représentées comme du texte, avec un caractère par octet.

Enregistrement		Longueur	Contenu			
"salarie"		3185				
60						
	Nom	Longueur	Contenu			
	61	10				
		Chaine IAS	Longueur	Contenu		
		16	04	"John"		
		Chaine IAS	Longueur	Contenu		
		16	01	"P"		
		Chaine IAS	Longueur	Contenu		
		16	05	"Smith"		
	Fonction	Longueur	Contenu			
	40	0A				
		Chaine IAS	Longueur	Contenu		
		16	08	"Director"		
	Matricule	Longueur	Contenu			
	42	01	33			
	Date	Longueur	Contenu			
	engagement	0A				
		Date	Longueur	Contenu		
		43	08	"19710917"		
	Nom	Longueur	Contenu			
	conjoint	12				
		Nom	Longueur	Contenu		
		61	10			
			Chaine IAS	Longueur	Contenu	
			16	04	"Mary"	
			Chaine IAS	Longueur	Contenu	
			16	01	"T"	
			Chaine IAS	Longueur	Contenu	
			16	05	"Smith"	
	[3]	Longueur	Contenu			
	A3	42				
		Ensemble	Longueur	Contenu		
		31	1F			
			Nom	Longueur	Contenu	
			61	11		
				Chaine IAS	Longueur	Contenu
				16	05	"Ralph"
				Chaine IAS	Longueur	Contenu
				16	01	"T"
				Chaine IAS	Longueur	Contenu
				16	05	"Smith"
		Date	Longueur	Contenu		
	naissance	0A				
			Date	Longueur	Contenu	
			43	08	"19571111"	
	Ensemble	Longueur	Contenu			
	31	1F				
			Nom	Longueur	Contenu	
			61	11		
				Chaine IAS	Longueur	Contenu
				16	05	"Suzan"
				Chaine IAS	Longueur	Contenu
				16	01	"B"
				Chaine IAS	Longueur	Contenu
				16	05	"Jones"
		Date	Longueur	Contenu		
	Naissance	0A				
			Date	Longueur	Contenu	
			43	08	"19590717"	

## ANNEXE B

(à la Recommandation X.409)

### Résumé de la notation

La notation définie dans cette Recommandation est résumée ci-dessous sous une forme condensée et réorganisée.

Définition-Type	:: = identificateur ":: = " Type
Type	:: = Type-Prédéfini   Type-Défini
Type-Prédéfini	:: = BOOLEAN   INTEGER   INTEGER (Liste-Nombres-Nommés)   BIT STRING   BIT STRING (Liste-Nombres-Nommés)   OCTET STRING   NULL   SEQUENCE   SEQUENCE OF Type   SEQUENCE (Types-Eléments)   SET   SET OF Type   SET (Types-Membres)   Etiquette IMPLICIT Type   Etiquette Type   CHOICE (Liste-Types-Possibilités)   identificateur < Type >   ANY
Type-Défini	:: = identificateur   identificateur.identificateur
Liste-Nombres-Nommés	:: = Nombre-Nommé   Liste-Nombres-Nommés, Nombre-Nommé
Nombre-Nommé	:: = identificateur (Valeur-Numérique)
Valeur-Numérique	:: = nombre   Valeur-Définie
Types-Eléments	:: = Liste-Types-Optionnels   vide
Types-Membres	:: = Liste-Types-Optionnels   vide
Liste-Types-Optionnels	:: = Type-Optionnel   Liste-Types-Optionnels, Type-Optionnel
Type-Optionnel	:: = Type-Nommé   Type-Nommé OPTIONAL   Type-Nommé DEFAULT Valeur   Composants-De
Type-Nommé	:: = identificateur Type   Type
Composants-De	:: = COMPONENTS OF Type -- Séquence ou Ensemble.
Liste-Types-Possibilités	:: = Type-Nommé   Liste-Types-Possibilités, Type-Nommé
Etiquette	:: = [Classe Valeur-Numérique]
Classe	:: = UNIVERSAL   APPLICATION   PRIVATE   vide
Définition-Valeur	:: = identificateur Type ":: = " Valeur
Valeur	:: = Valeur-Prédéfinie   Valeur-Définie
Valeur-Prédéfinie	:: = TRUE   FALSE   nombre   -nombre   identificateur ("chaîne" B   "chaîne" H   Liste-Identificateurs)   "chaine"   NULL   ,Valeurs  identificateur-Valeur   Type Valeur

Fascicule VIII.7 – Rec. X.409

Exemple: Identificateur-Fichier :: = CHOICE |

nom-Relatif [0] IMPLICIT Chaîne-IAS,  
-- nom de fichier (par exemple, «Rapport d'Activité de Mars»)  
nom-Absolu [1] IMPLICIT Chaîne-IAS,  
-- noms de fichiers et répertoire les contenant  
-- (par exemple, «Rapport d'Avancement de Mars <Williams>»)  
-- d'autres formes d'identificateurs de fichier sont réservées pour étude ultérieure -- }

CH3 Il faut affecter un nom de référence à chaque possibilité dont l'usage n'est pas parfaitement évident d'après son type.

Exemple: Identificateur-Fichier :: = CHOICE |

nom-Relatif [0] IMPLICIT Chaîne-IAS,  
-- nom de fichier (par exemple, «Rapport d'Avancement de Mars»)  
nom-Absolu[1] IMPLICIT Chaîne-IAS,  
-- noms de fichiers et répertoire les contenant  
-- (par exemple, «Rapport d'Avancement de Mars <Williams>»)  
[2] IMPLICIT Numéro-Série -- identificateur de fichier attribué par le système -- }

CH4 Lorsque l'étiquetage implicite est la règle dans une application particulière de la présente Recommandation, utiliser un choix même pour un seul type, quand la possibilité d'une autorisation ultérieure de plusieurs types est envisagée. Cela écarte la possibilité de recourir à un étiquetage implicite et facilite ainsi la transition.

Exemple: Valeux :: = [APPLICATION 12] CHOICE {Chaîne-IAS}

en prévision de

Valeux :: = [APPLICATION 12] CHOICE {Chaîne-IAS, Voice}

CH5 Utiliser un choix lié pour représenter une variable dont le type est celui d'une possibilité particulière d'un choix non lié défini auparavant.

Exemple: Nom-Fichier-Absolu :: = nom-Absolu < Identificateur-Fichier -- voir CH3

Voir également la règle TAS.

### III.10 Quelconque

AN1 Utiliser un quelconque pour représenter une variable dont le type est non spécifié, ou spécifié dans une autre Recommandation.

Exemple: Contenu-Fichier :: = ANY

-- élément de données dont le type est spécifié dans la Recommandation X.fictive

<b>Valeur-Définie</b>	:: = identificateur ; identificateur.identificateur
<b>Liste-Identificateurs</b>	:: = identificateur ; Liste-Identificateurs, identificateur
<b>Valeurs</b>	:: = Liste-Valeurs-Nommées ; vide
<b>Liste-Valeurs-Nommées</b>	:: = Valeur-Nommée ; Liste-Valeurs-Nommées, Valeur-Nommée
<b>Valeur-Nommée</b>	:: = identificateur Valeur : Valeur
<b>Définition-Macro</b>	:: = identificateur MACRO ":: = " BEGIN Corps-Macro END
<b>Corps-Macro</b>	:: = Production-Type Production-Valeur Productions-Utilisées
<b>Type-Production</b>	:: = TYPE NOTATION ":: = " Liste-Possibilités
<b>Production-Valeur</b>	:: = VALUE NOTATION ":: = " Liste-Possibilités
<b>Productions-Utilisées</b>	:: = Liste-Productions   vide
<b>Liste-Productions</b>	:: = Production   Liste-Productions Production
<b>Production</b>	:: = identificateur ":: = " Liste-Possibilités
<b>Liste-Possibilités</b>	:: = Possibilité   Liste-Possibilités "!" Possibilité
<b>Possibilité</b>	:: = Liste-Symboles
<b>Liste-Symboles</b>	:: = Symbole   Liste-Symboles Symbole
<b>Symbole</b>	:: = Terminal   Non-Terminal   Définitions-Incluses
<b>Terminal</b>	:: = "chaîne"
<b>Non-Terminal</b>	:: = identificateur   "chaîne"   "identificateur"   "nombre"   "vide"   type   type (identificateur)   valeur (Type)   valeur (identificateur Type)
<b>Définitions-Incluses</b>	:: = <Liste-Définitions-Incluses>
<b>Liste-Définitions-Incluses</b>	:: = Définition-Incluse   Liste-Définitions-Incluses Définition-Incluse
<b>Définition-Incluse</b>	:: = Définition-Type : Définition-Valeur
<b>Définition-Module</b>	:: = identificateur DEFINITIONS ":: = " BEGIN Corps-Module END
<b>Corps-Module</b>	:: = Liste-Définitions ; vide
<b>Liste-Définitions</b>	:: = Définition   Liste-Définitions Définition
<b>Définition</b>	:: = Définition-Type : Définition-Valeur   Définition-Macro

Des commentaires peuvent être inclus dans la notation. Ils sont précédés de deux tirets (« -- ») et terminés par deux tirets ou par la fin de la ligne (voir AI n° 5).

APPENDICE III  
(à la Recommandation X.409)

**Règles d'utilisation de la notation**

La souplesse des types de données et de la notation formelle définis dans cette Recommandation permet la conception d'une large gamme de protocoles. Mais cette souplesse peut parfois être source d'une certaine confusion, en particulier quand on utilise la notation pour la première fois. Le présent appendice vise à réduire cette confusion à un nouveau minimum en donnant des règles et des exemples d'utilisation de la notation. Pour chacun des dix types de données *prédéfinis*, une ou plusieurs règles d'utilisation sont proposées. Les types de données *définis* (exemple, chaîne-IAS) spécifiés dans la présente Recommandation ne sont pas traités ici.

**III.1 Booléen**

- BO1 Utiliser un Booléen pour représenter la valeur d'une variable logique (c'est-à-dire à deux états), par exemple, la réponse affirmative ou négative à une question.

Exemple: **Salarie :: = BOOLEAN**

- BO2 Pour affecter un nom de référence à un Booléen, en choisir un qui décrive l'état *vrai*.

Exemple: **Marié :: = BOOLEAN**

*et non*

**Statut-Marital :: = BOOLEAN**

Voir également la règle IN4.

**III.2 Entier**

- IN1 Utiliser un entier pour représenter la valeur d'une variable cardinale ou entière – de grandeur illimitée à toutes fins pratiques.

Exemple: **Solde-Compte-Bancaire :: = INTEGER** – – *en cents; négatif signifie découvert*

- IN2 Définit, comme valeurs remarquables, les valeurs minimale et maximale autorisées d'un entier.

Exemple: **Quantième-Mois :: = INTEGER {premier(1), dernier(31)}**

- IN3 Utiliser un entier, avec des valeurs remarquables, pour représenter la valeur d'une variable pouvant prendre trois états ou plus. Affecter les valeurs en commençant par zéro si leur seule contrainte est d'être distinctes.

Exemple: **Jour-Semaine :: = INTEGER {lundi(0), mardi(1), mercredi(2), jeudi(3), vendredi(4), samedi(5), dimanche(6)}**

- IN4 Utiliser un entier, avec des valeurs remarquables, pour représenter la valeur d'une variable qui peut avoir seulement deux états en un premier temps, mais qui pourra avoir des états additionnels dans une utilisation ultérieure du protocole.

Exemple: **Statut-Marital :: = INTEGER {célibataire(0), marié(1)}**  
*en prévision de*

**Statut-Marital :: = INTEGER {célibataire(0), marié(1), veuf(2)}**

**III.3 Chaîne binaire**

- B11 Utiliser une chaîne binaire pour représenter des données binaires dont le format et la longueur ne sont pas spécifiés ou spécifiés dans une autre Recommandation, et dont la longueur en nombre de bits n'est pas nécessairement un multiple de huit.

Exemple: **Page-Télécopie-G3 :: = BIT STRING**  
– – *sequence binaire conforme à la Recommandation T.4*

- B12 Définir le premier bit et le dernier bit significatifs d'une chaîne binaire de longueur fixée comme bits remarquables.

Exemple: **Franchise :: = BIT STRING {premier(0), dernier(3)}**

TA4 Lorsqu'un membre particulier d'un ensemble a déjà été étiqueté dans un type particulier à une application, on ne doit pas utiliser pour l'étiqueter à nouveau une étiquette spécifique à un contexte, sauf si ceci est nécessaire pour distinguer les membres entre eux (ou risque de l'être ultérieurement). Lorsque le membre de l'ensemble a été étiqueté dans un type universel, pour un nouvel étiquetage, il faut utiliser une étiquette spécifique à un contexte.

Exemple: **Enregistrement-Produit :: = SET |**  
**Code-Uniforme,**  
**description [0] IMPLICIT Chaîne-IAS,**  
**no-Inventaire [1] IMPLICIT INTEGER,**  
**niveau-Inventaire [2] IMPLICIT INTEGER;**

**Code-Uniforme :: = [APPLICATION 13] IMPLICIT INTEGER**

TAS Utiliser des étiquettes spécifiques à un contexte pour distinguer les possibilités d'un choix. Affecter des étiquettes numériques commençant par zéro, si la seule contrainte qui leur est imposée est d'être distinctes.

Exemple: **Attribut-Client :: = CHOICE |**  
**nom [0] IMPLICIT Chaîne-IAS,**  
**adresse-Postale [1] IMPLICIT Chaîne-IAS,**  
**numéro-Compte [2] IMPLICIT INTEGER,**  
**solde-D0 [3] IMPLICIT INTEGER -- en francs -- }**

TA6 Lorsqu'une possibilité particulière de choix a été définie en utilisant un étiqueté de type particulier à une application, on ne doit pas utiliser d'autres étiquettes spécifiques à un contexte, sauf si ceci est nécessaire pour distinguer les possibilités entre elles (ou risque de l'être ultérieurement).

Exemple: **Indicateur-Produit :: = CHOICE |**  
**Code-Uniforme,**  
**description [0] IMPLICIT Chaîne-IAS,**  
**no-Inventaire [1] IMPLICIT INTEGER;**

**Code-Uniforme :: = [APPLICATION 13] IMPLICIT INTEGER**

TA7 Lorsqu'une possibilité particulière de choix a déjà été étiquetée dans un type universel, on ne peut utiliser pour celle-ci qu'un étiqueté spécifique à un contexte, sauf si le choix a précisément pour objet de permettre plusieurs types universels.

Exemple: **Identificateur-Client :: = CHOICE |**  
**nom Chaîne-IAS,**  
**numéro INTEGER;**

TA8 Utiliser un étiqueté de type à usage privé pour définir un type de données qui trouve son utilisation dans une organisation particulière ou dans un pays particulier et qui doit pouvoir être distingué (au moyen de sa représentation) de tous les autres types de données utilisées par cette organisation ou par ce pays.

Exemple: **Numéro-Badge-ACME :: = [PRIVATE 2] IMPLICIT INTEGER**

TA9 Ces dernières règles utilisent, dans les exemples, l'étiquetage implicite chaque fois que cela est autorisé. Il en résulte une représentation compacte, ce qui est hautement souhaitable dans certaines applications. Dans d'autres applications, la compacité peut être moins importante que, par exemple, la possibilité d'effectuer une vérification rigoureuse du type. Dans ce dernier cas, on peut utiliser l'étiquetage explicite.

Voir également les règles ST1, ST2, CH1 et CH2.

### III.9 Choix

CH1 Utiliser un choix pour représenter une variable qui est choisie à partir d'un ensemble de variables dont le nombre est connu et petit. Identifier chaque variable possible par un étiquetage spécifique au contexte.

Exemple: **Identificateur-Fichier :: = CHOICE |**  
**nom-Relatif [0] IMPLICIT Chaîne-IAS,**  
**-- nom de fichier (par exemple, «Rapport d'Avancement de Mars»)**  
**nom-Absolu [1] IMPLICIT Chaîne-IAS,**  
**-- noms de fichiers et répertoire les contenant**  
**-- (par exemple, «Rapport d'Avancement de Mars <Williams>»)**  
**numéro-Série [2] IMPLICIT INTEGER -- identificateur de fichier affecté par le système -- }**

CH2 Utiliser un choix pour représenter une variable qui est choisie à partir d'un ensemble de variables dont la disposition a des chances de changer d'une version du protocole à la suivante. Identifier chaque variable possible en l'étiquetant dans un type spécifique au contexte.

Fascicule VIII.7 – Rec. X.409

Fascicule VIII.7 – Rec. X.409

---

## 8 Couche 7/OSI : APPLICATION

Cette couche donne lieu à l'établissement d'une assez grande variété de standards spécifiques d'une gamme d'applications variée. Ces standards peuvent être spécifiés par l'OSI, mais aussi par des branches professionnelles (par exemple: banque pour les applications bancaires ou interbancaires) qui en demande ensuite la prise en compte par l'OSI. De grandes applications sont déjà traitées : messagerie interpersonnels (X400), gestion de fichiers (FTAM), Messagerie industrielle (MMS), Systèmes transactionnels (TP). D'autres comme l'Appel de Procédures Distantes (RPC) ou l'Administration de Réseaux (Networks management) sont encore à l'état de projet.

Des éléments de services communs ont aussi été spécifiés : ACSE, ROSE, CCR, RTSE.

La complexité de cette couche a conduit à définir une architecture propre : ALS (Application Layer Structure) pour organiser les services qui y sont inclus.

Cette architecture, modifiée plusieurs fois; utilise les concepts d'objet, qui sont de plus en plus inclus dans les services de niveau Application.

### 8.1 Rôle et Services :

#### Les Services Application

→ donnent aux **processus d'application** les moyens d'accéder à "l'environnement OSI" , c'est à dire aux autres sous-systèmes interconnectés selon les logiciels de communication OSI.

→ fournissent, grâce au service Présentation (ou au Service Session si la couche 6 est vide) toutes les fonctionnalités attachées aux couches 1 à 6 OSI

Diverses fonctions peuvent être ajoutées dans les entités d'Application  
soit

dans un Service d'Application commun à plusieurs types d'applications

soit dans des Services d'Application spécifiques

par exemple    accès à des bases de données  
                  transfert de fichiers  
                  soumission de travaux  
                  messagerie interpersonnels  
                  messagerie pour systèmes de production  
                  appel de procédures distantes  
                  systèmes transactionnels  
                  accès à des bases de données  
                  transactions bancaires  
                  transmission d'ordres

Les standards ou projets de standards portent aussi bien sur les services communs

ACSE : Application Common Service Elements (OSI)

ROSE : Remote Operation Service Elements (OSI/CCITT)

MTS : Service de Transfert de Messages (CCITT)

CCR : Commitment, Concurrency and Recovery

les services spécifiques

FTAM : "File Transfert Acces and Manipulation" (OSI)

IPM : Messagerie interpersonnes (CCITT)

MMS : "Manufacturing Messages Service" (OSI)

RPC : Remote Procedure Call (ECMA, projet OSI)

VTP : Terminal virtuel (OSI)

TPI : "Text Processing Interchange"(traitement de texte)

TP : Transactions Processing (OSI)

RDA : Remote Data Acces (Pojet OSI)

X500 : Annuaire (CCITT)

CMIS : Common Management Information Service (OSI)

Ces deux classes de protocoles ont d'abord suggéré de diviser la couche Application en deux sous-couches ou plus : une sous-couche inférieure pour les Services communs et une sous-couche supérieure aux protocoles beaucoup plus divers pour les applications spécifiques. Depuis on a décidé d'implanté des "éléments de service" qui coopèrent dans des structures non hiérarchisées (ALS : Application Layer Structure), chaque sous-ensemble étant lié à une connexion de Présentation.

Le problème essentiel consiste à manipuler ou à transférer des données directement utilisables. Sur chaque système celles-ci sont codées dans une syntaxe spécifique. On doit donc utiliser une "syntaxe de transfert" commune aux applications communicantes; la syntaxe locale est traduite sur chaque système dans cette syntaxe (concrète) de transfert.

## 8.2 Fonctions des Services Application

La couche Application constitue l'unique moyen pour un processus d'Application d'accéder à l'environnement OSI. Elle sert de fenêtre entre processus d'Application correspondants pour **échanger des informations significatives**.

Pour cela chaque processus d'application est vu par son homologue à travers une entité d'application (AE) qui regroupe les aspects de ce processus ayant rapport à l'OSI.

Cette entité d'Application est composée d'un élément utilisateur et d'**éléments de service d'application spécifiques ou communs (ASE)**. Ces éléments sont structurés en groupes de fonction appelé services.

En plus du **transfert d'information** ces services peuvent comprendre ceux de la liste ci-dessous :

- identification des partenaires (par exemple par leur nom, adresse, description spécifique, description générique),
- détermination de leur disponibilité actuelle à entrer en communication,
- délivrance de l'autorisation à communiquer,
- agrément sur les mécanismes de préservation du secret,
- authentification des partenaires susceptibles d'entrer en communication,
- détermination de la méthodologie d'imputation des coûts,
- détermination de l'adéquation des ressources,
- détermination de la qualité des services acceptables (par exemple temps de réponse, débits, taux d'erreur acceptables et les coûts associés),
- synchronisation des applications coopérantes,
- choix des règles de dialogue, d'initialisation ou de libération,
- accord sur la responsabilité de récupération d'erreurs,
- accord sur la procédure de contrôle d'intégrité des données,
- **identification des contraintes portant sur la syntaxe des données** (structures de données, jeux de caractères).

La couche Application contient toutes les fonctions impliquant des communications entre systèmes ouverts et qui n'ont pas été réalisées par les couches inférieures. Ces fonctions peuvent être réalisées par des programmes ou des opérateurs humains.

Quand une occurrence particulière d'un processus d'application souhaite communiquer avec une occurrence d'un autre processus d'application d'un autre système ouvert elle doit faire appel à une occurrence d'entité d'application qui devient responsable d'établir une **association** avec une occurrence similaire dans l'autre système ouvert. Quand cette association est réalisée les processus d'application peuvent communiquer.

Les processus d'application associés doivent **pouvoir échanger des informations significatives quelqu'ils soient**. Pour cela ils doivent utiliser des représentations communes des données échangées. Ces concepts de représentation de données, appelées

syntaxes, sont du ressort de la couche Présentation . Par contre leur signification, leur sémantique, reste du ressort de la couche Application.

## 8.3 Syntaxe abstraite et syntaxe de transfert

Les différentes Applications qui peuvent communiquer imposent de véhiculer des informations très complexes, incluant éventuellement des chaînes de caractères de jeux divers. (contrairement aux couches inférieures où l'on peut se contenter de spécifier la valeur binaire de séquences d'octets).

Dans la couche Application, ces données complexes vont être **spécifiées** en appliquant des **règles de notation indépendantes de la technique de codage** utilisée pour représenter ces données. Une telle spécification de données de la couche Application est appelée **syntaxe abstraite** et est identifiée de manière nom ambiguë par son **nom de syntaxe abstraite**.

On appelle **syntaxe concrète** les conventions utilisées pour une représentation spécifique de ces données. Une **syntaxe de transfert** est une syntaxe concrète utilisée dans le transfert des données entre systèmes ouverts.

Les entités d'application peuvent utiliser n'importe quelle syntaxe concrète (**syntaxe locale**). La transformation entre les syntaxes locales et la syntaxe de transfert commune nécessaire à la communication est réalisée par la couche Présentation.

Ainsi, pour la couche Application, la couche Présentation ajoute au service Session les facilités suivantes :

choix de la syntaxe  
transformation de la syntaxe

Celle-ci comprend les conversions de code et de jeux de caractères, les modifications de disposition des données et l'adaptation des actions sur les structures de données.

Le choix de la syntaxe permet de fournir au départ une syntaxe par défaut et de modifier ce choix ensuite.

L'association d'une syntaxe abstraite et d'une syntaxe de transfert est appelée **contexte de présentation** . Du point de vue de l'utilisateur un contexte de présentation représente une utilisation spécifique d'une syntaxe abstraite.

L'OSI a normalisé un langage de description de données, indépendant de la représentation des données transférées dans le réseau, c'est à dire une **syntaxe abstraite** appelée **Abstract Syntax Notation 1 (ASN.1)**.(OSI 8825-1-CCITT X208, ITU-T X.690). A cette syntaxe abstraite est associé un jeu de **règles de codages (règle d'encodage de base)** (OSI 8825-1, CCITT X209) qui déterminent les valeurs des octets de données transmis à la couche Session (syntaxes de transfert).

Cette syntaxe de transfert ASN.1 et les règles de codages sont utilisées de manière systématique dans les spécifications des protocoles correspondants aux services Application normalisés, mais aussi dans des Applications de plus haut niveau comme l'échange de données informatisé (EDI) ou d'autres domaines d'applications distribuées.

## 8.4 Structure de la couche Application

### ALS : Application Layer Structure

#### Norme: OSI 9545

Les systèmes de communication OSI sont conçus pour supporter les besoins de communications d'applications (c'est à dire de processus de traitement de l'information) qui demandent une coordination des activités de traitement de deux Systèmes Ouverts ou plus.

La couche Application (7/OSI), en particulier, définit des procédures qui supportent un traitement d'information distribué. Ces procédures sont contenues dans des "Entités d'Application". En pratique ces entités sont génériques et une ou plusieurs invocations (AEI : Application Entity Invocation) peuvent être implantées à un instant donné sur un Système Ouvert.

Les services Application s'appuient sur les couches inférieures, en particulier la couche Présentation qui fournit des facilités pour représenter les informations échangées et la couche Session qui contient les mécanismes de contrôle des interactions (points de synchronisation, jetons).

La couche Application diffère des autres couches du Modèle de Référence OSI sur différents points, notamment sa structure.

#### 8.4.1 Concepts de base

Dans le Modèle de Référence, la coopération entre Systèmes Ouverts réels est modélisée en terme d'interactions entre Processus d'Application (AP). Un Processus d'Application est une représentation abstraite des éléments d'un système réel qui exécute le traitement de l'information pour une application particulière. Les AP peuvent communiquer de manière permanente ou intermittente; d'autre part, dans un système distribué, l'ensemble des AP qui coopèrent peut varier au cours du temps.

Pour communiquer, chaque Processus d'Application utilise une ou plusieurs entités d'Application (AE). Une Entité d'Application représente un ensemble de facultés de communication d'un Processus d'Application particulier.

Une AE représente une et une seule AP. Plusieurs AP peuvent être représentées par des AE de même type. Une AP peut être représentée par un ensemble d'AE de différents types (éventuellement du même...).

Une AE est décomposée en "Eléments de Service Application" (ASE).

Un ASE est un ensemble de fonctions qui fournit des fonctionnalités de communication dans un dessein spécifique.

Les ASE pairs de deux Systèmes Ouverts échangent des APDU (Unité de données de protocole d'Application) qui sont placées dans la zone de données utilisateur de PPDU (PDU de Présentation), par exemple P\_CONNECT pour l'établissement d'une association ou P\_DATA dans la plupart des cas.

Les communications entre AE distantes peuvent être simples ou multiples. Une communication simple est réalisée par l'association d'un ensemble d'ASE situés dans chaque Système Ouvert. L'association d'un tel ensemble et sa rupture sont réalisées par un ASE particulier: ACSE (Elément de Service de Commande d'Association). Lorsque des relations multiples et coordonnées doivent être établies, on utilise une communication multiple.

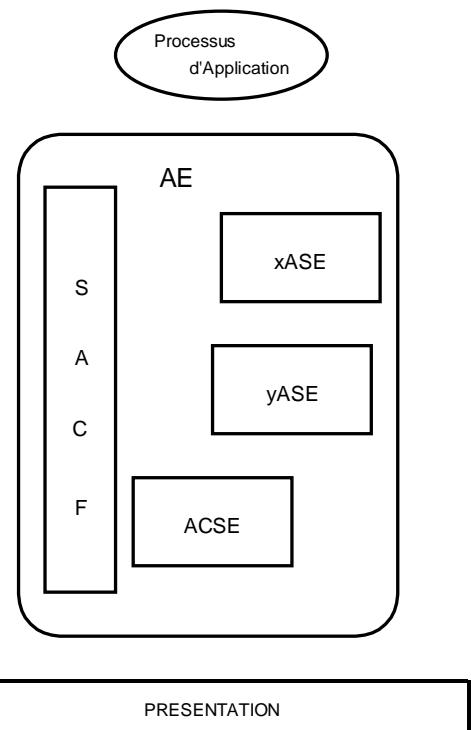
#### 8.4.2 Objet Association Simple :SAO

*nota* : des modifications récentes ont été introduites dans le standard; elles modifient légèrement la notion de SAO, la remplaçant par le concept d'ASO : Application Service Object.

Les Objets Association Simple regroupent un ensemble d'ASE qui doivent coopérer pour fournir un service cohérent à un Processus d'Application. Il comporte un ou plusieurs ASE spécifiques, ACSE et une fonction de commande d'association simple: SACF (Single Association Control Function).

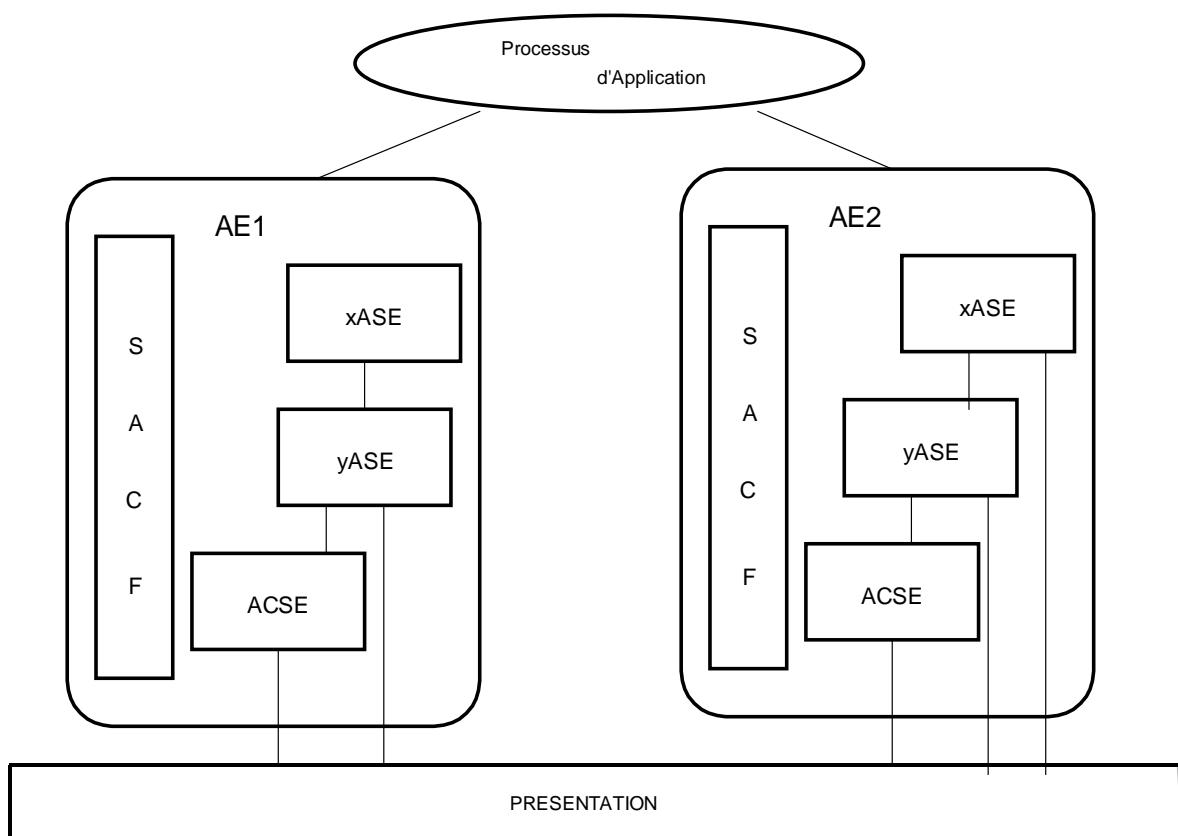
SACF modélise la coordination des interactions des ASE de l'Objet Association Simple (SAO) et leur utilisation coordonnée du service Présentation. Les ASE d'un SAO peuvent travailler de manière indépendante ou hiérarchique

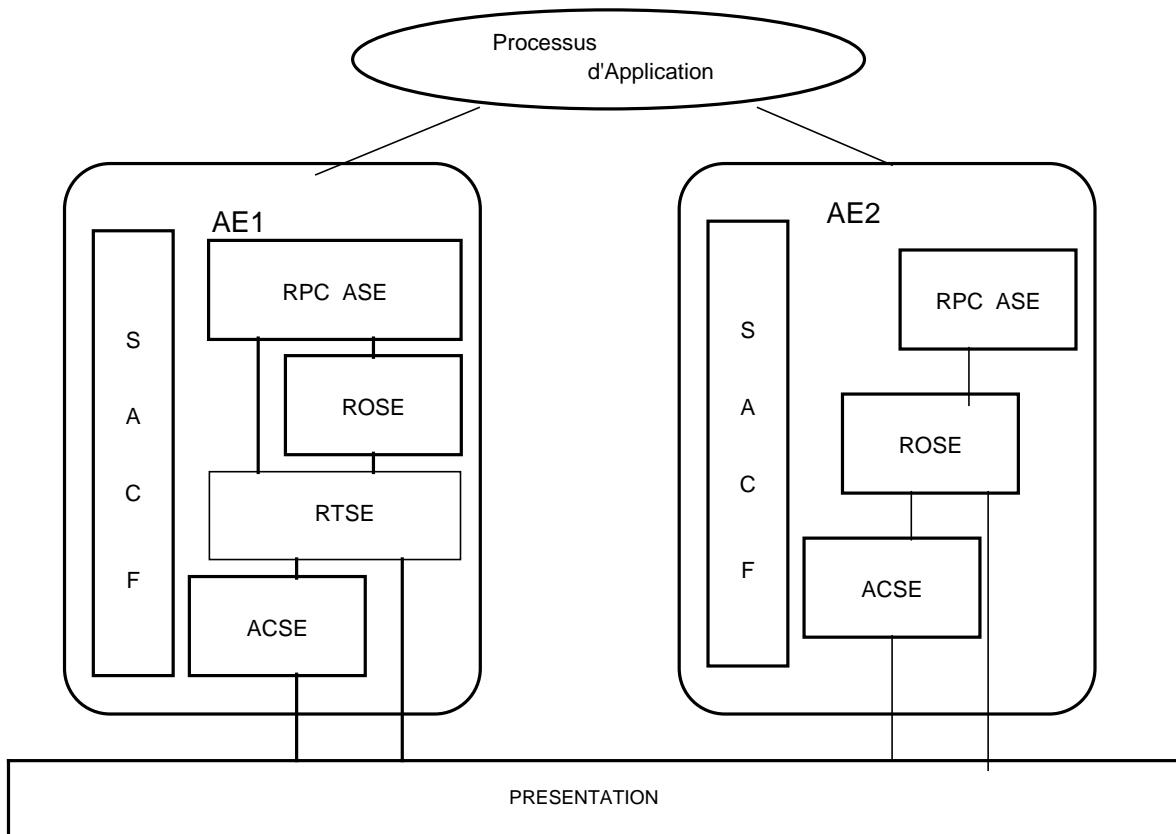
Ainsi toutes les ASE d'un SAO utilisent ACSE pour établir leurs associations. Dans certains cas, les ASE interagissent sans véritable relation hiérarchique (à ce sujet voir les relations entre les ASE CCR et TP, notion de CMS).



#### Exemples de structures de SAO:

Les deux exemples ci-dessous correspondent aux deux architectures prévues pour réaliser un système d'appels de procédures distantes. Ce service RPC (Remote Procedure Call) utilise une ASE spécifique RPCASE, un élément de service d'opérations distantes ROSE (Remote Operation Service Element), ACSE et, éventuellement, un service de transfert fiable fourni par RTSE (Reliable Transfert Service Element). Dans ce cas particulier, la coopération de ces ASE est assez hiérarchique.





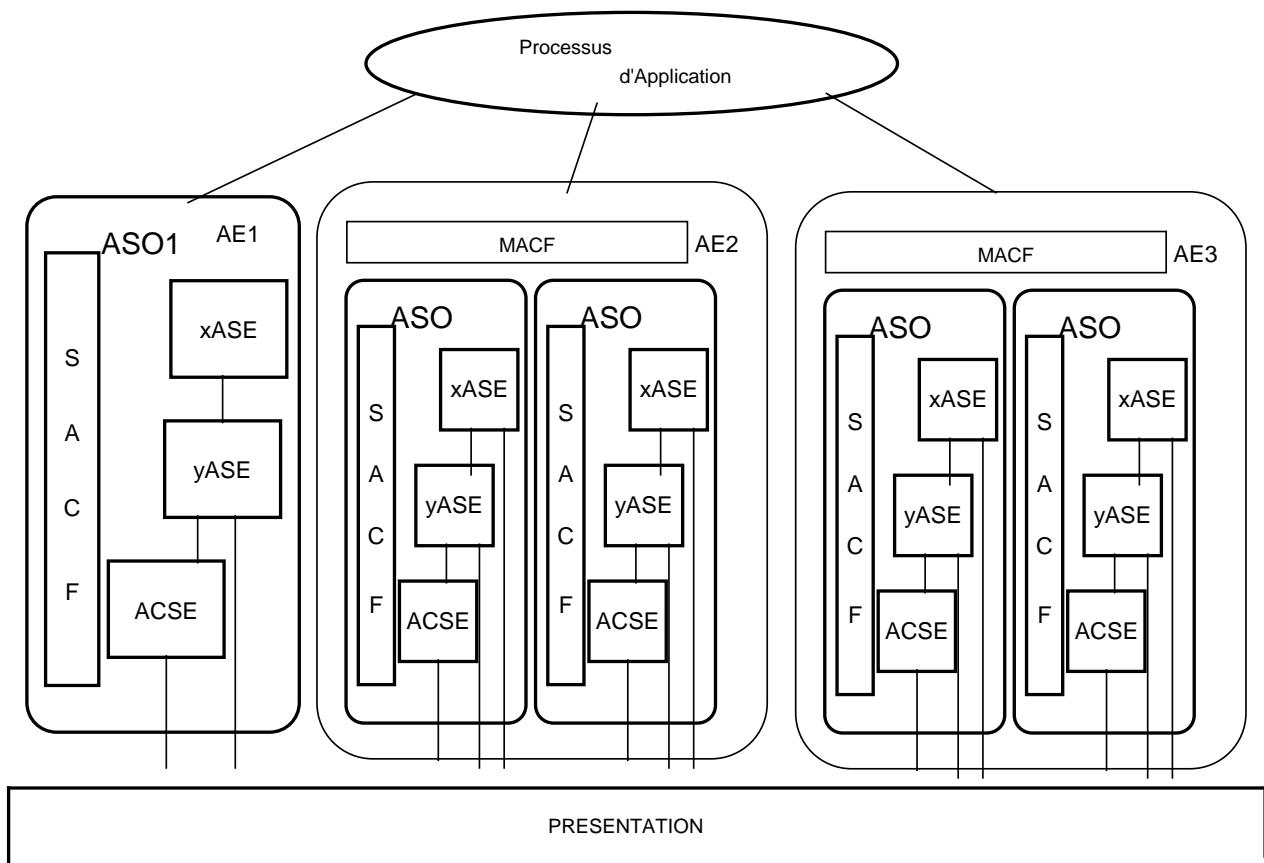
### 8.4.3 Structure d'une Entité d'Application

Une Entité d'Application peut utiliser une communication simple ou multiple. Dans ce cas elle est composée de plusieurs SAO dont les activités doivent être coordonnées. Ceci nécessite :

- le séquencement des activités des différentes associations.
- le maintien de la consistance des relations entre les activités des différentes associations.
- des règles spécifiques d'utilisation.

Ces fonctions sont fournies par l'élément de service d'associations multiples MACF (Multiple Association Control Function).

On obtient ainsi la structure générale ci-dessous :



On appelle "invocation d'Entité d'Application" (AEI : Application Entity Invocation) une utilisation spécifique d'une partie ou de toutes les fonctionnalités d'une AE donnée, permettant de supporter les besoins de communications à l'occasion d'un traitement d'information spécifique.

Un "Contexte d'Application" est un ensemble de règles partagées par **deux invocations d'AE** qui gouverne leur comportement dans une Association d'application donnée, pour permettre leur coopération.

#### **8.4.4 Noms et Fonctions répertoire**

Les fonctions répertoire d'application (Application Directory Functions) traitent les adresses Présentation, les intitulés d'Entité d'Application (AE-Title) et les informations d'adressage des protocoles d'application. Elles sont exécutées par l'AE pour en dériver les fonctions d'adressage nécessaires.

Ces informations peuvent être distantes ou locales et disponibles directement par les fonctions répertoire. Lorsqu'elles sont distantes, des protocoles OSI sont utilisés pour y accéder, soit par un ASE particulier, soit en utilisant les services d'une autre Entité d'Application, voir d'une AE d'un processus d'application différent.

Ces fonctions sont indépendantes des AE, comme les fonctions d'administration ou de sécurité. Elles permettent d'identifier huit types d'objets par leur nom:

- 1) Intitulé de processus d'application
- 2) Intitulé d'Entité d'Application -
- 3) Identificateur d'invocation d'AP
- 4) Identificateur d'invocation d'AE -
- 5) Identificateur d'association d'application -
- 6) Intitulé de type d'AP -
- 7) Intitulé de type d'AE -
- 8) Intitulé de système

#### **8.4.5 Définition d'une syntaxe abstraite**

Une syntaxe abstraite est composée des règles de spécification formelle des données. Ces règles sont indépendantes des techniques de codage utilisées pour représenter les données.

Pour un ASE donné, la structure des APDU est spécifiée par une ou plusieurs syntaxes abstraites. La structure des données utilisateur convoyées par ces ASE (dans les APDU) sur une Association est spécifiée par une ou d'autres syntaxes abstraites.

Un nom de syntaxe abstraite est associé à la définition d'un ensemble d'APDU ou à la définition d'un ensemble de données utilisateur.

Ce nom est utilisé par le service Présentation pour négocier une représentation de ces informations ou APDU qui soit mutuellement acceptée.

---

## 9 Couche 7/OSI : SERVICES COMMUNS D'APPLICATION

### 9.1 ACSE : Commande d'Association Association Control Service Element

Normes OSI : Service ⇒ IS 8649  
Protocole ⇒ IS 8650

#### 9.1.1 Introduction :

Le Modèle de Référence représente des communications entre des processus d'application en termes de communication entre leurs "Entités d'Application".

Une Entité d'Application regroupe une ensemble d'"Eléments de Service Application" (ASE). Pour interagir, ces Entités d'Application doivent être associées.

Cette "Association" d'Entités d'Application est réalisée explicitement et permet d'identifier l'ensemble des ASE qui constituent l'entité, les options retenues et toutes les autres informations nécessaires pour obtenir l'interopérabilité des processus d'application.  
Ces informations (ASEs, options, etc.) constituent un "Contexte d'Application".

ACSE est un élément de service (ASE) constituant de toute Entité d'Application. Il permet d'établir et de rompre l'association entre deux Entités d'Application paires.

#### 9.1.2 Service fourni

ACSE permet de réaliser une Association simple entre deux invocations d'Entités d'Application.

Il offre 4 services :  
A\_ASSOCIATE  
A\_RELEASE

permettent d'associer et de dissocier des applications en fonctionnement normal.  
Ces services sont confirmés.

A\_ABORT  
A\_P\_ABORT  
traitent les ruptures anormales d'association (sur défaut).

A\_ABORT est un service simple (non confirmé). A\_P\_ABORT est un service fourni directement par ACSE aux deux entités utilisatrices.

A\_ASSOCIATE initie une Association et permet de définir le Contexte d'Application utilisé (nom de contexte et paramètres).

### 9.1.3 Relation avec les autres ASE d'une entité d'Application

Les autres ASE standardisés par l'OSI ne possèdent aucun mécanisme d'établissement ou de rupture d'association. Ils ont recours pour cela à ACSE. Pour leurs autres fonctionnalités, ils utilisent d'autres ASE (par exemple ROSE ou CCR) et/ou directement le service Présentation.

Souvent les constructeurs offrent les services ACSE et Présentation dans le même interface ou la même bibliothèque.

### 9.1.4 Service requis

ACSE s'appuie sur le service Présentation et, à travers lui, le service Session. Une correspondance un pour un est établie entre une Association d'applications et une connexion de Présentation. ACSE nécessite donc un accès aux primitives :

P\_CONNECT  
P\_RELEASE  
P\_ABORT

Si l'utilisateur de la primitive d'association A\_ASSOCIATE spécifie qu'il n'utilise qu'un seul contexte de Présentation, ce contexte inclura aussi la syntaxe abstraite spécifiée par le protocole ACSE. Dans le cas contraire, A\_ASSOCIATE utilise son propre contexte de Présentation. Le service Présentation doit donc offrir la fonctionnalité : Contextes multiples

Lors de l'établissement de l'Association, le demandeur spécifie les unités fonctionnelles du service Session à mettre en oeuvre. Ce choix affecte le service de rupture d'association A\_RELEASE de la manière suivante :

Si les jetons de Session sont utilisés, l'initiateur de A\_RELEASE doit les posséder; l'accepteur de A\_RELEASE ne peut refuser la rupture d'association que si l'unité fonctionnelle de Session "Terminaison négociée" est en service.

### 9.1.5 Description succincte des services

La table ci-dessous donne les paramètres de la primitive A\_ASSOCIATE. On notera que ces paramètres permettent d'identifier les contextes d'Application et de Présentation, les adresses appelantes et appelées (en retour l'adresse réelle de l'appelé), la facilité de service demandée, le point de synchronisation initial et la position initiale des jetons.

Nom de paramètre	Req	Ind	Resp	Cnf
------------------	-----	-----	------	-----

Calling Application Entity Title	U	U=		
Called Application Entity Title	U	U=		
Responding Application Entity Title			U	U=
Application Context Name	M	M=	M	M=
User Information	U	U=	U	U=
Result			M	M=
Calling Presentation Adress	P	P		
Called Presentation Adress	P	P		
Responding Presentation Adress			P	P
Single Presentation Context	U	U=		
Presentation Context Definition List	P	P		
Presentation Context Definition Result		P	P	P
Default Presentation Context Name	P	P		
Default Presentation Context Result		P	P	P
Quality of Service	P	P	P	P
Presentation Requirements	P	P	P	P
Session Requirements	P	P	P	P
Initial Synchronization Point Number	P	P	P	P
Initial Assignment of tokens	P	P	P	P
Session-Connection Identifier	P	P	P	

nota: M obligatoire

U option utilisateur

P présence sujette aux conditions de OSI8822

= paramètre égal à valeur de colonne précédente

Dans les primitives A\_RELEASE et A\_P\_ABORT on trouvera, en particulier, un code Raison et, éventuellement, des informations utilisateur pour préciser la cause de la rupture.

### 9.1.6 Description succincte du protocole

Le protocole est assez simple . Il comporte essentiellement l'échange de 5 APDU : AARQ, AARE, RLRQ, RLRE, ABRT. L'automate d'état ne comporte que 5 états :

Les états Repos et Associé , l'état d'attente de confirmation d'association et les états d'attente de réponse (de la part de l'utilisateur) d'acceptation d'association ou de rupture.

AARQ est utilisée par la primitive A\_ASSOCIATE pour demander l'établissement d'une association. Elle comporte 5 champs paramètres dont un numéro de version obligatoire. Les deux suivants sont optionnels ; ce sont les titres des entités d'Application appelante et appelée. Le quatrième (obligatoire) est le nom du Contexte d'Application. Le dernier champ, optionnel, peut transporter des données utilisateur. La syntaxe de ce PDU est donnée à titre d'exemple.

AARQapdu ::= [ APPLICATION 0 ] IMPLICIT SEQUENCE

```

protocolVersion      [0] IMPLICIT BIT STRING
                     {Version1 (0)},
calledAETitle        [1] ApplicationTitle
                     OPTIONAL,
callingAETitle       [2] ApplicationTitle
                     OPTIONAL,
applicationContextName [3] ApplicationContextName,
userInformation      [4] AssociationData
                     OPTIONAL

```

ApplicationTitle ::= OBJECT IDENTIFIER

ApplicationContextName ::= OBJECT IDENTIFIER

AssociationData ::= EXTERNAL

AARE est utilisée en réponse à AARQ. Il comporte aussi 5 champs paramètres: protocolVersion, result, respondingAETitle, ApplicationContextName, userInformation. La version et le nom de contexte sont seuls obligatoires.

RLRQ est utilisée par la primitive A\_RELEASE pour demander la rupture de l'association; RLRE sera utilisé en réponse à RLRQ. Ces APDU ne comportent que deux champs optionnels : raison et information utilisateur.

ABRT est utilisée par les primitives A\_ABORT. Un premier paramètre obligatoire indique la source (utilisateur ou service) de cet abandon. Le second, optionnel, permet de transporter des données utilisateur.

Ces APDU sont placées dans les données utilisateur des PDU de Présentation utilisées pour les transporter.

Les paramètres des primitives de ACSE qui ne sont pas pris en compte par ces APDU sont directement projetés sur les paramètres correspondants des primitives d'appel du service Présentation, notamment les adresses, les noms de contexte, la qualité de service demandée ou les caractéristiques du service Session requis.

## 9.2 RTSE : Service de Transfert Fiable

### Reliable Transfert Service Element

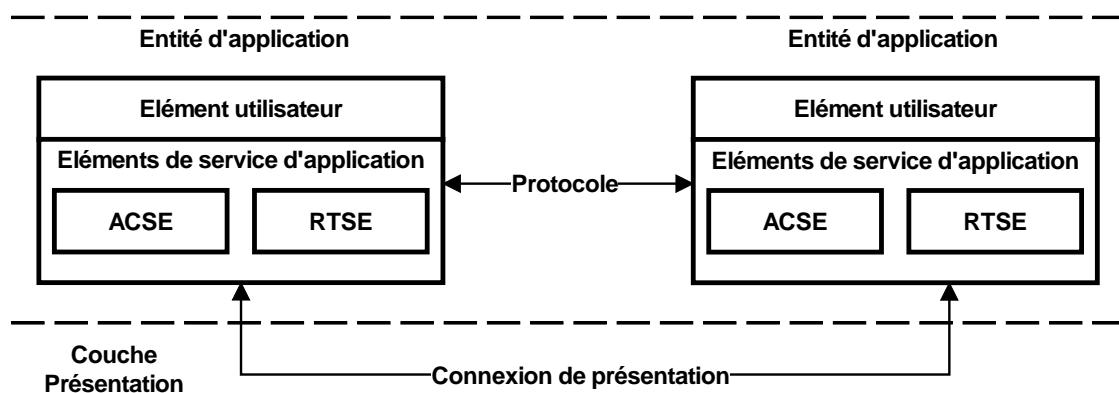
Norme OSI : 9066      Norme ITU (CCITT) : X.218 et X.228

#### 9.2.1 Introduction

Le transfert fiable est un mécanisme indépendant de l'application qui permet d'assurer le transfert d'unités de données d'un protocole d'application entre systèmes ouverts et la

reprise à la suite d'une défaillance de la communication et du système terminal et de minimiser l'importance des retransmissions.

Le transfert fiable garantit que chaque APDU, émise par une entité d'application émettrice vers une entité d'application réceptrice, est transférée une fois exactement ou que l'entité émettrice est avisée d'une anomalie. Il est exécuté dans le contexte d'une association d'application, établie par ACSE, qui définit la relation entre les entités d'application. Le contexte d'application illustré par le schéma ci-dessous définit un contexte minimal pour une application impliquant RTSE. Défini à l'origine dans le cadre de la messagerie X400, le contexte d'application correspondant est celui d'un agent de transfert de message (MTA) qui englobe l'élément de service MTSE. RTSE peut être utilisé dans d'autres contextes, en particulier FTAM.



## 9.2.2 Description du service

Le service de Transfert fiable comporte 7 services élémentaires qui peuvent être regroupés en trois sous-ensembles.

Il peut fonctionner selon 2 modes : normal et « X410-1984 ». Ce mode, qui implique des restrictions, est fourni pour permettre l'interfonctionnement avec des applications anciennes. Ce mode est défini, si nécessaire, en phase d'établissement d'association.

### 9.2.2.1 Etablissement et Rupture d'association

Les services d'« Etablissement d'association » et de « Fermeture d'association » permettent à un utilisateur de RTSE de demander l'établissement et la terminaison négociée d'une association d'application. Il ne peut demander cette terminaison que s'il bénéficie du « Tour » (Voir Droit de terminaison dans la Session ISO). Ces deux services sont des services confirmés.

L'association peut aussi être terminée brutalement soit par un utilisateur soit par le service RTSE lui-même en cas de défaillance du système de communication. Les deux services « Coupure d'Association par le Fournisseur » et « Coupure d'Association par l'Utilisateur » sont des services non confirmés.

Tous ces services s'appuient sur ACSE.

### **9.2.2.2 Transfert de données**

Le transfert de données peut se faire selon 2 modes de Dialogue : mode « Monologue » ou mode « Bidirectionnel à l'alternat ». Le mode de fonctionnement est défini lors de l'établissement d'association.

Le service de Transfert de données, est un service engendré par le fournisseur (confirmé automatiquement) qui ne comporte pas de primitive « Réponse ». Le récepteur délivre une confirmation, positive ou négative, à son utilisateur et émet une PDU qui indique que le message a pu être délivré ou non dans le temps spécifié pour le transfert.

Ce service est conditionné à la possession du Tour, c'est à dire du droit d'émettre des données utilisateur. Il ne peut donc y avoir de collision.

Outre les données à transférer (APDU), la requête de Transfert fiable contient un paramètre « Temps de Transfert » indiquant le laps de temps pendant lequel le fournisseur de RTSE doit transférer avec succès l'APDU à l'autre utilisateur. Ce paramètre doit être fourni par le demandeur du service.

La primitive de confirmation du Transfert contient le paramètre « Résultat » qui spécifie si les données ont été transférées et mises en sécurité par le RTSE récepteur ou si cela n'a pu être fait dans le laps de temps demandé. Exceptionnellement, une confirmation négative peut être signalée même si le transfert est tout à fait correct.

### **9.2.2.3 Changement de Tour**

Ce service simple (non confirmé) permet à un utilisateur de RTSE de demander le Tour, c'est à dire le droit de transmettre des données ou de terminer l'association.

Cette fonction est exécutée, en fait, par la Session à travers le Transfert de Jetons.

Ce sous-ensemble se compose d'un service de « Demande de Changement de Tour » et d'un service de « Changement de Tour » qui permet à un utilisateur de céder le Tour, soit de manière spontanée soit à la suite d'une demande. Ce Tour ne peut être cédé que si aucune primitive de Transfert de Donnée n'est en instance.

La Demande de Changement de Tour transfert la priorité de l'action à entreprendre à l'autre utilisateur qui peut décider du moment auquel il cédera le Tour.

Le Tour initial est défini lors de l'établissement d'association.

### **9.2.3 Services requis**

RTSE utilise les services fournis par ACSE et la Présentation ISO, en particulier, les services de la Présentation qui sont réalisés en pratique par la Session ISO : **Gestion d'Activité**, Pose de points de synchronisation mineurs, Jetons, Reprise, Gestion d'anomalies. Un certain nombre de primitives sont « mappées » (projetées) directement sur le service ACSE (par exemple la terminaison normale) ou le service Présentation, par exemple la reprise en cas d'anomalie ; dans ce cas aucune APDU spécifique n'est utilisée.

### **9.2.4 Protocole RTSE**

Le protocole RTSE ne comporte que 6 PDU :

RTAB : utilisée pour la rupture brutale d'association, quelque soit son origine (utilisateur ou fournisseur. Aucune PDU n'est utilisée pour une terminaison normale.

- RTORQ : utilisée pour la demande d'ouverture d'association
- RTOAC : utilisée pour l'acceptation d'ouverture d'association
- RTORJ : utilisée pour le refus d'ouverture d'association
- RTTR : utilisée pour le transfert de données
- RTTP : utilisée pour la demande de Jeton

Malgré ce nombre restreint de PDU et de primitives de service, l'automate de la machine protocolaire (RTPM) est trop complexe pour être donné ici. En effet elle doit tenir compte du fonctionnement normal, mais aussi de tous les cas de reprise sur anomalie.

Nous nous contenterons de donner une description succincte du protocole, et en particulier des paramètres utiles pour son bon fonctionnement.

### **APDU RTORQ**

Utilisée pour la demande d'ouverture, elle peut comporter 6 paramètres optionnels : la taille du point de reprise, la taille de la fenêtre, le mode de dialogue, des données utilisateurs (uniquement dans la procédure d'établissement), un identificateur de connexion de session (uniquement dans les cas de reprise d'association) et le protocole d'application (dans le mode X410-1984 uniquement).

### **APDU RTOAC**

En réponse à une demande d'établissement, cette acceptation peut comporter 4 paramètres optionnels : la taille du point de reprise, la taille de la fenêtre, des données utilisateurs (uniquement dans la procédure d'établissement), un identificateur de connexion de session (uniquement dans les cas de reprise d'association)

### **APDU RTORJ**

Le refus d'association ne comporte qu'au plus 2 paramètres optionnels : La raison du refus en mode X410-1984 et des données utilisateur uniquement en cas de reprise.

Si l'association d'application n'est pas acceptée par le fournisseur d'ACSE, aucune indication n'est fournie à la machine protocolaire et aucune action n'est réalisée. Si ACSE accepte l'association, RTSE examine la demande et l'accepte et émet une indication d'ouverture vers l'utilisateur, ou la refuse par RTORJ. Dans ce cas RTSE n'émet aucune indication d'ouverture vers son utilisateur. Sur la réponse de l'utilisateur, RTOAC est transmis.

La terminaison normale d'association est réalisée par ACSE sans transmission de PDUs spécifiques. (terminaison par le service inférieur).

### **APDU RTTR**

Cette PDU ne comporte qu'un **champ obligatoire** : les données utilisateur.

En cas d'interruption du transfert, aucune PDU n'est transmise mais la machine protocolaire reçoit une Indication d'Interruption d'Activité de Présentation et y répond. La machine qui reçoit une primitive de Confirmation d'Interruption d'Activité de Présentation initie la reprise.

### **APDU RTTP**

La PDU utilisée pour le changement de Tour ne comporte qu'un paramètre optionnel : la priorité.

Sur une primitive de Demande de Tour, la PDU RTTP est émise par l'intermédiaire d'une primitive de Demande de Jeton de Présentation.

La primitive de Cession de Tour ne provoque l'envoi d'aucune PDU. Elle est simplement projetée sur une primitive de Passation de Commande de Présentation (cession de Jeton)

### **APDU RTAB**

Dans les cas d'erreur grave, où une reprise ne peut être réalisée, l'utilisateur ou le fournisseur de RTSE peut entreprendre une rupture brutale d'association. Il utilise alors la PDU RTAB qui peut comporter 3 paramètres optionnels : la raison de la rupture, le paramètre renvoyé, des données utilisateur (pour mieux expliciter cette rupture). A titre indicatif, la raison de rupture peut être : problème local, paramètre invalide, activité non reconnue, problème temporaire, erreur de protocole (de la machine protocolaire (RTPM), erreur permanente, rupture par l'utilisateur et transfert(déjà) achevé.

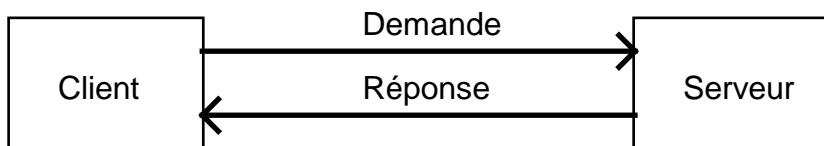
## 9.3 ROSE : Service d'opérations distantes

### 9.3.1 Introduction

Le concept d' "opérations distantes" est utilisé comme véhicule supportant des applications interactives.

#### 9.3.1.1 Concepts

D'une manière générale, le concept d'"opération" est introduit dans le cadre d'une architecture **orientée objet**: une opération, distante ou locale, est une interaction élémentaire demande/réponse [request/reply], quelquefois demande seulement. Si la structure d'une telle opération entre deux entités est simple et régulière, son contenu syntaxique et sémantique peut être très complexe.



La demande prend la forme d'une structure de données :

< invoque > ::= < type d'opération > < arguments >

Le type d'opération distingue les différentes opérations applicables au type d'objet. Les arguments portent les types de données des paramètres fournis par le demandeur ( Invoker ). L'élément "réponse" prend la forme d'une structure de données **retour** :

- soit retour immédiat
- soit retour erreur

Cette structure a la forme :

< retourResultats > ::= < résultats >  
ou      < retourErreurs > ::= < type\_erreur > <paramètres\_erreur >

où résultats est un type de données paramètre dont les valeurs sont fournies par l'objet sur lequel l'opération a été demandée. Le type\_erreur identifie les types d'erreurs pouvant se produire et paramètres\_erreur donne des informations sur le diagnostic fourni par l'objet invoqué.

#### 9.3.1.2 Fiabilité

Une telle structure de commande, qui prévoit un retour normal et une ou plusieurs exceptions contribue beaucoup à la fiabilité du système. Une opération est dite *totale* si elle inclut tous les cas possibles, y compris les cas de défaut. Si chaque opération d'un type est une opération totale, ce type est **robuste**.

Dans le cas d'une opération distante, la fiabilité d'exécution peut être rangée dans une des trois classes suivantes :

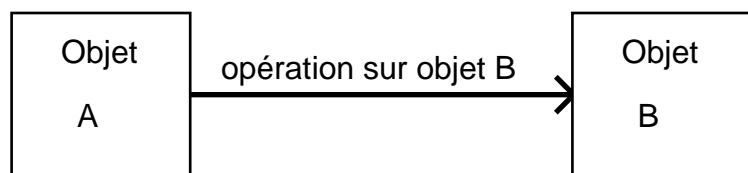
Exécution une fois exactement c'est la garantie la plus forte.

Exécution au moins une fois garantit que l'exécution a été réalisée au moins une fois. Ce cas est valable pour les opérations "idempotent" (par exemple une lecture de données).

Exécution au plus une fois garantit que l'exécution n'a pas été réalisée plus d'une fois.

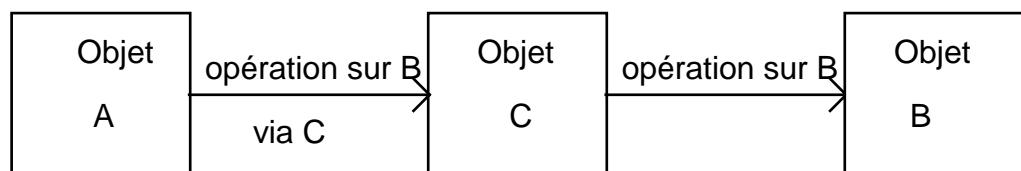
### 9.3.1.3 Exécution

L'exécution de toute opération sur un objet est réalisée dans un environnement d'exécution.



Il est recommandé de modéliser l'environnement d'exécution d'une opération par un type abstrait, interposé entre les objets extrêmes.

Le modèle ci-dessus devient :



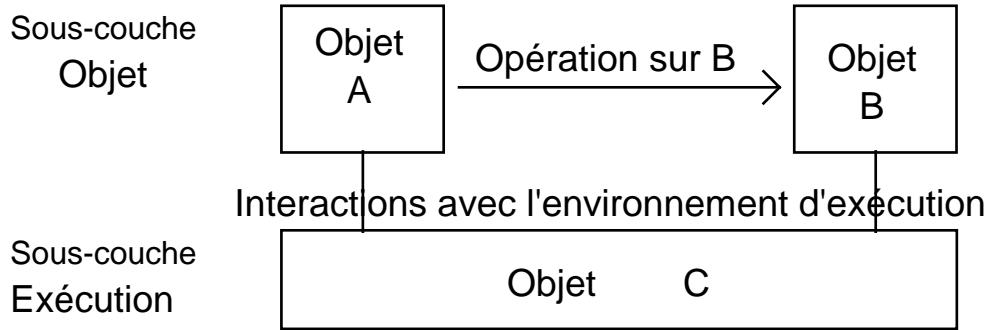
C : type abstrait d'environnement d'exécution

Entre les objets A et C, l'opération sur B est transportée de manière transparente dans les paramètres de l'opération sur C.

Le type abstrait C peut utiliser différentes techniques :

- opération distante basée sur un transfert en file
- appel de procédure locale synchrone (retour en fin d'échange)
- appel de procédure distante synchrone (retour fin d'échange)
- passage de message asynchrone (retour immédiat)

Ce modèle peut être redessiné en couches :



L'environnement d'exécution supporte seulement l'exécution des opérations (qui est un phénomène transitoire).

La persistance des instances d'objets et des résultats sur ceux-ci est une propriété des objets concernés.

#### 9.3.1.4 Nommage

A chaque objet est associé un nom de type. Sur cet objet, les types d'opération qui peuvent être effectuées sont distingués par un **nom d'opération** et un **numéro d'opération** qui en donne une forme compacte. Nom et numéro d'opération sont définis dans la spécification du type de l'objet.

#### 9.3.1.5 Liaison et contextes

Durant l'exécution d'une opération, il y a une relation entre les instances d'objets qui lie les objets entre eux.

Cette "liaison" (**binding**) peut n'exister que durant l'occurrence d'une opération ou peut se prolonger sur des opérations multiples. On distingue donc deux types de liaisons :

- "Liaison d'exécution" concernée par les ressources utilisées dans l'environnement d'exécution.
- "Liaison d'objet" concernée par la sémantique de la relation entre objets. Elle fait partie de la spécification du type de l'objet.

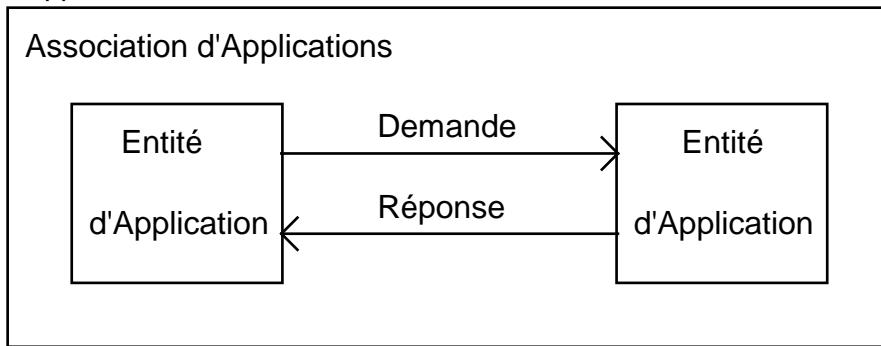
Cette "**liaison d'objet**" ne subsiste que pour une opération; on peut donc utiliser un "serveur sans états" pour la gérer.

Si la liaison subsiste pour plusieurs opération, on doit introduire la notion de "**contexte**". Dans ce cadre on échange des "titres" (handle: poignée) qui référencent des informations identifiées auparavant.

## 9.3.2 Modèle pour les opérations distantes.

### 9.3.2.1 Types d'opérations

Une opération invoquée par un "demandeur" (invoker) est exécutée par une autre entité d'application: l'exécutant.



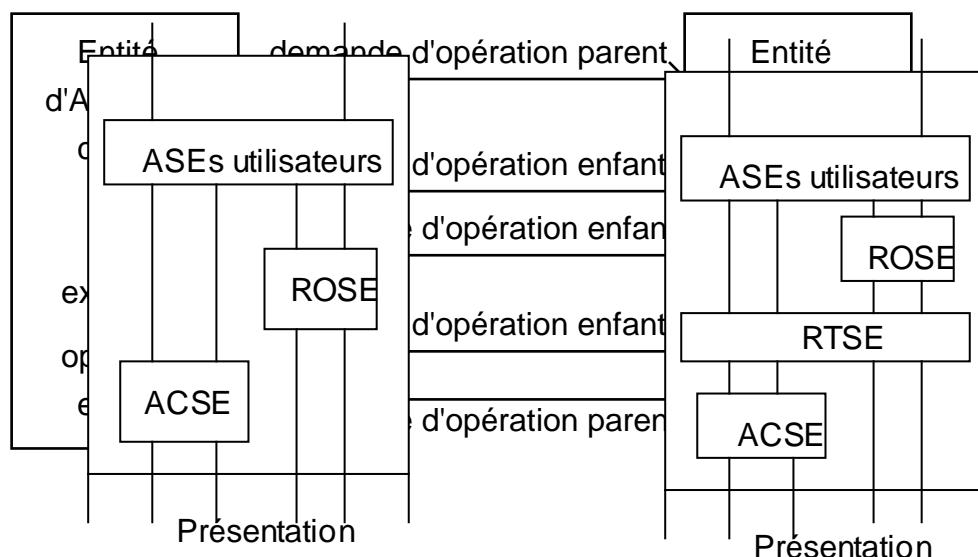
Les opérations sont classées en deux modes selon la réponse fournie :

- synchrone : réponse en fin d'échange
- asynchrone : d'autres opérations peuvent être demandées sans attendre l'arrivée de la réponse.

On définit ainsi 5 classes :

- 1 - synchrone avec réponse en cas de succès ou de défaut
- 2 - asynchrone avec réponse en cas de succès ou de défaut
- 3 - asynchrone avec réponse en cas de défaut seulement
- 4 - asynchrone avec réponse en cas de succès seulement
- 5 - asynchrone sans réponse

Une demande d'opération peut provoquer, pour réaliser son exécution, des demandes d'autres opérations. On dit que ces opérations sont **reliées** (linked) et on distingue les opérateurs parents et enfants.



Ces demandes (et les réponses associées) sont échangées dans le cadre d'une Association d'applications établie par une entité initiatrice qui peut seule la rompre.

On utilise 3 classes d'Association :

- 1 - seul l'initiateur peut invoquer des opérations
- 2 - seul l'accepteur peut invoquer des opérations
- 3 - les deux entités peuvent invoquer des opérations. Cette classe est nécessaire pour les opérations reliées.

### **9.3.2.2    Architecture OSI et services requis.**

L'association d'applications utilise l'ACSE (OSI 8649, X217) alors que l'échange des opérations est réalisé par l'Elément de service pour opérations distantes ROSE (OSI 9072).

On peut aussi utiliser, de manière optionnelle, l'élément de service de transfert fiable RTSE (OSI 9066, X218). On a donc, selon le cas, une des deux architecture ci-dessus .

Différents ASEs utilisateurs sont déjà configurés pour utiliser le service ROSE, notamment l'appel de procédures distantes RPC (Remote Procedure Call), le courrier électronique (X400) ou l'administration de réseaux (SM system management).

### **9.3.3    Service fournis**

#### **9.3.3.1    Types d'opérations distantes**

Dans un contexte hétérogène, pour définir toutes les opérations distantes nécessaires à la coopération de deux applications, il est utile de mettre en place une syntaxe de définition de ces opérations, qui permettra de s'abstraire des particularités de chacun des systèmes. On utilise la notion de macro définie dans la syntaxe ASN.1.

La spécification actuelle fournit des macros permettant de définir des opérations et les fonctions associées nécessaires :

BIND pour établir la liaison entre objets demandeur et exécutant et préciser l'endroit où commence le lien.

UNBIND pour rompre cette liaison, à un endroit déterminé

ERROR pour indiquer une situation de réponse négative

OPERATION pour définir un ensemble d'opérations utilisables par un couple d'objet demandeur/exécutant.

Une macro additionnelle, APPLICATION-SERVICE-ELEMENT, permet de définir un ASE pouvant utiliser le serveur d'opérations distantes; Une autre, APPLICATION-CONTEXT, permet de définir un contexte d'application entre un ensemble d'ASEs avec les opérations qu'elles peuvent échanger.

Cette définition sous forme de macros ne fournit qu'une spécification générale à projeter sur des environnements d'exécution variés. Ces macros ne sont que des éléments "papier" et ne sont employées que pour définir des fonctions avec leurs paramètres ainsi qu'un nom symbolique sous forme d'un numéro d'opération. Un serveur pourra ainsi envoyer à ses utilisateurs potentiels une sorte de catalogue des fonctions mises à disposition.

Par exemple :

Un serveur, écrit en pascal, peut fournir les fonctions

n°	nom de fonction	paramètres d'entrée	paramètres de sortie
1	dérivée	X,Y,Z réels	R réel
2	intégrale	X,A,B réels	S réel
3	imprime	F nom_fichier	J booléen
4	fonctionW	I,J entiers	N,M entiers

Un programme utilisateur, écrit en C par exemple, dispose des fiches macros OPERATION correspondantes. S'il désire appeler la fonction "intégrale", il fera appel dans son programme à intégrale (X,A,B,&S);

Une transformation devra être effectuée pour créer la primitive adressée au service ROSE

```
RO_INV_req invoke_id = 1
          op_val      = 2
          arg         = X,A,B
```

Il recevra en échange (après exécution à distance de la fonction)

```
RO_RES_ind    invoke_id = 1
              op_val     = 2
              result     = S
```

Cette indication sera transformée en retour de la fonction intégrale(X,A,B,&S)

Ainsi deux interlocuteurs ne parlant pas le même langage peuvent interagir.

Dans cet exemple nous avons fait appel à deux primitives du service ROSE décrit ci-dessous.

### 9.3.3.2 Élément de service pour opérations distantes ROSE

ROSE fournit un environnement d'exécution aux opérations identifiées ci-dessus. Ses services permettent d'invoquer des opérations distantes et d'en indiquer le résultat :

RO-INVOKE permet à une entité d'application de demander une opération

RO-RESULT et RO-ERROR permettent à une entité d'application exécutante de retourner une réponse positive ou négative à cette demande d'opération

RO-REJECT-U et RO-REJECT-P permettent à une entité utilisatrice (U) ou fournisseur (P) du service d'opérations distantes d'informer l'autre (ou les deux) entité(s) utilisatrice(s) qu'un problème a été détecté et que l'opération n'a pu être exécutée, soit à la suite d'un problème de communication (RO-REJECT-P) soit à la suite d'un problème d'exécution (RO-REJECT-U).

Ces cinq services sont des services formellement non confirmés (seules les primitives requête et indication sont disponibles); en pratique, les services RESULT ou ERROR fournissent une réponse au service INVOKE.

Les macros OPERATION et ERROR sont projetées (mapping) sur le service ROSE. Elles sont dites utilisateurs de ROSE (ROSE user).

Les macros BIND et UNBIND sont projetées, soit sur le service ACSE, soit sur le service RTSE selon l'architecture retenue.

### 9.3.3.3 Notation

Les tables ci-dessous fournissent une description formelle des quatre macros en syntaxe ASN.1.

Nous ne commenterons que la macro OPERATION qui permet de définir des opérations quelconques entre deux objets.

Le mot clé est OPERATION

L'opération peut être identifiée soit par un entier soit, globalement, par un identificateur d'objet (extension de la syntaxe ASN.1).

Les paramètres nécessaires à l'exécution de l'opération sont typés dans la clause "argument". Les résultats ou erreurs possibles sont aussi spécifiés de même que les opérations enfants utilisables (LINKED).

L'utilisation d'un nommage global par un identificateur d'objets permet d'utiliser la même opération dans plusieurs syntaxes abstraites (plusieurs associations d'applications).

A titre indicatif, nous donnons ci-dessous les caractéristiques des primitives

RO\_INVOKE

RO\_RESULT

Le paramètre "Opération-Value" porte le code de l'opération invoquée (code de l'OPERATION ou éventuellement BIND, UNBIND, ERROR).

"Operation-Class" permet de spécifier la classe (1 à 5, synchrone ou asynchrone avec la réponse attendue)

"Argument" permet de passer les paramètres de l'opération demandée.

"Invoke-ID" identifie la demande et permet d'y relier la réponse correspondante (Result, Error, Reject).

"Linked-ID" identifie, si besoin, l'identificateur de l'opération parent de l'opération demandée.

"Priority" donne la priorité de l'opération demandée.

"Result" fournit le résultat de l'opération exécutée.

### 9.3.4 Eléments de protocole ROSE

A chaque primitive correspond une APDU qui permet de convoyer l'opération demandée ou sa réponse. Les champs de ces APDU sont construits à partir des paramètres de la primitive.

Ainsi, la PDU ROIV est transmise à la suite d'une requête RO\_INVOKE\_Req . Elle comporte les champs Invoke-ID, Linked\_ID, Operation-Value et argument.

Aux autres primitives correspondent les PDU RORS    Result ROER    Error RORJ    Reject

Les automates correspondants sont assez simples. Toutefois pour tenir compte de la diversité des services inférieurs (Présentation ou RTSE) ils ont été découpés en deux sous-automates, correspondant à deux pseudo sous-couches, communiquant par deux primitives TRANS\_Req et TRANS\_Ind qui servent à échanger les RO\_PDU.

Le sous-automate de niveau inférieur est spécifique du service inférieur utilisé . RTSE fournit un transfert de données confirmé et nécessite un échange de jeton de données.

## 9.4 CCR : Commitment, Concurrency and Recovery

- **Validation, annulation et reprise**
- **Engagement, concomitance et rétablissement**
- **Engagement, concurrence et reprise**

Normes OSI :    IS 9804              Norme ITU (CCITT) : X861 – X851 – X852

Les applications réparties qui mettent en œuvre plus d'un processeur autonome, présentent des défis particuliers. Il existe une possibilité de conflit pour des accès concurrents à une ressource par d'autres applications et une possibilité d'échec partiel lorsque un (ou plusieurs) processeur(s) impliqués dans une application répartie échoue(ent) alors que les autres continuent de fonctionner. Ces problèmes sont peu fréquents : la probabilité que plusieurs processus tentent de mettre à jour le même enregistrement dans une base de données au même moment est faible. De même il est improbable que durant un transfert de fonds, un processus veuille enregistrer un dépôt alors qu'un autre échoue à enregistrer un retrait. La plupart des applications s'exécute successivement la plupart du temps. Cependant, ces cas peuvent être très gênant, quoique rares. Les applications doivent prendre des dispositions spéciales lorsque ces cas se produisent :

**Engagement** (ou validation) : pour qu'un processus serveur puisse donner l'assurance qu'il exécutera la tâche demandée sans introduire de difficultés (postérieures)

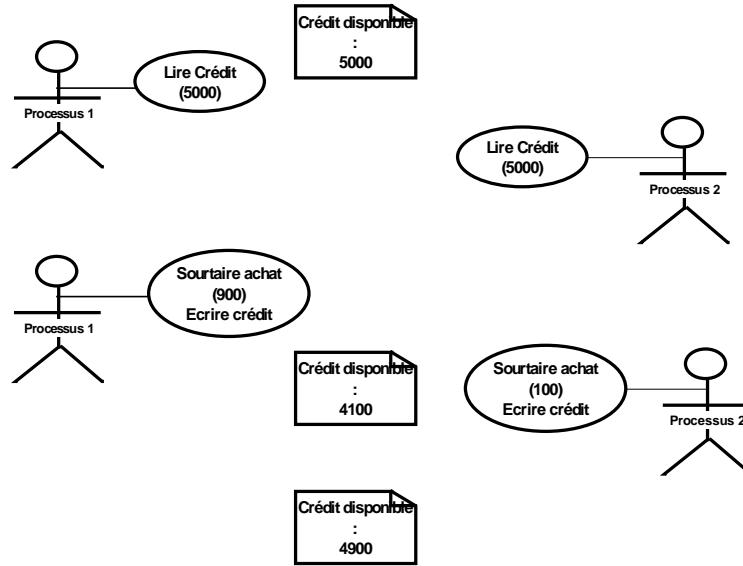
**Concurrence** (ou concomitance) : pour se protéger contre les intrusions dues à des opérations qui se croisent et interfèrent dans l'exécution de la tâche demandée.

**Reprise** (ou rétablissement) : qui établit des procédures permettant de surmonter les incidents et échecs survenus à un ou plusieurs processus durant l'exécution de l'application distribuée.

Le service commun d'application CCR de l'ISO répond à ces fonctionnalités.

### 9.4.1 Concepts et processus

Dans des applications réparties s'exécutant sur un réseau plusieurs processus, ne se connaissant pas les uns les autres peuvent tenter d'utiliser une ressource non partageable au même moment, par exemple pour mettre à jour un enregistrement. L'exemple ci-contre illustre un tel conflit : sur un compte deux sommes différentes sont retirées simultanément, créant une situation financière erronée.



En plus de ce problème de concurrence, les applications réparties présentent d'autres difficultés introduites par le fait que plusieurs processus interagissant sur différents systèmes sont impliqués et que l'un peut tomber en défaut alors que les autres continuent à fonctionner normalement. Que se passe-t-il lorsqu'un processus demande à un autre d'exécuter une activité quelconque et que l'un des autres processus est en panne ? Que se passe-t-il quand il redémarre ?

Le concept central dans une application répartie est la notion **d'action atomique**.

Une action atomique est une séquence d'opérations telle que :

- Soit toutes les opérations soient exécutées avec succès alors qu'aucune autre opération, non reliée, ne s'est intercalée .
- Soit toutes les opérations sont terminées de telle manière que l'état du système et de toutes les données ne garde aucun effet d'une opération quelconque qui a été exécutée.

Toutes les opérations sont réalisées sans interférence ou aucune ne l'est (sans laisser de trace).

Le concept d'action atomique, familier dans les systèmes d'exploitation, est complexe dans les systèmes répartis à cause de la séparation et de l'indépendance des processeurs. Il n'existe pas un processus unique qui commande l'ordonnancement des tâches et l'accès aux données pour tous les processus actifs à travers le réseau. Ce pilotage requiert un effort réparti, partagé entre les processus qui peuvent être inconscients de l'existence des autres processus.

Les processus conçus pour exécuter des actions atomiques sont appelés des **processus transactionnels**. Une **transaction** est initiée par un **client** et exécutée par un **serveur**.

De manière plus précise, une transaction doit respecter les **propriétés ACID** :

- Atomicité
- Consistance
- Isolation
- Durabilité

L'**atomicité** implique que les opérations en rapport dans une transaction soient toutes exécutées ou qu'aucunes ne le soient.

La **consistance** implique que les effets des opérations apparentées dans une transaction soient effectuées correctement, précisément et **avec validité**, en respect avec la sémantique de l'application. Les données manipulées par une transaction passent d'un état consistant à un autre état consistant.

L'**isolation** implique qu'à part les opérations de la transaction considérée, aucune opération ou fonction d'une autre transaction ne puisse accéder aux résultats partiels obtenus. **Cette définition implique que des transactions qui travaillent sur des données communes** (à leur interface) **soient sérialisées**. Cette propriété est assurée par un contrôle de **concurrence** qui garantit qu'une action atomique ne peut être validée tant que d'autres actions atomiques qui en ont changé la valeur ne sont pas validées et qu'aucun changement n'est fait sur une données lors de son utilisation par l'action atomique, excepté par les branches de celle-ci.

La **durabilité** implique que les résultats d'une transaction terminée (complète) ne peuvent en aucune manière être altérés, par exemple en cas de défaut ou de panne. Pour annuler l'effet d'une transaction il faut exécuter une transaction de compensation. Une restauration (Recovery) assure la progression correcte d'une action atomique en cas de panne. Elle est basée sur des mécanismes de synchronisation et de **sauvegarde (log)** des données.

Des **données sûres** sont des données qui survivent à une panne d'une application et sont disponibles à l'entité d'application (du logiciel de communications transactionnel) après que des procédures de reprise locales l'ait restaurée. Des données liées sont des données sûres qui existent durant que la transaction s'exécute. Les données d'une action atomique sont des données sûres qui maintiennent :

- la connaissance qu'une action atomique est en cours.
- l'état du contrôle de concurrence.
- l'information permettant de restaurer toute données dans son état initial.

Une transaction s'exécute en **deux phases** :

- Durant la phase 1, le serveur reçoit une description de l'action atomique (séquence des opérations) et prend les arrangements pour exécuter cette transaction. Le serveur peut exécuter la transaction demandée mais doit garder la possibilité de revenir à l'état précédent la transaction. Il peut aussi enregistrer l'action désirée en attendant l'instruction finale de la réaliser.
- Quand l'action atomique est complètement définie, la phase 2 commence. Durant cette phase 2, le serveur « engage » (commit) l'exécution réussie, achevé, de la transaction si cela est possible ; autrement il avorte la transaction et rend compte de l'échec au client.

Une fois que le serveur est engagé, il doit terminer les opérations demandées, dans tous les cas quelles que soient les problèmes rencontrés. Si le serveur tombe en panne durant l'exécution des opérations qui constituent la transaction, il doit reprendre le traitement au redémarrage. Pour cela, les transactions doivent être enregistrées au préalable: c'est l'opération de **journalisation**.

Quand le serveur a engagé l'exécution de la transaction, ni lui, ni le client ne peuvent faire avorter la transaction.

## 9.4.2 Service CCR

Ce service peut être découpé en trois sous-ensemble :

### Description/Représentation du service :

Les services C-BEGIN et C-PREPARE marquent le début et la fin d'une séquences d'opérations qui doivent être traitées comme une transaction simple.

C-BEGIN est un service qui peut être confirmé ou non et comporte les 4 primitives (Req, Ind, Resp, Conf). Une application qui initie une transaction fait suivre C-BEGIN d'une liste d'opérations auxquelles elle souhaite que le serveur répond. Dans certaines applications cette liste est clairement indiquée. Sinon l'application envoie une requête C-PREPARE pour indiquer la fin de la définition de la transaction et demander au répondeur d'indiquer sa capacité à s'engager pour compléter la transaction.

### 9.4.2.1 Réponse à une demande pour une Transaction

Le serveur utilise les services C-READY ou C-REFUSE pour accepter ou refuser de participer à une transaction. Par C-READY, il se déclare capable de *compléter la transaction mais aussi de retourner aux conditions* qui précédaient la transaction. Le serveur attend le message final du client avant d'effectuer la transaction ou de retourner à l'état précédent.

C-REFUSE indique que le serveur ne peut pas exécuter la transaction et qu'il n'y aura aucun changement de son coté. Il n'attend pas de suite à ce message.

### 9.4.2.2 Compléter ou Annuler la transaction

C-COMMIT et C-ROLLBACK sont des services confirmés qui fournissent au client la possibilité de s'engager à compléter la transaction ou de l'annuler et de revenir à l'état précédent le service C-BEGIN. La décision du client est prise d'après les informations reçues du serveur.

C-RESTART fournit la facilité de tenir compte de plusieurs types d'incidents. Il peut être utilisée par un serveur n'importe quand avant l'utilisation de C-READY ou par le client si il observe un incident du système de communication ou s'il n'est pas sûr de l'état du serveur. Par les paramètres de C-RESTART le client peut demander au serveur de reprendre au début de l'action atomique ou au dernier point d'engagement (commit) ou de reprise (rollback). Ceci est particulièrement utile pour des transactions mettant en jeu plusieurs serveurs (Voir protocole OSI-TP)

## 9.4.3 Protocole CCR

Le protocole CCR est utilisé pour *convoyer* des information de coordination entre un client et un serveur. Toutes les actions correspondant aux transactions sont traitées localement dans les processus client et serveur. Le protocole CCR est donc simple : il achemine simplement les informations indiquées par l'utilisateur. Pour cela il repose sur le service Présentation pour définir son contexte de transfert de données et pour atteindre le service Session qui établit les *points de synchronisation*. C- PREPARE et C-READY peuvent utiliser les données typées (P-TYPED-DATA et S-TYPED-DATA) pour échanger des informations sans être contraints par la détention d'un jeton de données qui est lié au jeton de synchronisation (Voir Session)

## 9.5 X500 : Service d'Annuaire

Un *service d'annuaire* (Directory) permet d'associer une adresse à un nom de ressource ou d'utilisateur ainsi qu'un certain nombre d'autres attributs qui les caractérisent. Dans les systèmes répartis il est vite apparu qu'il était plus fiable et moins redondant de permettre à des systèmes hétérogènes d'interroger leurs annuaires particuliers, voire de les faire coopérer. Il s'agit en pratique d'une application de base de données avec des contraintes de mise à jour assez faibles (volumes restreints).

L'ISO et le CCITT ont normalisé conjointement un système d'annuaire spécifié par le CCITT dans la série X500.

### 9.5.1 Composants de l'annuaire X500

L'annuaire (Directory) est une collection de systèmes ouverts qui coopèrent pour maintenir une **base de données** logique d'informations portant sur un **ensemble d'objets du monde**

**réel**. Les *utilisateurs*, hommes **et** programmes d'ordinateurs, peuvent lire ou modifier l'information, ou une de ses parties, s'ils en ont la permission. Chaque utilisateur est représenté par une Agent Utilisateur de l'Annuaire: DUA (Directory User Agent). Celui-ci permet d'utiliser l'annuaire par un *point d'accès*. L'information dans la base est collectivement connue comme la Base d'Information de l'Annuaire: DIB (Directory Information Base).

L'annuaire X500 fournit à ses utilisateurs un service abstrait, ensemble de facilités d'accès bien définies.

La base de données (DIB) est constituée d'informations sur des objets. Elle est composée d'**entrées** (d'annuaire) dont chacune consiste en un ensemble de données sur *un objet*. Ces entrées sont organisées sous forme d'un arbre (DIT: Directory Information Tree). Chaque entrée a un identificateur (*distinguished name*) constitué à partir de l'identificateur de son supérieur dans l'arbre.

## 9.5.2 Description du service

On peut le décomposer en quatre parties:

### 9.5.2.1 Service de qualification

Il comporte des commandes (**requêtes**) pour imposer des limites à l'usage des ressources. Chaque requête peut être accompagnée de **paramètres de sécurité** pour la protection des informations (*signature numérique*). Un service d'*authentification forte* peut aussi être fourni. Ces requêtes peuvent être appliquées à plusieurs entrées grâce à des *filtres*.

### 9.5.2.2 Service d'interrogation de l'annuaire

Il comporte des fonctions de lecture d'une entrée, de comparaison d'un attribut particulier, de listage des subordonnés d'une entrée, de recherche d'information pour toutes les entrées qui répondent à un filtre, par exemple pour fournir un service de type "**pages jaunes**". Il permet aussi d'abandonner une requête en cours.

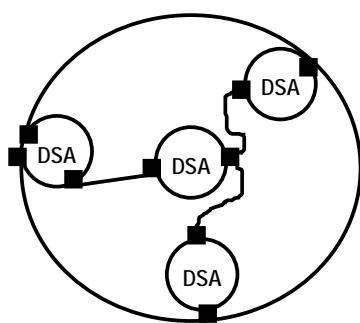
### 9.5.2.3 Service de modification de l'annuaire

Il permet d'ajouter ou de retrancher une entrée ou d'en modifier les attributs (ajout, retrait, modification), modification en particulier de l'identificateur.

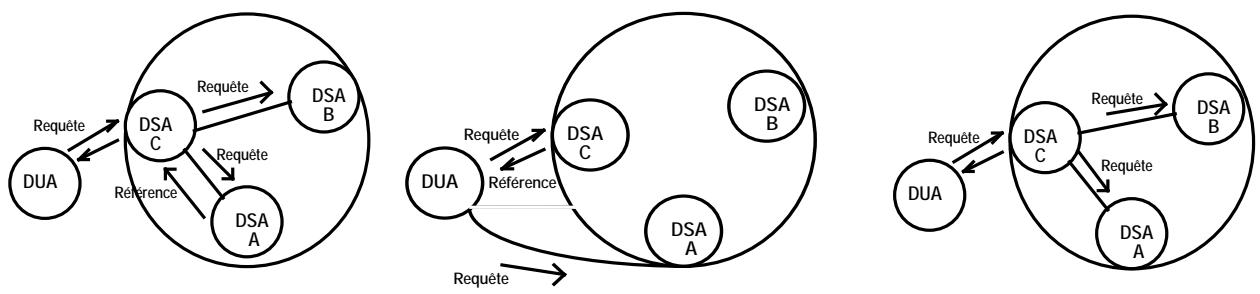
### 9.5.2.4 Autres fonctions

Si un service tombe en défaut une indication d'erreur est transmise. Un service peut ne pas aboutir parce que l'agent ne peut transmettre sa requête au point d'accès approprié. Dans ce cas l'annuaire peut **renvoyer une référence** (*referral*) qui suggère un point d'accès alternatif. Il peut aussi **chaîner la requête**.

## 9.5.3 Annuaire distribué



La base de données (DIB) peut être répartie sur des systèmes interconnectés. Les bases locales sont entièrement dépendantes de l'implantation. Chacune est accessible par l'intermédiaire d'un *agent du système d'annuaire* (DSA: Directory System Agent). Un DSA peut utiliser des informations de sa base locale ou interagir avec d'autres DSA pour transmettre les requêtes. Il peut utiliser les renvois de références (*referral*) ou le chaînage. Il peut aussi diffuser les requêtes.



---

## 10MESSAGERIE X400 (COURRIER ELECTRONIQUE)

### 10.1 Présentation

#### 10.1.1 Introduction

Il existe actuellement des messageries sur des systèmes isolés. Des personnes sur des sites variés voulant communiquer par courrier électronique doivent pouvoir être reliés directement à toutes ces messageries . D'autres part, souvent, on ne peut leur transmettre des messages que lorsque leur propre système de messagerie est en fonctionnement (nécessité d'une connexion établie lors de l'envoi d'un message). Ceci restreint l'utilisation du service et en rend les coûts élevés.

La solution à ce problème réside dans l'**interconnexion des systèmes de messagerie**.

Les messages des utilisateurs sont :

- déposés dans la messagerie locale
- acheminés vers la (ou les) messageries(s) distantes où sont connectés le (ou les) utilisateur(s) destinataire(s)
- délivrés aux destinataire

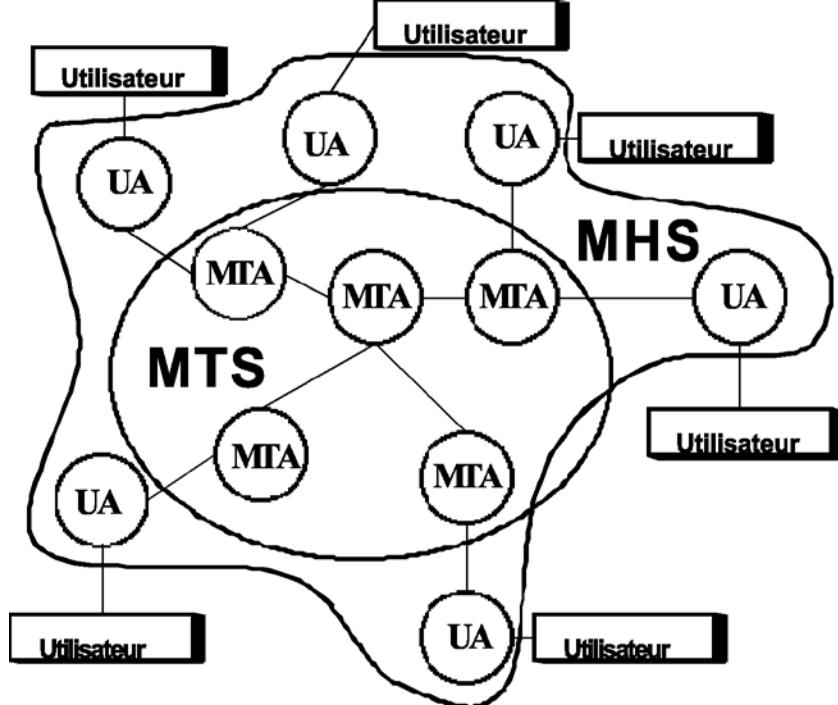
#### 10.1.2 Modèle fonctionnel CCITT

Dans ce modèle on distingue 3 entités **fonctionnelles** :

- **utilisateur**: émetteur effectif ou récepteur final du message. C'est un usager de la messagerie ou un programme.
- **agent de transfert de message (ATM)**: Il assure avec éventuellement d'autres ATM, l'acheminement ou routage et le transfert des messages. L'ensemble des ATM constitue un réseau **STM: Système de Transfert de Messages**.
- **agent utilisateur (UA)**: Il représente l'usager auprès d'un ATM. Il modélise :
  - une boîte à lettres
  - le logiciel de dialogue qui assiste l'usager pour préparer, émettre et recevoir les messages. Entre UA et MTA on distingue **deux interactions** fondamentales : le **dépôt** et la **remise**.

Ce modèle est illustré par le schéma ci-dessous qui fait apparaître ses 2 niveaux:

- MHS : système de messagerie
- MTS : système de transfert de messages.



Système de Messagerie

Ce système utilise un **mécanisme "dépôt"**

- **acquittement**
- **remise**" décrit ici.
- utilisateur
- prépare un message
- demande à son UA de l'émettre vers un ou plusieurs destinataires
- UA émetteur
- **dépose** auprès de son MTA le message à destination des UA des destinataires
- MTA d'entrée
- **acquitte** ce dépôt (si la transaction est correcte) ; cet acquittement contrôle seulement la syntaxe du dépôt, la validité de l'abonnement au service, etc. mais *pas* l'existence des UA récepteurs.
- transfère le message vers un ou plusieurs ATM selon ses propres conventions de **routage** ( duplication éventuelle du message ; cette duplication est faite le plus tard possible).
- MTA final
- remet le message à l'UA récepteur
- UA récepteur
- remet le message au destinataire

En **service optionnel**, le SMT peut générer et transmettre des **avis de remise** ou des **avis de non-remise** selon que le message a pu être distribué par le MTA final à l'UA récepteur ou que le message n'a pu être transmis

### 10.1.3 Classes d'agents utilisateurs

Les UA peuvent être regroupées en familles ayant des besoins spécifiques communs, par exemple : messagerie inter-personnes (IPMS, courrier électronique), transfert de fichiers soumission de travaux. Ces classes d'UA existent ou sont en cours de conception.

D'autres seront étudiées, qui utiliseront le même service de Transfert de Messages. D'autres interfaces, les unités d'accès (AU) permettent d'accéder à d'autres services par exemple un service télématique ou le service postal (impression des messages)

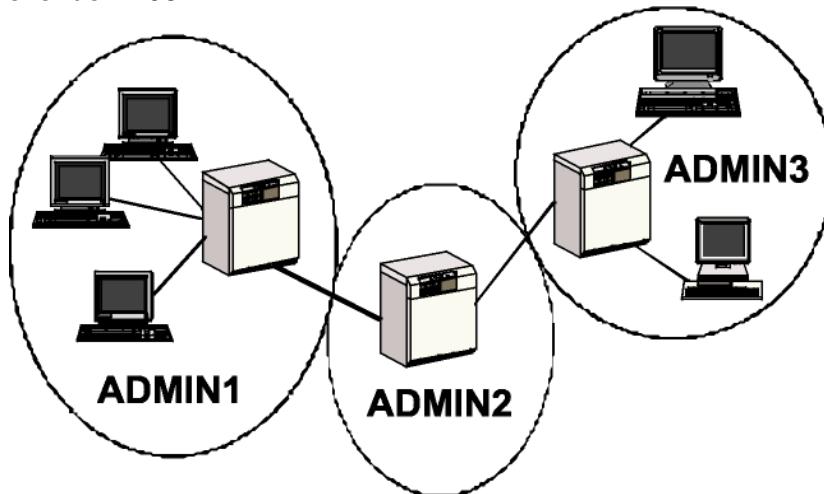
### 10.1.4 Normalisation

La normalisation CCITT X400 comporte actuellement 9 recommandations :

- X400: Principes du système et du service
- X402: Architecture globale
- X411: Service de Transfert de Messages
- X419: Spécification de protocoles
- X420: Messagerie de personne à personne
- X403: Essais de conformité
- X407: Conventions pour les services abstraits
- X209: Règles de conversions de codes (ex X408)
- X413: Service abstrait d'enregistrement des messages.

Elles correspondent à la norme OSI 10021-x.

Certaines recommandations du Livre Rouge du CCITT ont été déplacées, par exemple la syntaxe ASN1/X409 qui est devenue X208.



### 10.1.5 Adressage

Pour assurer le transfert et le routage des messages , il est nécessaire d'établir un **plan d'adressage des UA**. Ces adresses seront consignées dans des **annuaires**; ceux-ci peuvent suivre le standard X500.

Le nom descriptif d'un UA est appelé **nom d'O/R (Origine-Remise)**. Il comporte un certain nombre d'attributs permettant de désigner de manière non-ambigüe un expéditeur ou un destinataire.

destinataire de messages (nom, prénom, fonction, adresse, numéro X121 par exemple). On appelle **adresse d'O/R** un type de nom d'O/R qui contient des caractéristiques aidant le MTS à situer le point de raccordement de l'UA correspondante.

Le **Service de transfert de messages (MTS)** est partagé en "domaines de gestion administratifs ou privés" (ADMD; PRMD) modélisés comme un **Agent de Transfert de Messages (MTA)** unique, même s'il est réalisé par plusieurs MTA.

Un utilisateur est rattaché à **un seul UA**.

Une **adresse de messagerie** est une adresse d'O/R particulière comportant un nom de domaine de gestion administratif, un nom de pays et un ensemble d'attributs d'utilisateur.

#### 10.1.5.1 Nom d'O/R :

Le nom descriptif de chaque utilisateur est attribué à son UA. Il est appelé **nom d'O/R**. Il consiste en une séquence étiquetée composée d'une liste d'attributs standards et d'une liste (optionnelle) d'attributs spécifiques au domaine.

La liste d'attributs standards est une séquence de 9 éléments, tous optionnels, mais dont on doit trouver un sous-ensemble suffisant. Ce sont :

- nom de pays
- nom de domaine administratif
- adresse X121 (par exemple adresse Transpac) identificateur de terminal (télex, télécopie) nom de domaine privé nom d'organisation
- identificateur unique d'UA nom de personne
- SEQUENCE OF ( noms d'unité organisationnelle )

La séquence suivante est un exemple de nom d'O/R (adresse Atlas 400)

X400: /C=FR /A=ATLAS /P=PAPYetFils /O=LYON /OU=Syst1 /S=PDupont /C : Pays /A : Domaine d'administration public /P: domaine d'administration privé /O : Organisation /OU : Unité d'organisation /S : Nom (prénom, initiales, généalogie)

La forme 2 d'adresse comporte l'adresse X121 et en option un identificateur de terminal

La forme 1 a trois variantes. Elle comporte toujours le nom de pays et le nom de domaine administratif plus un autre attribut au moins : adresse X121 , identificateur numérique unique d'UA ou un autre attribut parmi les cinq restants.

#### 10.1.5.2 Annuaire :

Certaines spécifications fonctionnelles des annuaires ont été déterminées; elles correspondent aux fonctions ci-dessous:

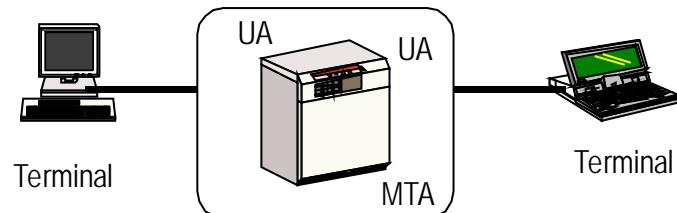
- vérifier l'existence d'un nom d'O/R

- retourner l'adresse d'O/R qui correspond au nom d'O/R présenté
- déterminer si le nom d'O/R présenté correspond à un destinataire ou une liste de distribution
- retourner la liste des membres d'une liste de distribution
- retourner en réponse à un nom partiel, la liste des noms d'O/R possibles
- permettre aux utilisateurs de parcourir les entrées des annuaires dans l'ordre ou sélectivement
- indiquer les capacités de l'entité auquel se réfère le nom d'O/R
- assurer les fonctions de mise à jour de l'annuaire. Ces annuaires doivent avoir des qualités de facilité d'utilisation, souplesse, disponibilité, extensibilité et fiabilité.

### 10.1.6 Systèmes physiques

Un même système physique peut contenir :

- Un MTA et une collection d'UA
- Un MTA seul
- Un UA seul



Rq : Il est possible d'éclater un UA sur deux systèmes physiques dont l'un comporte un MTA.

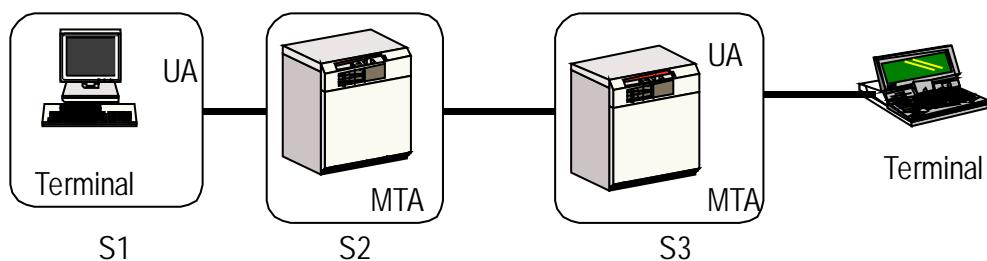
Ce schéma illustre les configurations de base :

- S1
- fournit toutes les fonctionnalités de l'UA selon un protocole particulier.

Il est réalisé par un terminal intelligent ou un micro (ou mini) ordinateur.

- transfert les messages par le système S2

- S2 fait office de **Centre de transit**, utile dans les réseaux complexes.
- S3 est un système à UA et MTA corésidents. L'usager dispose d'un terminal simple visu ou videotex.



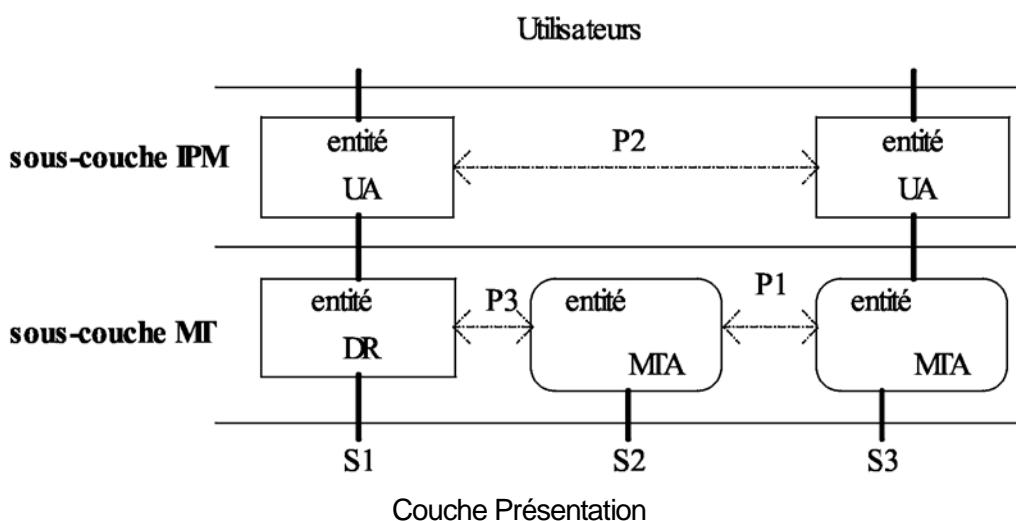
## 10.2 Services.

### 10.2.1 Décomposition

Ce modèle fonctionnel suggère de décomposer la couche Application en **2 sous couches** :

- service **TRANSFERT de MESSAGES**
- service **COURRIER ELECTRONIQUE (IPM)**

Pour implanter d'autres applications, il suffira de changer ce service IPM seulement. Cette structure est illustrée sur le schéma ci-dessous.



Dans le système S1, l'entité UA s'appuie sur une entité DR (Dépôt/Remise) qui assure l'interface avec le MTA de S2. Une variante consiste à ne pas implanter cette entité dépôt/remise mais à éclater l'entité UA sur les systèmes S1 et S2 en utilisant un protocole spécifique de type arythmique entre les deux sous-ensembles. Dans ce cas le système S1 n'a pas besoin d'être toujours connecté à S2. Ces sous-couches sont normalisées dans des Recommandations du CCITT différentes :

- X411: " Service Transfert de messages " décrit les services et protocoles des entités DR et MTA
- X420: " Service Messagerie de personne à personne " décrit le service et le protocole IPM
- XXXX: " Opérations distantes et Serveur de transfert fiable " décrit les **Opérations distantes** (notions utilisées par exemple dans le protocole P3) et le **Serveur de transfert fiable :RTS** (qui met en oeuvre, par exemple, les protocoles P1 et P3) du système de messagerie MHS. P1 et P3 sont décrits dans X411.

D'autres protocoles d'accès au MTS ont été spécifiés (par exemple P12)

### **10.2.2 Opérations distantes :**

Certains protocoles d'Application sont intrinsèquement des protocoles d'**interaction** qui permettent à une entité de demander à une entité distante l'exécution d'une certaine opération. L'autre entité essaie d'exécuter cette opération et rend compte du résultat. Pour cela les entités échangent des PDU particulières : Unités de données de protocole d'Opération (**OPDU**). Elles permettent de lancer l'opération, d'indiquer si le résultat en est positif ou négatif ou si l'entité distante a refusé de l'exécuter. Ces mécanismes sont illustrés sur les figures ci-dessous extraites de la recommandation X410. Cette partie du protocole X410 est un sous-ensemble du protocole correspondant au service d'opérations distantes (ROSE) de l'OSI .

### **10.2.3 Serveur de transfert fiable.**

Le Serveur de transfert fiable: RTS est la partie d'une entité d'Application responsable de la création et du maintien des associations entre cette entité et ses homologues et du transfert fiable de PDU d'Application à l'aide de ces associations.

Une association est soit unidirectionnelle soit bidirectionnelle à l'alternat . Dans ce cas les échanges sont régis par le "tour" qui passe d'une entité à l'autre.

Ce Serveur est pris en charge par les services OSI de Présentation (peu) et de Session (beaucoup). Il offre des primitives de :

- ouverture d'association
- fermeture d'association confirmées
- demande de tour
- cession de tour
- transfert
- signalisation d'anomalies.

Il requiert du Service Session les Unités fonctionnelles Noyau, signalisation d'anomalies, gestion d'activité, demi-duplex et synchronisation mineure.

Le type Opération est défini par la macro suivante:

**OPERATION MACRO :: -**

BEGIN

**TYPE NOTATION**

:: = "ARGUMENT" Type-Nommé Résultat Erreurs | vide

**VALUE NOTATION**

:: = valeur (VALUE INTEGER)

**Résultat**

:: = vide | "RESULT" Type-Nommé

**Erreurs**

:: = vide | "ERRORS" [Noms-Erreur]

**Type-Nommé**

:: = identificateur type | type

**Noms-Erreurs**

:: = vide | liste-Identificateurs

**Liste-Identificateurs**

:: = identificateur | Liste-Identificateurs "," identificateur

END

Cette définition permet de constater que la représentation d'une opération est celle d'un Entier.

*Exemple* — Si l'Opération qui énumère les fichiers d'un répertoire distant est du type suivant:

**OPERATION**

**ARGUMENT SET** |

**Nom-Répertoire Chaîne-IAS OPTIONAL**

      — Prend la valeur par défaut spécifiée par annuaire — Par-Défaut

      clé-Tri INTEGER [nom-Fichier(0), date-Modification(1)} DEFAULT nom-Fichier}

**RESULT nom-Fichier SEQUENCE OF Chaîne-IAS**

**ERRORS répertoire-inexistant, accès-Refusé**

et à la valeur 7, elle peut être codée comme montré ci-dessous. Dans cet exemple, l'opération rend toujours compte de son résultat:

Entier	Longueur	Contenu
02	01	07

Le type Erreur est défini par la macro suivante:

**ERROR MACRO :: -**

BEGIN

**TYPE NOTATION** :: = "PARAMETER" Type-Nommé | vide

**VALUE NOTATION** :: = valeur (VALUE INTEGER)

**Type-Nommé** :: = identificateur type | type

END

Description  
formelle de  
l'opération:

énumération-Fichiers OPERATION  
ARGUMENT SET |  
nom-Répertoire Chaîne-IAS OPTIONAL,  
clé-Tri INTEGER {nom-Fichier(0), date-Modification(1)}  
DEFAULT nom-Fichier  
RESULT noms-Fichiers SEQUENCE OF Chaîne-IAS  
ERRORS {répertoire-Non-Existant\*, accès-Refusé}  
:: = 7

\* Cette erreur n'est pas décrite ci-dessous: elle est traitée de façon analogue à  
accès-Refusé.



Description  
informelle  
équivalente de  
l'opération:

- ① ● Argument-Enumération-Fichiers :: = SET |  
nom-Répertoire Chaîne-IAS OPTIONAL,  
clé-Tri INTEGER {nom-Fichier(0), date-Modification(1)}  
DEFAULT nom-Fichier
- ② ● Résultat-Enumération-Fichiers :: = SEQUENCE OF Chaîne-IAS  
-- noms de fichiers
- ③ ● Code d'opération énumération-Fichiers: 7
  - Il est rendu compte des résultats positifs et négatifs
  - Erreurs signalées: répertoire-Inexistant et accès-Refusé.



Unités de données de  
protocole de:

- l'opération
- du résultat positif
- du résultat négatif

OPDU Lancement:  
[1] SEQUENCE {ID-Lancement INTEGER, OPERATION, argument ANY} ③ ①  
OPDU Résultat-Positif:  
[2] SEQUENCE {ID-Lancement INTEGER, résultat ANY} ②  
OPDU Résultat-Négatif:  
[3] SEQUENCE {ID-Lancement INTEGER, ERROR, paramètre ANY} ④

Description  
informelle  
équivalente  
de l'erreur:

- ④ ● Paramètre-Accès-Refusé :: =  
BIT STRING {aucun(0), lecture(1), ajout(2), modification(3)}  
-- accès accordé
- ⑤ ● code d'erreur accès-Refusé: 3



Description  
formelle  
de l'erreur:

accès-Refusé ERROR  
PARAMETER accès-Accordé BIT STRING |  
aucun(0), lecture(1), ajout(2), modification(3)|  
:: = 3

FIGURE 1/X.410

Relation entre les types de données d'opération et d'erreur et les OPDU  
(Opération avec compte rendu du résultat)

Description formelle de l'opération:

**répertoire-Par-Défaut OPERATION**  
**ARGUMENT nom-Répertoire Chaîne-IAS**  
:: = 6



Description formelle équivalente de l'opération:

- ① ● Argument-Répertoire-Par-Défaut :: = Chaîne-IAS -- nom de répertoire
- ② ● Code d'opération Répertoire-Par-Défaut: 6
- Il n'est rendu compte ni du résultat positif, ni du résultat négatif.



Unité de données de protocole de:

- lancement de l'opération

OPDU Lancement:  
[1] SEQUENCE {ID-Lancement INTEGER, OPERATION, argument ANY}

②

①

FIGURE 2/X.410

Relation entre le type de données opération et les OPDU  
(L'opération ne rend pas compte de son résultat)

## 10.3 Protocoles

A chaque sous-couche correspond une variété de protocoles destinés soit à assurer le transfert des messages soit une composition adaptée au type d'application. Nous ne donnerons qu'une description sommaire des protocole P1, P2 et P3. Beaucoup d'autres sont à l'étude. Tous ces protocoles se distinguent par

- Un faible nombre de types de PDU
- Des structures de PDU complexes avec un grand nombre de champs optionnels
- Des enchaînements simples.

Ils donnent lieu à des réalisations de taille assez modeste. Cependant l'interface utilisateur (non normalisé), pour être puissant et facile d'emploi, complique les implantations réelles.

### 10.3.1 Structures des APDU :

Les APDU présentées à l'interface avec les couches inférieures sont composées d'une **enveloppe** et d'un **contenu**. Ce contenu, qui en constitue la zone de données, est construit dans la sous-couche supérieure . Il comporte une **en-tête** et un **corps**. Ce corps correspond aux données utilisateur. Cette structure est illustrée sur le schéma suivant.

L'enveloppe est décrite dans la Recommandation X411 L'en-tête est décrite dans la Recommandation X420

Enveloppe	Contenu	
	En-tête	Corps (données)

### 10.3.2 Service et Protocole (P1) Transfert de Messages :

Ce service fournit aux entités Agent Utilisateur (UAE) le moyen de transférer des messages dans un délai fini en effectuant, si nécessaire, des conversions de type de codage sur leur contenu. Ses primitives permettent d'assurer:

- L'établissement et la terminaison d'un dialogue UAE - couche MT (MTL)
  - La modification par l'UAE des valeurs de ses paramètres d'enregistrement gérés par la couche MT
  - La commande temporaire par l'UAE des types et longueurs des messages que la MTL peut lui envoyer
- Le dépôt par une UAE d'un message à envoyer à un ou plusieurs destinataires
- La détermination qu'un message pourrait être remis à un ou plusieurs destinataires
- (Essai)
- La remise d'un message à une UAE spécifiée par la MTL
  - La fourniture d'un avis de non-remise si un message n'a pu être délivré à une UAE destinataire
- La fourniture d'un avis de remise si l'UAE expéditrice l'a demandée
  - La possibilité de demander l'annulation d'un message transmis avec demande de remise différée.

Pour cela les services élémentaires suivants sont fournis :

- Activation
- Désactivation
- Enregistrement
- Directive
- **Dépôt**
- **Essai**
- **Remise**
- Avis
- Annulation
- Changement de mot de passe

Le protocole correspondant utilise 3 MPDU :

- |                       |                        |
|-----------------------|------------------------|
| • MPDU-Utilisateur    | Contient une APDU      |
| • MPDU-Rapport-Remise | Rapport de remise      |
| • MPDU-Essai          | Test de l'acheminement |

Une description formelle de ces MPDU est donnée ci-dessous. Elle indique tous les champs possibles et leurs caractéristiques.

L'enveloppe de la MPDU-Utilisateur contient par exemple les paramètres suivants:

- nom de l'émetteur
- identifiant du message pour tous les destinataires
- type
- priorité
- publication de la liste des récepteurs (O/N)
- récepteur de "secours" (O/N) par destinataire

- demande d'accusé de remise (O/N) paramètres de gestion
- nom du domaine de gestion
- date d'entrée
- action du domaine (transféré, rerouté, etc.)

Le contenu est une UAPDU transmis par la sous-couche supérieure.

La MPDU-Rapport-Remise est plus simple et peut comporter les paramètres

- enveloppe
  - nom de l'émetteur
  - identifiant du rapport
  - paramètres de gestion (trace) : contenu
  - identifiant du message concerné
  - information de trace sur ce message (option)
  - information de trace finale
- heure de remise
- ou
- raison de non-remise

### **10.3.3 Service (IPM) et Protocole (P2) Agent Utilisateur:**

Dans cette sous-couche on peut trouver différents Agents Utilisateurs. Nous ne décrirons rapidement que l'UA de messagerie de personne à personne.

Le Service de Messagerie de personne à personne (IPM) donne à ses utilisateurs la possibilité de communiquer en envoyant et en recevant des messages sans avoir besoin d'établir au préalable des connexions. Les services élémentaires suivants sont fournis:

- Accès au Service de Transfert de Messages
- Expédition et réception de messages IP
- Fourniture d'éléments de service de dépôt, remise, et de conversion :
  - urgence de remise; destinataires multiples
  - conversion explicite ou non-conversion
  - conversion implicite
  - remise différée; non-communication de non-remise
  - destinataire suppléant autorisé
  - avis de remise
  - renvoi de contenu
    - Fourniture d'éléments de service d'actions entre UA
  - indication de destinataires de copie muette
  - avis de non-réception
  - avis de réception
  - indication de retransmission automatique
    - Fourniture d'éléments de service d'information sur UA

- information message par message "
- Indication de demande de réponse
- corps à plusieurs parties
  - Annulation de remise différée
  - Essai
  - Conservation pour remise ultérieure
  - Désignation de destinataire supplémentaire.

Le protocole P2 correspondant utilise les UAPDU suivantes:

- UAPDU-IM Message
- UAPDU-SR Rapport de remise

Une description formelle de ces UAPDU est donnée ci-dessous.

L'en-tête de l'UAPDU-IM (message de personne à personne) peut, par exemple, comporter les éléments suivants dont la majeure partie est optionnelle :

- identifiant de message - - émetteur
- visas
  - récepteurs en copie
  - récepteurs
- en réponse à ...
- en remplacement de ...
- en référence à ... (des messages)
- **sujet**
- répondre avant (date)
- répondre à ...
- importance pour chaque récepteur
- accusé de réception demandé (O/N)
- réponse demandée (O/N)

Le corps de cette UAPDU-IM contient le message de l'utilisateur L'UAPDU-SR ne comporte pas de corps. Son en-tête a les éléments suivants:

- identifiant de message
- information de réception : date ou
- information de non-réception : raison et commentaires

#### **10.3.4 Autres services**

D'autres services sont à l'étude qui utilisent le service de messagerie fiable. Ils mettent en jeu des Agents Utilisateurs spécifiques . On citera :

TFMM : Transfert de fichiers en mode messagerie

## STMM : Soumission de travaux en mode messagerie

**P1 DÉFINITIONS :: =**

**BEGIN**

-- P1 fait usage des types définis dans le module ci-après

-- T73: Recommandation T.73

**MPDU** :: = CHOICE [0] IMPLICIT MPDU-Utilisateur, MPDU-Service;

**MPDU-Service** :: = CHOICE [1] IMPLICIT MPDU-Rapport-Remise,  
[2] IMPLICIT MPDU-Essai;

**MPDU-Utilisateur** :: = SEQUENCE {Enveloppe-UMPDU, Contenu-UMPDU};

**Enveloppe-UMPDU** :: = SET {  
Identificateur-MPDU,  
expéditeur Nom-OR,  
initiaux Types-Codage OPTIONAL,  
Type-Contenu,  
ID-UA-Contenu OPTIONAL,  
Priorité DEFAULT normal,  
Indicateur-Message DEFAULT {},  
remise-Différée [0] IMPLICIT Date-Heure OPTIONAL,  
[1] IMPLICIT SEQUENCE OF Info-Bilatérales-Domaine OPTIONAL,  
[2] IMPLICIT SEQUENCE OF Info-Destinataire,  
Informations-Trace};

**Contenu-UMPDU** :: = OCTET STRING

-- date et heure

**Date-Heure** :: = Heure-UTC

-- informations diverses de l'enveloppe

**Identificateur-MPDU** :: = [APPLICATION 4] IMPLICIT SEQUENCE {  
Identificateur-Global-Domaine, Chaîne-IAS}

**Type-Contenu** :: = [APPLICATION 6] IMPLICIT INTEGER {p2(2)}

**ID-UA-Contenu** :: = [APPLICATION 10] IMPLICIT Chaîne-Imprimable

**Priorité** :: = [APPLICATION 7] IMPLICIT INTEGER {  
normal(0), non-Urgent(1), urgent(2)}

**Indicateur-Message** :: = [APPLICATION 8] IMPLICIT BIT STRING {  
divulgation-Destinataires(0),  
conversion-Interdite(1),  
destinataire-Supplément-Autorisé(2),  
demande-Renvoi-Contenu(3)}

-- Informations bilatérales propres au domaine

**Info-Bilatérales-Domaine** :: = SEQUENCE {  
Nom-Pays,  
Nom-Domaine-Administratif,  
Info-Bilatérales};

FIGURE 16/X.411 (partie 1 de 4)

Définition formelle de la MPDU UTILISATEUR

*Suite des définitions P1*

**Info-Bilatérales** :: = ANY  
-- Informations sur le destinataire

**Info-Destinataire** :: = SET {  
destinataire Nom-OR,  
[0] IMPLICIT Identificateur-Complémentaire,  
[1] IMPLICIT Indicateur-Destinataire,  
[2] IMPLICIT Conversion-Explicite OPTIONAL}

**Identificateur-Complémentaire** :: = INTEGER

**Indicateur-Destinataire** :: = BIT STRING -- Voir la figure 19/X.411

**Conversion-Explicite** :: = INTEGER {texte-iA5-Télétex(0), télétex-Télex(1)}

-- Informations de trace

**Informations-Trace** :: = [APPLICATION 9] IMPLICIT SEQUENCE OF  
SEQUENCE {Identificateur-Global-Domaine,  
Info-Fournies-Domaine}

**Info-Fournies-Domaine** :: = SET {  
arrivée [0] IMPLICIT Date-Heure,  
différée [1] IMPLICIT Date-Heure OPTIONAL,  
action [2] IMPLICIT INTEGER{relayé(0), rerouté(1)},  
converti Types-Codage OPTIONAL,  
précédent Identificateur-Global-Domaine OPTIONAL}

-- Identificateur global de domaine

**Identificateur-Global-Domaine** :: = [APPLICATION 3] IMPLICIT SEQUENCE {  
Nom-Pays,  
Nom-Domaine-Administratif,  
Identificateur-Domaine-Privé OPTIONAL}

**Nom-Pays** :: = [APPLICATION 1] CHOICE {  
Chaîne-Numérique,  
Chaîne-Imprimable}

**Nom-Domaine-Administratif** :: = [APPLICATION 2] CHOICE {  
Chaîne-Numérique,  
Chaîne-Imprimable}

**Identificateur-Domaine-Privé** :: = CHOICE {  
Chaîne-Numérique,  
Chaîne-Imprimable}

-- Nom d'O/R

**Nom-OR** :: = [APPLICATION 0] IMPLICIT SEQUENCE {  
Liste-Attributs-Standard,  
Liste-Attributs-Définis-Domaine OPTIONAL}

---

---

FIGURE 16/X.411 (partie 2 de 4)

Définition formelle de la MPDU UTILISATEUR

*Suite des definitions P1*

**Liste-Attributs-Standard** :: = **SEQUENCE** {  
    **Nom-Pays OPTIONAL,**  
    **Nom-Domaine-Administratif OPTIONAL,**  
    **[0] IMPLICIT Adresse-X121 OPTIONAL,**  
    **[1] IMPLICIT ID-Terminal OPTIONAL,**  
    **[2] Nom-Domaine-Privé OPTIONAL,**  
    **[3] IMPLICIT Nom-Organisation OPTIONAL,**  
    **[4] IMPLICIT Identificateur-Unique-UA OPTIONAL,**  
    **[5] IMPLICIT Nom-Personne OPTIONAL,**  
    **[6] IMPLICIT SEQUENCE OF Unité-Organisationnelle OPTIONAL;**  
**Liste-Attributs-Définis-Domaine** :: = **SEQUENCE OF Attributs-Définis-Domaine**  
**Attributs-Définis-Domaine** :: = **SEQUENCE** {  
    **type Chaîne-Imprimable,**  
    **valeur Chaîne-Imprimable}**  
**Adresse-X121** :: = **Chaîne-Numérique**  
**ID-Terminal** :: = **Chaîne-Imprimable**  
**Nom-Organisation** :: = **Chaîne-Imprimable**  
**Identificateur-Unique-UA** :: = **Chaîne-Numérique**  
**Nom-Personne** :: = **ENSEMBLE** {  
    **nom [0] IMPLICIT Chaîne-Imprimable,**  
    **prénom [1] IMPLICIT Chaîne-Imprimable OPTIONAL,**  
    **initiales [2] IMPLICIT Chaîne-Imprimable OPTIONAL,**  
    **qualificateur-Généalogique [3] IMPLICIT Chaîne-Imprimable OPTIONAL;**  
**Unité-Organisationnelle** :: = **Chaîne-Imprimable**  
**Nom-Domaine-Privé** :: = **CHOICE** {  
    **Chaîne-Numérique,**  
    **Chaîne-Imprimable}**  
-- *Types de codage*  
**Types-Codage** :: = **[APPLICATION 5] IMPLICIT SET** {  
    **[0] IMPLICIT BIT STRING** {  
        **non-défini(0), tLX(1), texte-IAS(2),**  
        **fax-G3(3), tIF0(4), tTX(5), vidéotex(6),**  
        **voix(7), sFD(8), tIF1(9)**  
    }  
    **[1] IMPLICIT Params-Non-Essentiels-G3 OPTIONAL,**  
    **[2] IMPLICIT Params-Non-Essentiels-Télétext OPTIONAL,**  
    **[3] IMPLICIT Capacités-Présentation OPTIONAL.**  
    -- *D'autres paramètres non essentiels sont réservés pour étude ultérieure --*

---

---

FIGURE 16/X.411 (partie 3 de 4)

**Définition formelle de la MPDU UTILISATEUR**

*Suite des définitions P1*

```
Params-Non-Essentiels-G3 ::= BIT STRING,  
                           bidimensionnel(8),  
                           haute-Définition(9),  
                           longueur-Illimitée(20),  
                           longueur-B4(21),  
                           largeur-A3(22),  
                           largeur-B4(23),  
                           non-Comprimé(30);  
  
Params-Non-Essentiels-Télétex ::= SET {  
                           jeux-Caractères-Graphiques [0] IMPLICIT Chaîne-T61 OPTIONAL,  
                           jeux-Caractères-Commande [1] IMPLICIT Chaîne-T61 OPTIONAL,  
                           formats-Page [2] IMPLICIT OCTET STRING OPTIONAL,  
                           capacités-Diverses-Terminal [3] IMPLICIT Chaîne-T61 OPTIONAL,  
                           usage-Privé [4] IMPLICIT OCTET STRING OPTIONAL};  
  
Capacités-Présentation ::= T73.Capacités-Présentation
```

FIGURE 16 X.411 (partie 4 de 4)

Définition formelle de la MPDU UTILISATEUR

### *Suite des définitions P1*

<b>MPDU-Rapport-Remise</b>	:: = <b>SEQUENCE</b>
	<b>Enveloppe-Rapport-Remise, Contenu-Rapport-Remise,</b>
<b>Enveloppe-Rapport-Remise</b>	:: = <b>SET</b>
	<b>rapport Identificateur-MPDU,</b> <b>expéditeur Nom-OR,</b> <b>Informations-Trace}</b>
<b>Contenu-Rapport-Remise</b>	:: = <b>SET</b>
	<b>initial Identificateur-MPDU,</b> <b>intermédiaires Informations-Trace OPTIONAL,</b> <b>iD-UA-Contenu OPTIONAL,</b> <b>[0] IMPLICIT SEQUENCE OF Info-Destinataire-Concerné,</b> <b>renvoyé [1] IMPLICIT Contenu-UMPDU OPTIONAL,</b> <b>informations-Taxation [2] ANY OPTIONAL]</b>
<b>Info-Destinataire-Concerné</b>	:: = <b>SET</b>
	<b>destinataire [0] IMPLICIT Nom-OR,</b> <b>[1] IMPLICIT Identificateur-Complémentaire,</b> <b>[2] IMPLICIT Indicateur-Destinataire,</b> <b>[3] IMPLICIT Dernières-Informations-Trace,</b> <b>destinataire-Prévu [4] IMPLICIT Nom-OR OPTIONAL,</b> <b>[5] IMPLICIT Informations-Supplémentaires OPTIONAL</b>
<i>-- Dernières informations de trace</i>	
<b>Dernières-Informations-Trace</b>	:: = <b>SET</b>
	<b>arrivée [0] IMPLICIT Date-Heure,</b> <b>convertis Types-Codage OPTIONAL,</b> <b>[1] Rapport</b>
<b>Rapport</b>	:: = <b>CHOICE</b>
	<b>[0] IMPLICIT Info-Remises,</b> <b>[1] IMPLICIT Info-Non-Remises}</b>
<b>Info-Remise</b>	:: = <b>SET</b>
	<b>remise [0] IMPLICIT Date-Heure,</b> <b>type-UA [1] IMPLICIT INTEGER</b> <b>public(0), privé(1), DEFAULT public</b>
<b>Info-Non-Remise</b>	:: = <b>SET</b>
	<b>[0] IMPLICIT Code-Raison,</b> <b>[1] IMPLICIT Code-Diagnostic OPTIONAL</b>
<b>Code-Raison</b>	:: = <b>INTEGER</b>
	<b>incident-Transfert(0),</b> <b>incapable-Transférer(1),</b> <b>conversion-Non-Effectuée(2)</b>

**FIGURE 17/X.411** (partie 1 de 2)

## Définition formelle de la MPDU RAPPORT DE REMISE

*Suite des définitions P1*

**Code-Diagnostic** :: = **INTEGER** :  
nom-OR-Non-Reconnu(0),  
nom-OR-Ambigu(1),  
saturation-MTA(2),  
bouclage-Détecté(3),  
uA-Indisponible(4),  
délai-Max-Expiré(5),  
types-Codage-Non-Acceptés(6),  
contenu-Trop-Long(7),  
conversion-Impossible(8),  
conversion-Interdite(9),  
conversion-Implicite-Non-Déclarée(10),  
paramètres-Non-Valides(11)

-- Informations supplémentaires

**Informations-Supplémentaires** :: = **Chaîne-Imprimable** -- de longueur limitée et réservées  
-- pour étude ultérieure --

---

---

FIGURE 17 X.411 (partie 2 de 2)

**Définition formelle de la MPDU RAPPORT DE REMISE**

---

---

-- Suite des définitions P1 --

**MPDU Essai** :: = **Enveloppe-Essai**  
**Enveloppe-Essai** :: = **SET** :  
essai Identificateur-MPDU,  
expéditeur Nom-OR,  
Type-Contenu,  
ID-U-A-Contenu OPTIONAL,  
initiaux Types-Codage OPTIONAL,  
Informations-Trace,  
Indicateur-Message DEFAULT II,  
longueur-Contenu [0] IMPLICIT INTEGER OPTIONAL,  
[1] IMPLICIT SEQUENCE OF Info-Bilatérales-Domaine OPTIONAL,  
[2] IMPLICIT SEQUENCE OF Info-Destinataire

**END** -- Fin des DEFINITIONS P1

---

---

FIGURE 18 X.411

**Definition formelle de la MPDU ESSAI**

**P2 DEFINITIONS :: =**

BEGIN

-- P2 fait usage des types definis dans les modules suivants:  
-- P1: X.411, § 3.4  
-- P3: X.411, § 4.3  
-- SFD: presente Recommandation, § 5  
-- T73: T.73, § 5

**UAPDU :: = CHOICE {**

[0] IMPLICIT UAPDU-IM,  
[1] IMPLICIT UAPDU-SR

-- UAPDU MESSAGE IP

**UAPDU-IM :: = SEQUENCE {En-Tête, Corps}**

-- en-tête

**En-Tête :: = SET {**

Id-Message-IP

expéditeur

visas-Expédition

destinataires-Principaux

destinataires-Copie

destinataires-Copie-Muette

en-Réponse-A

annule

référence

objet

date-Péremption

réponse-Avant

réponse-A-Utilisateurs

importance

niveau-Confidentialité

autoretransmis

[0] IMPLICIT Descripteur-OR OPTIONAL,

[1] IMPLICIT SEQUENCE OF Descripteur-OR

OPTIONAL, -- seulement si ce n'est pas l'expéditeur

[2] IMPLICIT SEQUENCE OF Destinataire OPTIONAL,

[3] IMPLICIT SEQUENCE OF Destinataire OPTIONAL,

[4] IMPLICIT SEQUENCE OF Destinataire OPTIONAL,

[5] IMPLICIT Id-Message-IP OPTIONAL,

-- omis si ce n'est pas en réponse à un message antérieur

[6] IMPLICIT SEQUENCE OF Id-Message-IP OPTIONAL,

[7] IMPLICIT SEQUENCE OF Id-Message-IP OPTIONAL,

[8] CHOICE Chaîne-T61 OPTIONAL,

[9] IMPLICIT Date-Heure OPTIONAL,

-- si omise, date de péremption = jamais

[10] IMPLICIT Date-Heure OPTIONAL,

[11] IMPLICIT SEQUENCE OF Descripteur-OR OPTIONAL,

-- chaque descripteur d'O/R doit contenir un nom d'O/R

[12] IMPLICIT INTEGER :faible(0), normale(1), haute(2);DEFAULT normale,

[13] IMPLICIT INTEGER :personnel(1), privé(2), confidentiel-Entreprise(3);

OPTIONAL,

[14] IMPLICIT BOOLEAN DEFAULT FALSE

-- indique que la ou les parties du corps du message retransmises l'ont été automatiquement --.

**Id-Message-IP :: = [APPLICATION 11] IMPLICIT SET {**

Nom-OR OPTIONAL,

Chaine-Imprimable}

**Nom-OR :: = P1.Nom-OR**

FIGURE 3/X.420 (partie 1 de 3)

Définition formelle de l'UAPDU IM

-- Suite des définitions P2

**Descripteur-OR**

:: = SET | -- au moins un des deux premiers membres doit figurer  
**Nom-OR OPTIONAL,**  
**nom-Forme-Libre [0] IMPLICIT Chaîne-T61 OPTIONAL,**  
**numéro-Téléphone [1] IMPLICIT Chaîne-Imprimable OPTIONAL;**

**Destinataire**

:: = SET |  
[0] IMPLICIT Descripteur-OR,  
demande-Rapport [1] IMPLICIT BIT STRING |  
avis-Réception(0),  
avis-Non-Réception(1),  
renvoi-Message-IP(2) **DEFAULT** ||,  
-- si demandé, le descripteur d'O/R doit contenir un nom d'O/R  
demande-Réponse [2] IMPLICIT BOOLEAN **DEFAULT FALSE**  
-- si valeur vrai, le descripteur d'O/R doit contenir un nom d'O/R --|

-- corps

**Corps**

:: = SEQUENCE OF Partie-Corps

**Partie-Corps**

:: = CHOICE |  
[0] IMPLICIT Texte-IAS  
[1] IMPLICIT TLX,  
[2] IMPLICIT Voix,  
[3] IMPLICIT Fax-G3,  
[4] IMPLICIT TIF0,  
[5] IMPLICIT TTX,  
[6] IMPLICIT Vidéotex,  
[7] Défini-Pays,  
[8] IMPLICIT Chiffré,  
[9] IMPLICIT Message-IP-Retransmis  
[10] IMPLICIT SFD,  
[11] IMPLICIT TIF1)

-- types de partie de corps

**Texte-IAS**

:: = SEQUENCE |  
SET (répertoire(0), IMPLICIT INTEGER iAS(5), iTA2(2))  
**DEFAULT iAS**  
-- des membres additionnels feront l'objet d'extensions ultérieures  
-- possibles à cet ensemble --,  
Chaîne-IAS)

**TLX**

:: = réservé pour étude ultérieure

**Voix**

:: = SEQUENCE |

SET, -- les membres sont réservés pour étude ultérieure  
BIT STRING|

FIGURE 3 X.420 (partie 2 de 3)

Définition formelle de l'UAPDU IM

-- Suite des définitions P2

Fax-G3	:: = SEQUENCE { SET { nombre-Pages [0] IMPLICIT INTEGER OPTIONAL, [1] IMPLICIT P1.Params-Non-Essentiels-G3), SEQUENCE OF BIT STRING}
TIF0	:: = Document-T73
Document-T73	:: = SEQUENCE OF T73.Élément-Protocole
TTX	:: = SEQUENCE { SET { nombre-Pages [0] IMPLICIT INTEGER OPTIONAL, compatible-Télex [1] IMPLICIT BOOLEAN DEFAULT FALSE, [2] IMPLICIT P1.Params-Non-Essentiels-Télétext OPTIONAL), SEQUENCE OF Chaîne-T61}
Vidéotex	:: = SEQUENCE { SET, -- les membres sont réservés pour étude ultérieure Chaîne-Vidéotex}
Défini-Pays	:: = ANY
Chiffré	:: = SEQUENCE { SET, -- les membres sont réservés pour étude ultérieure BIT STRING}
Message-IP-Retransmis	:: = SEQUENCE { SET { remise [0] IMPLICIT Date-Heure OPTIONAL, [1] IMPLICIT Informations-Remises OPTIONAL, UAPDU IM}
Informations-Remise	:: = P3.Enveloppe-Remise -- Ceci réutilise simplement un type de données et n'implique pas que les -- informations aient jamais été transportées dans P3.
SFD	:: = SFD.Document
-- A noter que les documents SFD et T73 utilisent le même espace d'étiquette particulière à une application, qui est -- différent de celui utilisé pour les autres protocoles MHS	
TIF1	:: = Document-T73

---

---

FIGURE 3/X.420 (partie 3 de 3)

Définition formelle de l'UAPDU IM

-- Suite des définitions P2

-- UAPDU RAPPORT D'ETAT D'IPM

**UAPDU-SR**

:: = SET {

[0] CHOICE {

non-Réception [0] IMPLICIT Informations-Non-Réception,  
réception [1] IMPLICIT Informations-Réception,  
concerné Id-Message-IP,  
destinataire-Effactif [1] IMPLICIT Descripteur-OR OPTIONAL,  
destinataire-Prévu [2] IMPLICIT Descripteur-OR OPTIONAL  
-- figure seulement si ce n'est pas le destinataire effectif.  
-- Le descripteur O/R doit contenir un nom d'O/R  
convertis P1.Types-Codage OPTIONAL }

**Informations-Non-Réception** :: = SET {

raison [0] IMPLICIT

INTEGER destruction-Par-UAE(0), retransmission-Auto(1),  
qualificateur-Non-Réception [1] IMPLICIT

INTEGER (périmé(0), annulé(1), abonnement-Terminé(2)) OPTIONAL,  
commentaires [2] IMPLICIT Chaîne-Imprimable OPTIONAL,  
-- sur la retransmission automatique

renvoyé [3] IMPLICIT UAPDU-IM OPTIONAL)

**Informations-Réception**

:: = SET {

réception [0] IMPLICIT Date-Heure,

envoi-Avis [1] IMPLICIT INTEGER {

explicite(0), automatique(1)} DEFAULT explicite,

[2] IMPLICIT P1.Informations-Supplémentaires OPTIONAL}

END -- Fin des définitions P2

---

---

FIGURE 4/X.420

Définition formelle de l'UAPDU SR

---

# 11 COUCHE 7/OSI : TRANSFERT ET GESTION DE FICHIERS FTAM

Le transfert de fichiers entre systèmes hétérogènes correspond à un besoin fondamental. Il en est de même pour l'accès à des fichiers ou leur manipulation sur un système à partir d'un programme ou d'une commande d'un utilisateur distant.

Ce problème est traité dans un ensemble de Standards de l'OSI regroupés dans la norme FTAM OSI/DIS 8571.

**FTAM : File Transfer , Acces and Manipulation** fournit un service complet de transfert, accès ou gestion de fichiers virtuels. Ce service est situé au niveau 7/OSI et s'appuie sur un service commun d'Application ( sous-couche ACSE : Application Control Service Elements ) et un service Présentation qui lui fournit les moyens d'un transfert transparent et ordonné notamment par l'utilisation d'une syntaxe de transfert permettant de traduire les syntaxes abstraites des deux applications communiquant.

## 11.1 Service fourni

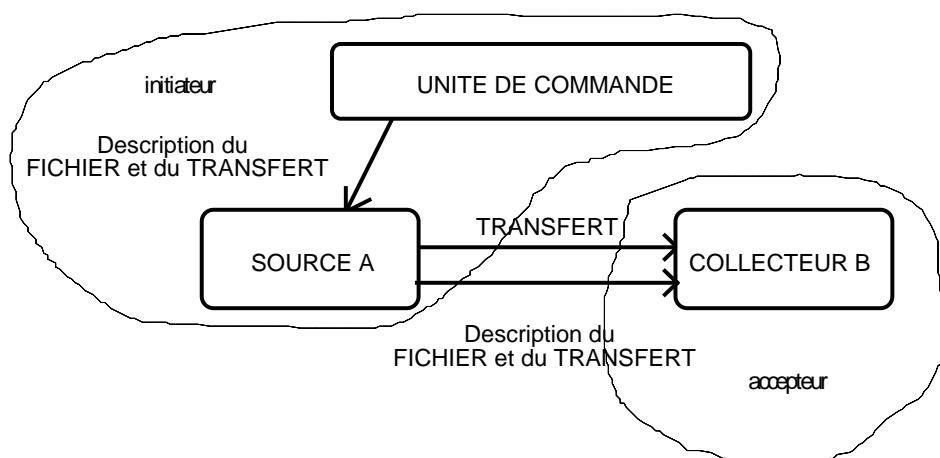
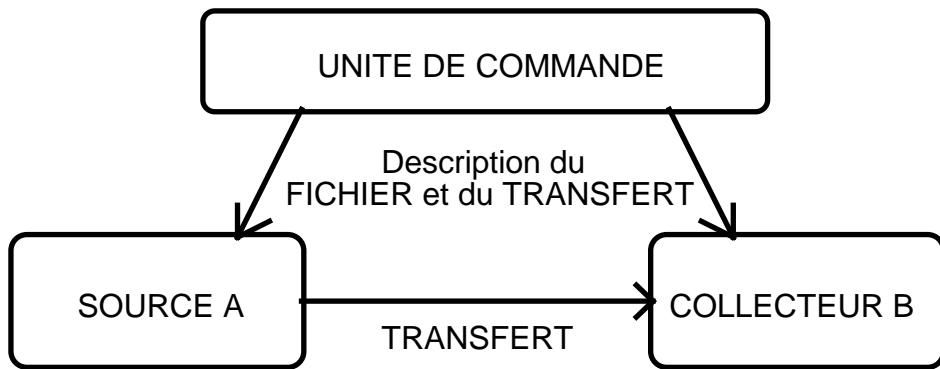
### 11.1.1 Généralités

#### 11.1.1.1 Commande de l'Activité Fichier virtuel

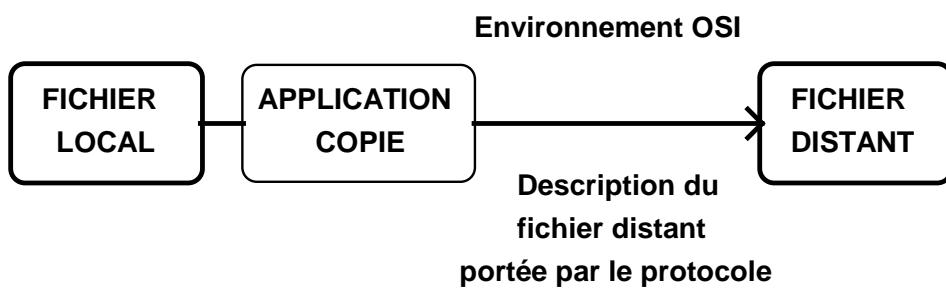
D'un point de vue logique, trois entités sont à prendre en compte :

- une unité de commande qui initie le transfert
- une source de fichier virtuel
- un collecteur de fichier virtuel

Pour simplifier la coordination et la commande, on suppose que l'unité de commande véhicule ces flux via l'une des deux entités du protocole qui agit comme son agent pour réaliser le transfert. Ceci est illustré sur le schéma ci-dessous :



Le dialogue devient asymétrique :

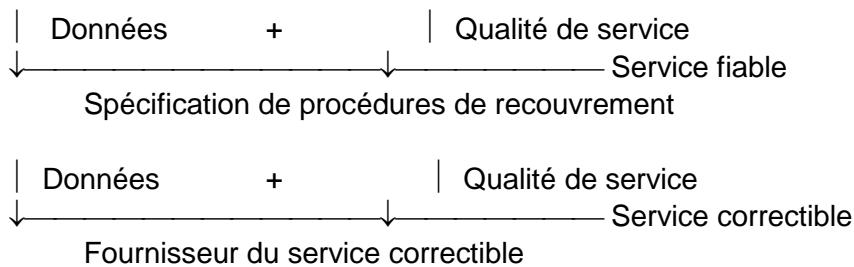


### 11.1.1.2 Service de fichier fiable et Service de fichier correctible

La communication entre initiateur et accepteur peut être décomposée en deux sous-couches :

- Service de fichier fiable avec lequel l'utilisateur n'a plus de responsabilité ou de contrainte de reprise, récupération, etc.

- Service de fichier correctible qui inclut les facilités de reprise sur défaut et d'administration. L'usager a la possibilité de choisir le mode de récupération des défauts qu'il souhaite utiliser



### 11.1.1.3 Classes de service. Unités fonctionnelles

Pour pouvoir traiter une grande variété d'applications, on dispose d'une nombre de fonctions très élevé.

Ces fonctions sont regroupées en unités fonctionnelles dont la mise en oeuvre pourra être négociée entre les deux entités communicantes.

On définit ainsi 5 classes de service qui correspondent à une sélection d'un sous-ensemble des unités fonctionnelles ci-dessous :

- Transfert
- Transfert et administration
- Accès
- "sans contrainte"

## 11.1.2 Fonctions associées au service fichier

### 11.1.2.1 Contrôle d'accès

Il est basé sur des listes de contrôle d'accès. Chaque entrée dans la liste fournit un ensemble d'actions permises si les conditions associées sont remplies.

### 11.1.2.2 Comptes

Un mécanisme de compte et de mesure de la charge d'utilisation est créé pour affecter les coûts de stockage à un compte particulier et les coûts d'accès à d'autres comptes (éventuellement). Des paramètres de charge permettent de calculer les coûts réels d'une opération avant de les affecter à un compte.

### 11.1.2.3 Contrôle de concurrence

L'objectif du contrôle de concurrence est d'assurer l'initiateur d'une action qu'il aura une vue consistante du fichier durant cette action en apportant les restrictions d'accès nécessaires sur les fichiers partagés.

Le niveau externe de fichier porte sur le fichier complet et ses attributs.

Le niveau interne porte sur le contenu du fichier lorsqu'il est ouvert.

Les verrous suivants peuvent être placés :

- Partagé - par tous
- Exclusif - pour un utilisateur
- non demandé - pour un utilisateur particulier
- pas d'accès - accès interdit à tous

Les opérations suivantes peuvent être contrôlées :

- lecture
- insertion
- remplacement
- effacement (erase)
- extension
- lecture d'attribut
- changement d'attribut
- \* suppression du fichier (delete)

Rq: la fonction effacement n'entraîne la suppression physique du fichier que s'il n'existe aucun autre lien à ce fichier.

## 11.2 Services requis

Le service FTAM s'appuie sur les services fournis par les sous-couches ou couches inférieures du modèle de référence.

### 11.2.1 ACSE

Il s'agit de l'élément de service de plus bas du niveau de la couche Application, généralement fourni avec la couche Présentation.

On utilise sa fonction de commande d'association pour

- établir
- relâcher

une association entre deux entités d'application.

Une seule activité fichier peut être en cours sur une association à un moment donné. Lorsque cette activité est terminée l'association peut être supprimée.

### 11.2.2 Présentation

Le service présentation fournit une syntaxe de transfert permettant de traduire la représentation locale du fichier. Il fournit aussi une syntaxe (et un contexte de transfert) pour assurer les fonctions de transfert, d'accès ou de manipulation de ces fichiers. Pour un transfert de fichier(s) une négociation de contextes de présentation devra être effectuée.

### 11.2.3 Session

Pour pouvoir assurer un service fiable il est nécessaire de disposer des fonctions de :

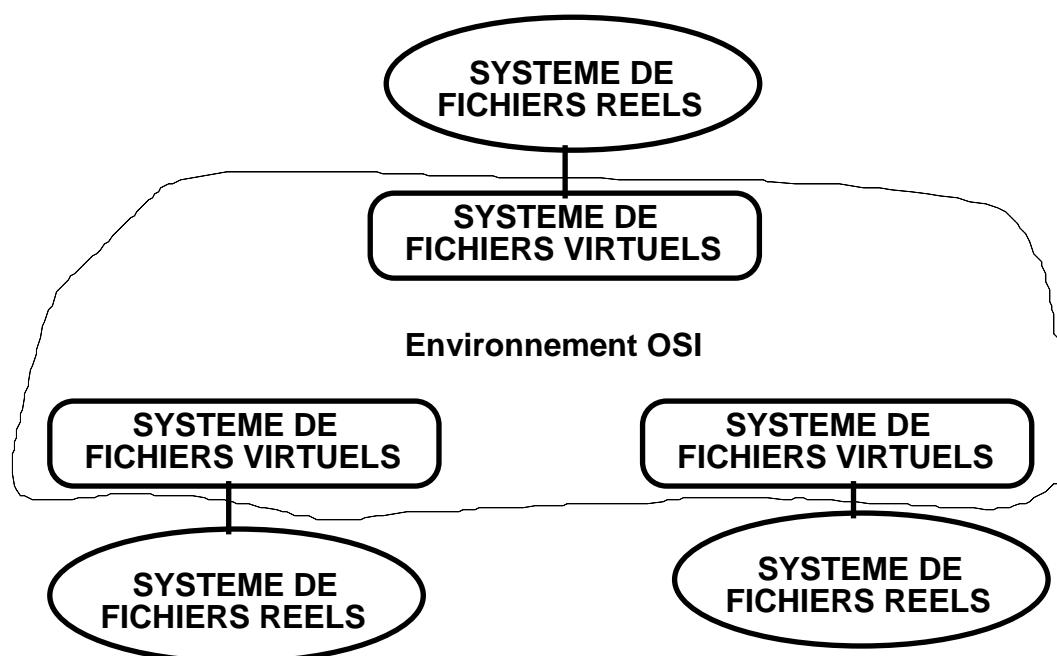
- insertion de points de synchronisation
- resynchronisation
- gestion de jetons (synchro, terminaison)

Ces fonction correspondent au sous-ensemble BSS.

## 11.3 Modèle de système de fichiers virtuel

### 11.3.1 Fichiers réels et fichiers virtuels : projection.

Pour répondre à la diversité très grande des systèmes de gestion de fichiers, il est nécessaire de concevoir un système virtuel permettant de décrire toutes les propriétés (ou presque) des fichiers réels et de réaliser une "projection" locale (mapping) d'un fichier réel sur un fichier virtuel. Cette projection est illustrée par le schéma ci-dessous.



Cette projection est faite entre les actions, les accès fichiers, les fichiers, leurs attributs et les ressources dans l'environnement réel.

### 11.3.2 Caractéristiques d'un fichier virtuel

Dans un système de fichiers virtuels, un fichier est une entité qui possède :

- un nom simple, non ambigu
- des attributs qui expriment ses propriétés : compte, historique, etc.
- des attributs décrivant sa structure logique et la dimension des données stockées
- des unités de données formant le contenu du fichier

Ces caractéristiques sont intangibles et toute action faite par deux initiateurs doit produire le même effet.

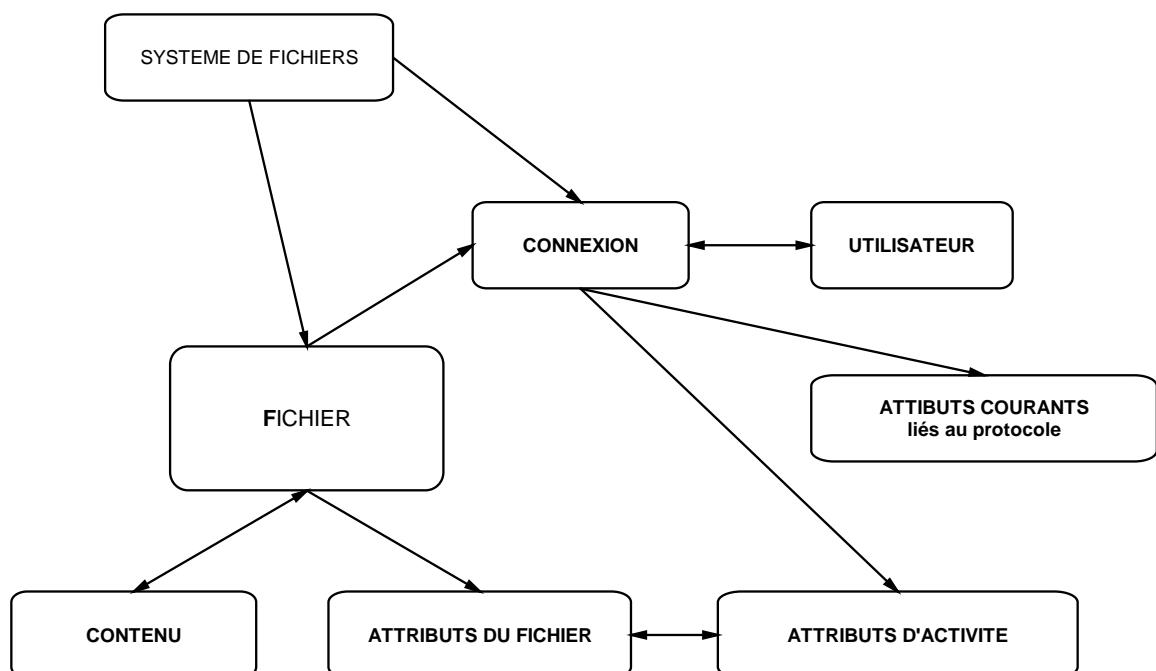
On doit ajouter

des attributs d'activité : authentification, options de transfert, coûts accumulés par exemple.

Ils sont créés et gérés par FTAM pour une activité particulière.

Enfin certains attributs portent des contraintes qui ne sont utiles qu'à un sous-ensemble des utilisateurs, par exemple un chemin d'accès complet pour un utilisateur d'une station inutile à ceux d'une autre.

### 11.3.3 Schéma d'un système de fichiers virtuels



(a) : Association 1 pour 1

#### 11.3.4 Structure d'accès

Un fichier peut contenir zéro, une ou plusieurs unités de données identifiables. Ces unités de données sont apparentées et sont reliées par une structure **arborescente**.

A chaque noeud de l'arbre est associé **zéro ou une** unité de données. La structure des informations du point de vue de l'utilisateur peut être différente de la structure d'accès et le service utilisateur devra être projeté sur la structure d'accès.

En général les structures d'information peuvent être séquentielles, hiérarchiques, en réseau ou relationnelles.

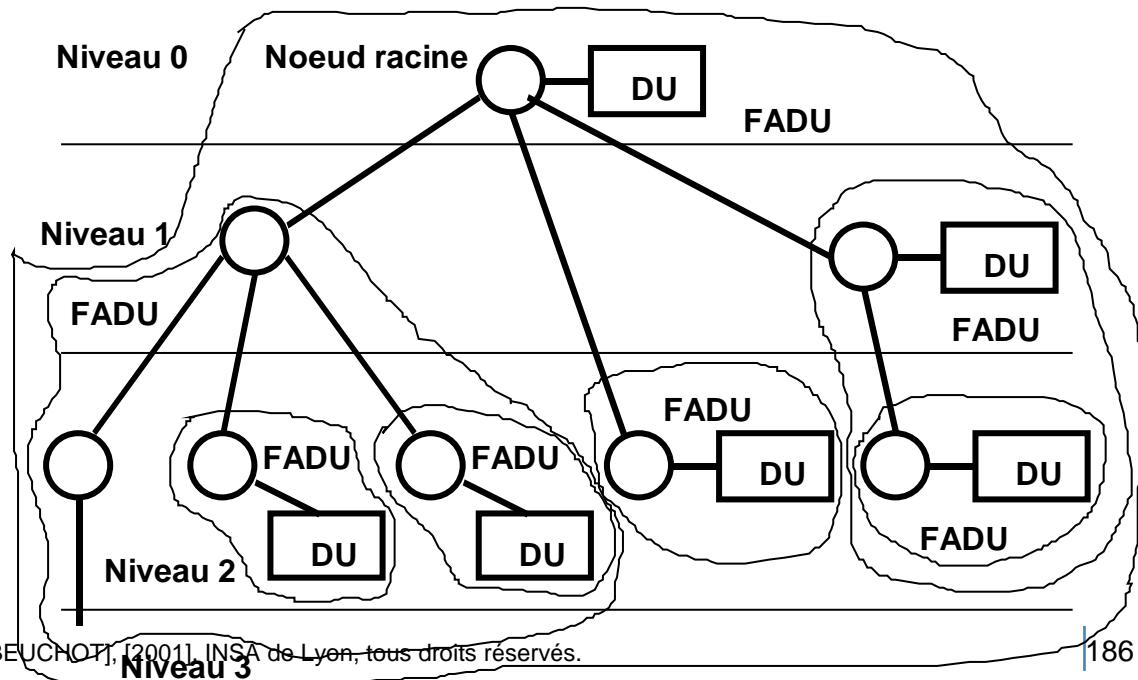
Actuellement seule les **structures séquentielles et hiérarchiques sont supportées par FTAM**. D'autres structures pourront être ajoutées (par exemple pour traiter les bases de données relationnelles).

Les unités de données sur lesquelles portent les opérations sont nommées **FADU : File Acces Data Unit**. Ces FADU peuvent elles aussi être structurées

Dans cette structure de fichiers on distingue 4 aspects :

- structure d'accès
- structure de présentation : décrit les syntaxes abstraites des unités de données définies dans la structure d'accès.
- structure de transfert : décrit la sérialisation des unités de données pour les besoins de communication
- structure d'identification : décrit le nommage des noeuds dans la structure d'accès et l'identification des fichiers à transférer.

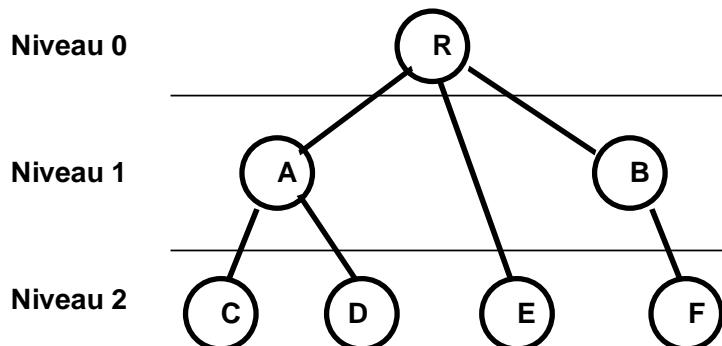
Le schéma ci-dessous donne un exemple de structure d'accès.



FADU : File Acces Data Unit

DU : Data Unit

Cette structure du système de fichiers virtuels induit un préordre pour traiter : localiser, transférer ou manipuler un ensemble de fichiers.



Ici le préordre sera R , A , C, D , E , B , F

### 11.3.5 Structures de présentation

Actuellement sept structures de présentation sont supportées par FTAM :

1) non structuré une seule unité de données, sans nom  
exemple : fichier flot (stream)

2) séquentiel plat  
suite d'unités de données non nommées (sur le modèle des fichiers d'entrées- sorties séquentiels Fortran)  
exemple : fichier séquentiel

3) ordonné plat  
suite d'unités de données nommées. La manipulation de données avec des noms dupliqués est possible (liens)  
exemple : fichier séquentiel indexé

4) ordonné plat à noms uniques.  
suite d'unités de données avec des noms spécifiques.  
exemple : peut coller à un fichier relatif ?

5) ordonné hiérarchique.  
Hiérarchie générale permise, index multiples; l'insertion est basée sur la position dans ces index ; deux noeuds peuvent avoir le même nom , ils sont distingués par l'ordre de parcours de l'arbre.

6) hiérarchique général.  
Hiérarchie générale mais avec contrôle complet sur le placement des nouveaux noeuds quand la structure est modifiée.

## 7) hiérarchique général à noms uniques

Les fichiers hiérarchiques correspondent aux "répertoires".

### 11.3.6 Types de documents

Le contenu d'un fichier peut être de différents types.

Il est possible de définir séparément le modèle de fichier, les contraintes et la syntaxe abstraite dans laquelle sont codées les données. Cependant, il est souvent plus commode de regrouper ces caractéristiques dans quelques ensembles couramment utilisés : ceci est réalisé par la définition de types de documents.

Un type de document est caractérisé par une dizaine de paramètres :

- identification du type de document
- portée souhaitée
- sémantique à appliquer pour interpréter le document
- structure d'accès par un ensemble de contraintes appliquées sur le modèle hiérarchique général
- syntaxe abstraite de l'information structurante et des unités de données dans les structures
- syntaxe de transfert
- résultat de la concaténation de deux instances d'un même type de document
- moyen de simplifier la structure d'accès pour faire voir une instance d'un document comme un type de document plus simple.

Le concept de type de document est récursif et permet par raffinements successifs de travailler sur des classes plus fines de documents.

exemples : texte non structuré , texte séquentiel (structuré en champs. .) , binaire non structuré , etc.

## 11.4 Service de transfert de fichiers

Le service de transfert de fichiers et le protocole qui le supporte, permet de créer étape par étape un environnement de travail dans lequel pourront prendre place les activités à exécuter.

Le dialogue doit permettre

- à l'initiateur et au répondeur, d'établir leurs identités mutuelles
- d'identifier le fichier à traiter

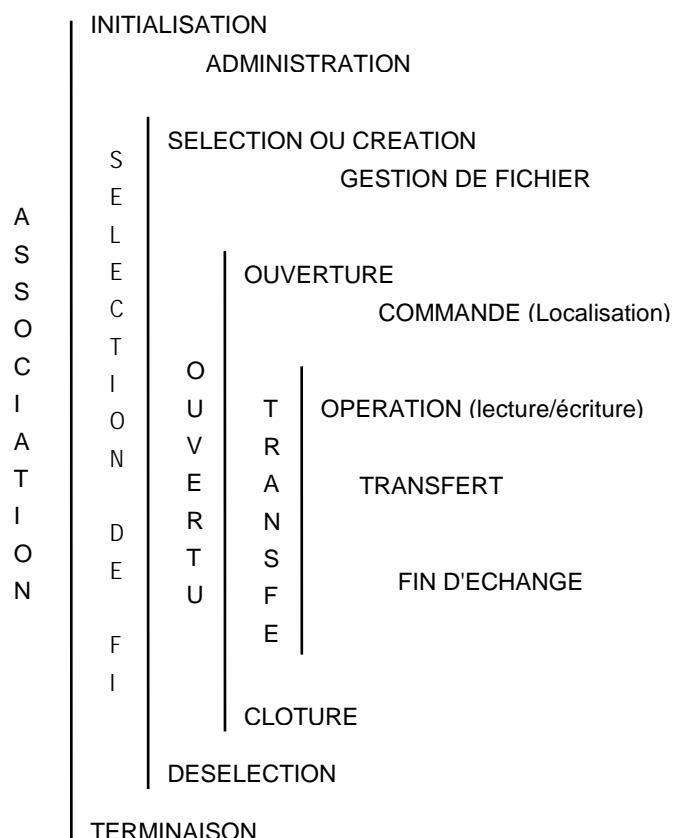
- D'établir les attributs décrivant le fichier et ses données
- de réaliser la gestion du fichier
- de localiser la position des données à traiter dans la structure des FADU
- de transférer, remplacer, effacer une ou plusieurs FADU

### 11.4.1 Phases et Régimes

Les périodes durant lesquelles un **état du service** est valide pour les deux applications qui correspondent est appelé un **régime**. En avançant dans les étapes on établit les régimes successifs correspondants.

Les périodes durant lesquels il y a des échanges protocolaires sont des **phases** (établissement, transfert, etc.)

Les activités progressent durant les phases.



- Le régime **association FTAM** commence après la **phase d'initialisation** et est clos par la **phase de terminaison**. Ces phases utilisent les primitives

F-INITIALIZE  
F-TERMINATE  
F-ABORT

Ce régime comporte aussi la **phase d'administration** du système de fichiers.

- Le régime **sélection de fichier** suit une **phase de sélection ou une phase de création** de fichier. Il se termine par une **phase de désélection ou une phase d'effacement** (delete). Il peut comporter une phase de gestion de fichier. Les primitives suivantes sont utilisées

F-SELECT  
F-CREATE  
F-READ-ATTRIB

F-CHANGE-ATTRIB  
F-DESELECT  
F-DELETE

- Le régime d'**ouverture de fichier** va de la **phase d'ouverture à la phase de clôture**. Il comporte une **phase de commande des opérations** qui permet de localiser le noeud origine dans la structure d'accès et de supprimer une FADU de cette structure (erase). Ces phases utilisent les primitives

F-OPEN  
F-LOCATE  
F-ERASE  
F-CLOSE

- Le régime de **transfert de données** comporte des **phases d'accès aux données**. Un transfert comporte trois opérations :
  - définition du type d'opération et identification des données d'accès à appliquer
  - tout transfert de données utile
  - fin de l'échange

Les primitives ci-dessous sont mises en oeuvre

F-READ  
F-WRITE  
F-DATA  
F-DATA-END  
F-TRANSFER-END

Ainsi pour réaliser un transfert de fichier les phases suivantes sont exécutées successivement :

- initialisation de l'association FTAM
- administration du système de fichier (si nécessaire)
- sélection du fichier
- administration du fichier (si nécessaire)
- ouverture du fichier
- accès aux données
- définition de l'opération (lecture ou écriture)
- transfert proprement dit
- fermeture du fichier
- désélection
- terminaison de l'association FTAM

## 11.4.2 Eléments de service

Le service FTAM fournit un **Transfert de données unidirectionnel avec des points de contrôle confirmés**.

Pour simplifier l'implantation des entités de protocole, les classes de service et les unités fonctionnelles ont été choisies pour donner des automates simples avec une interdépendance aussi faible que possible entre unités fonctionnelles.

### 11.4.2.1 Concaténation

Une **concaténation** des PDU est introduite pour éviter des échanges trop nombreux correspondant aux phases élémentaires. Les PDU deviennent ainsi de simples champs d'un message.

Cette concaténation est obligatoire pour la classe de service de transfert. Pour des classes de service plus puissantes, elle est optionnelle pour donner plus de flexibilité et supporter une plus grande variété d'applications.

### 11.4.2.2 Transparence

Pour le fournisseur du service présentation, les données utilisateur (fichier) comme les informations de contrôle du protocole d'application se présentent comme une suite de données. Si ces deux types de données se servent de la même syntaxe abstraite, des ambiguïtés peuvent se produire. Pour les éviter, FTAM utilise deux **contextes de présentation** différents pour faire transférer PCI et SDU.

### 11.4.2.3 Points de contrôle

Pour permettre des reprises du transfert et la récupération des défauts, des points de contrôle sont insérés dans le flux de données.

Source et collecteur doivent sauvegarder les informations donnant la position relative dans un fichier des différents points de contrôle.

Il est aussi nécessaire de garder des informations sur l'endroit de la structure où est inséré le point de contrôle. Ces informations sont de deux types :

information sur le contenu du fichier telles que position dans la structure d'accès ou relations entre unités de données (pointeurs, identificateur)

informations sur l'état de l'interprétation de la syntaxe abstraite ; pour les simplifier, l'insertion des points de contrôle est généralement réalisée à la limite d'unités de données complètes définies dans la syntaxe abstraite.

Ainsi, un fichier devra apparaître comme une suite d'éléments de données concaténés, chacun d'eux étant codable indépendamment des autres.

#### **11.4.2.4 Diagnostics et résultats**

Le protocole fournit deux niveaux d'information pour évaluer le succès ou l'échec d'une opération demandée.

Le premier donne des informations très générales sur le résultat d'action  
sur : le protocole lui-même  
le système de fichiers

Par exemple, dans une opération de suppression, la transition de sélection au niveau du protocole aboutit toujours alors que la suppression dans le système de fichiers peut ne pas être effective.

Le résultat d'une telle action peut être : succès, erreur récupérable ou erreur complète.

Le second niveau, donné par le paramètre diagnostic, peut contenir des informations plus détaillées et plus spécifiques de l'utilisateur.

#### **11.4.2.5 "Bordereau" et stockage rémanent**

La récupération sur défaut permet de poursuivre une activité même après une rupture du service présentation ou de l'association fournie par le service ACSE. Pour cela il est nécessaire de préserver les informations concernant l'activité et l'étape atteinte. Le corps de l'information à sauvegarder est appelé un "**bordereau**" (**docket**).

Ceci n'implique pas une concentration en un seul endroit, mais une **permanence** de cette information. Elle peut être stockée sur disque, en mémoire RAM rémanente (secourue) ou sur tout autre support.

#### **11.4.2.6 Mécanismes de récupération sur erreur**

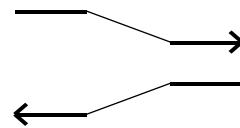
Ces mécanismes doivent fournir un **service de transfert fiable**.

Lorsque le système de transfert est détruit, le système reconstruit, **à partir des "bordereaux"** une connexion supportant le transfert et la replace dans l'état précédent le défaut.

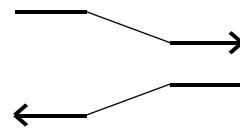
## 11.5    exemples

### 11.5.1    Transmission d'un fichier complet à un système distant

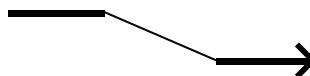
F-INITIALIZE



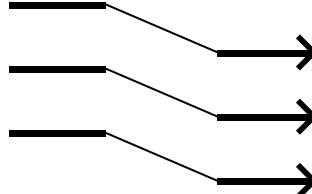
F-CREATE / F-OPEN



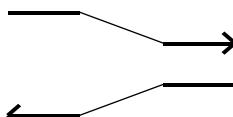
F-WRITE



F-DATA  
F-DATA



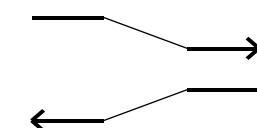
F-DATA



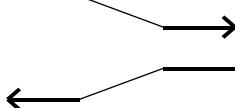
F-DATA-END / F-TRANSFER-END

F-TREANSFER-ENDrsp

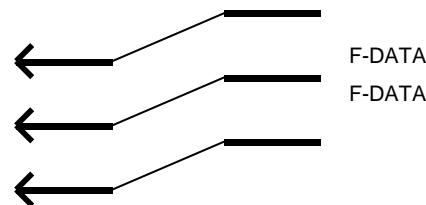
F-CLOSE / F-DESELECT



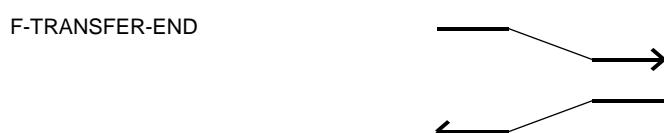
F-TERMINATE



### 11.5.2 Accès à une base de données

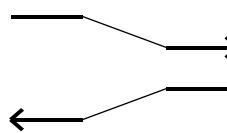


F-DATA-END

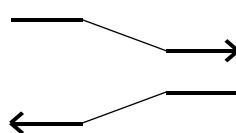


### 11.5.3 Serveur de fichiers sur un réseau local

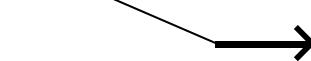
F-INITIALIZE



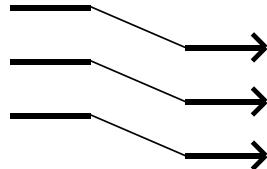
F-CREATE / F-OPEN



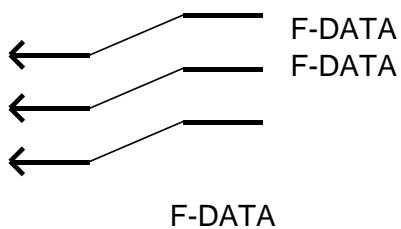
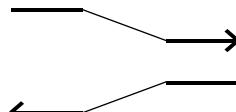
F-WRITE



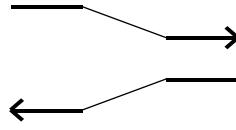
F-DATA  
F-DATA



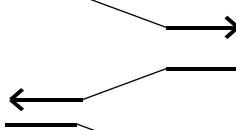
F-DATA  
F-TRANSFER-END



F-TRANSFER-END



F-CLOSE / F-DESELECT

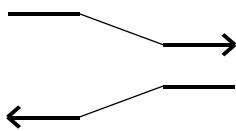


F-TERMINATE

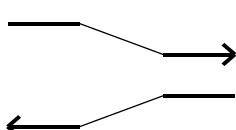


#### 11.5.4 Administration de fichier

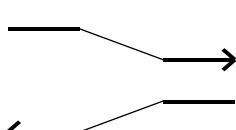
F-INITIALIZE



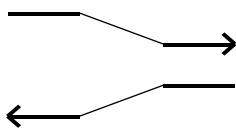
F-CREATE / F-OPEN



F-CLOSE/F-DESELECT



F-TERMINATE



## 11.6 Annexe1 : Objets définis en syntaxe ASN.1

La syntaxe abstraite ASN.1 est décrite dans le standard ISO 8824. Elle correspond à la syntaxe de transfert CCITT X409 Contexte application [1] iso-ftam[1]

syntaxe abstraite [2] ftam-pci [1]

texte-simple [2]

texte-structure [3]

binaire-simple [4]

binaire-structure [5]

structure-hiéroglyphique [6]

syntaxe de transfert[3] ftam-pci [1]

texte-simple [2]

texte-structure [3]

binaire-simple [4]

binaire-structure [5]

structure-hiéroglyphique [6]

modèle de fichier [4] hiérarchique [1]

ensemble de contraintes [5]

non-structuré [1]

plat-sequentiel [2]

plat-ordonné [3]

plat-ordonné-nom-unique [4]

hiérarchique-ordonné [5]

hiérarchique-général [6]

hiérarchique-général-nom-unique [7]

type de documents [6] texte-non-structuré [1]

texte-séquentiel [2]

binaire-non-structuré [3]

binaire-séquentiel [4]

hiérarchique-simple [5]

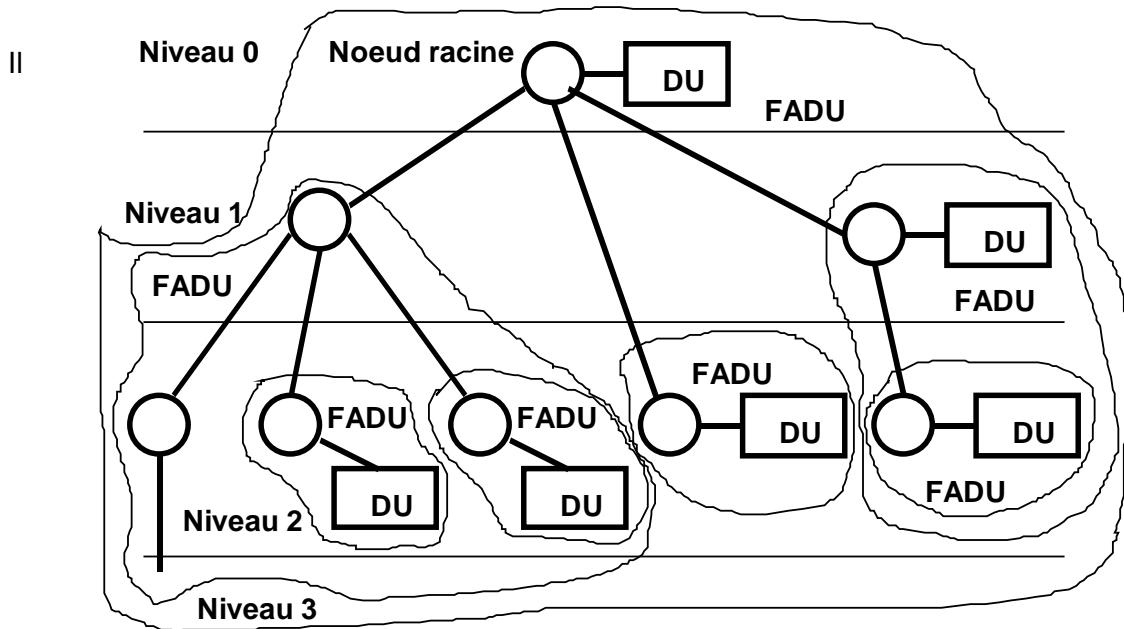
## 11.7 Annexe 2 : Modèle de système de fichiers

### 11.7.1 Concepts de base

Le système de fichiers comporte un nombre quelconque de fichiers

Les propriétés de chaque fichier sont décrites par l'ensemble des attributs du fichier. Ces attributs sont globaux et accessibles à tous les utilisateurs à un instant donné.

Tout fichier peut être vide ou avoir un contenu et une structure. Des attributs identifient les aspects structuraux du contenu.



Il existe un ensemble d'attributs d'activité FTAM associé à chaque régime. Ces attributs sont de deux types.

Pour le premier, il y a correspondance un pour un avec les attributs du fichier. Il indique la valeur active des attributs tels qu'ils sont perçus par l'initiateur.

Le second fournit la valeur courante de l'information d'état concernant les échanges en cours. Ces attributs sont dérivés des paramètres du protocole.

Un nombre arbitraire d'utilisateurs (supérieur ou égal à zéro) peut avoir initialisé des régimes FTAM à un instant donné, mais les échanges entre initiateur et répondeur ne correspondent, à un instant donné, qu'à un seul fichier dans le système de fichier du répondeur, fichier lié à un régime FTAM particulier.

NB : Voir schéma § 3.3 page 6

### 11.7.2 Structure d'accès

Le modèle de système de fichiers FTAM a une **structure hiérarchique**. La structure d'accès aux fichiers est un **arbre ordonné**.

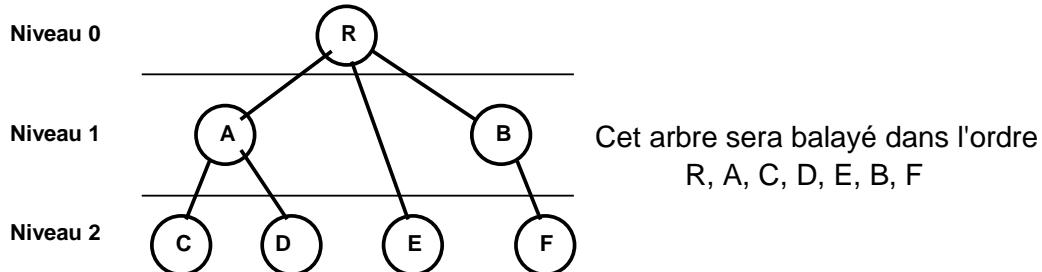
A chaque noeud de cet arbre est attaché **zéro ou une** unité de données.

Chaque noeud dans la structure donne accès à un sous-arbre. Ce sous-arbre est appelé **unité de données d'accès au fichier** ou FADU (File Acces Data Unit). Une FADU comporte les noeuds du sous-arbre et les unités de données qui leur sont attachées. Le noeud racine donne accès au fichier entier.

Optionnellement un nœud peut recevoir un nom

Le nombre de niveaux, le nombre d'arcs originaire d'un nœud, et la longueur d'un arc peuvent être quelconque.

Un **préordre** est établit pour balayer l'arbre. Il suit une séquence transversale. L'exemple ci-dessous illustre ce préordre :



- On ajoute le noeud racine d'un sous-arbre à la fin de la séquence transversale établie jusqu'ici.
- Chaque enfant d'un sous-arbre est traité dans son ordre d'apparition.

### 11.7.3 Description formelle

En syntaxe ASN.1 une FADU est définie comme ci-dessous :

```
FADU DEFINITIONS ::=  
BEGIN  
    FADU ::= Subtree  
    Subtree ::= SEQUENCE E  
        node Node-Descriptor-Data-Element,  
        data DU OPTIONAL,  
        children Children OPTIONAL ]  
  
    Children ::= SEQUENCE E  
        enter-subtree Enter-Subtree-Data-Element,  
        SEQUENCE OF Subtree,  
        exit-subtree Exit-Subtree-Data-Element ]  
  
    DU ::= SEQUENCE OF File-Contents-Data-Elements  
  
    Node-Descriptor-Data-Element ::=  
        [APPLICATION 20] IMPLICIT SEQUENCE E  
        name Node-Name OPTIONAL,  
        arc-length [1] IMPLICIT INTEGER DEFAULT 1,  
        data-exists [2] IMPLICIT BOOLEAN DEFAULT TRUE ]  
  
    Node-Name ::= CHOICE E  
        ftam-coded [0] IMPLICIT GraphicString,  
        user-coded EXTERNAL ]  
  
    Enter-Subtree-Data-Element ::=
```

[APPLICATION 21] IMPLICIT NULL

Exit-Subtree-Data-Element ::=  
[APPLICATION 22] IMPLICIT NULL

File-Contents-Data-Elements ::= EXTERNAL

Data-Element ::= CHOICE E  
Node-Descriptor-Data-Element,  
Enter-Subtree-Data-Element,  
Exit-Subtree-Data-Element,  
File-Contents-Data-Element ]

END

#### 11.7.4 Structure de transfert

La structure de transfert FTAM est dérivée de sa syntaxe abstraite. Un fichier est transféré comme une **suite d'éléments de données (Data-Elements définis ci-dessus)**.

Les informations structurantes : Descripteur de noeud, sous-arbre d'entrée, sous-arbre de sortie sont transmises dans la PCI.

Le contenu du fichier est transmis dans les données utilisateur. PCI et Données utilisateur utilisent des **contexte de transfert différents**.

Sept contextes d'accès sont définis :

HA Tous les éléments de la FADU sont transférés

HD La structure d'accès (PCI) seule est transférée. Il n'y a pas de contenu de fichier

FA Fichier plat. On transfert le descripteur de noeud et le contenu du fichier.

FL Le descripteur de noeud et le contenu de tous les fichiers d'un niveau qui contiennent des données sont transférés.

FS Le descripteur de noeud et tous les contenus de fichiers appartenant à ce noeud racine sont transférés.

UA Non structuré. Seuls sont transférés les éléments de données de type contenu de la FADU adressée.

US Les éléments de données de type contenu appartenant au noeud racine de la FADU sont transférés.

## 11.7.5 Actions sur un fichier

Sur un fichier complet

créer  
électionner (créer une relation)  
changer les attributs  
lire les attributs  
ouvrir un fichier  
clore un fichier  
désélectionner  
effacer(delete : désélectionne et efface)

Pour l'accès à un fichier

localiser première FADU  
dernière FADU  
FADU courante  
FADU suivante  
FADU précédente  
début  
fin  
Nom de noeud  
séquence de noms de noeuds  
numéro de noeud (dans le préordre racine = 0)  
numéro de niveau (seulement pour accès FL)

lire localiser et lire la FADU  
insérer  
remplacer  
étendre  
gommer (erase) (sauf racine)

---

## 12Couche 7/OSI : MESSAGERIE INDUSTRIELLE MMS

Manufacturing Message Specification

Ce service Application constitue la base des standards MAP (Manufacturing Automation Protocol). Les fonctions qu'il fournit sont directement accessibles à l'utilisateur. Il est défini dans le standard OSI 9506. La partie 9506-1 spécifie le service et la partie 9506-2 le protocole.

Ce service est basé sur un mécanisme **client-serveur** : des systèmes "serveurs" (automates, robots, manipulateurs, etc.) sont utilisés à la demande par les "clients" que sont les systèmes de supervision ou d'autres automates. Un équipement peut être à la fois client et serveur. L'architecture n'est pas purement client-serveur; elle permet la notification d'événements au client par le serveur.

En pratique, il est commode de le structurer en deux sous-couches : Un service "temps-réel" qui fournit les fonctionnalités prévues et un service "communication ou messagerie" chargé de transmettre les PDU entre les entités qui coopèrent et éventuellement d'annuler des demandes en cours.

### 12.1 Service requis

MMS ne contient aucun mécanisme d'adressage et s'appuie sur une association entre entités d'applications établie par le service ACSE . Après établissement de cette association les PDU sont transférées directement sur la connexion de Présentation correspondante.

Nota : Les fonctionnalités de ce service pourraient aussi être mises en oeuvre dans un système de communication non-OSI qui fournirait simplement un mécanisme de transfert de blocs de données bidirectionnel à l'alternat et un mécanisme d'adressage.

### 12.2 Service fourni

Le service MMS offre une très grande variété de fonctions: 84 services élémentaires et 2 modalités, qui sont regroupées logiquement en 9 sous-ensembles pour en faciliter la description. Ce sont :

Gestion de contexte  
Gestion de l'Equipement Virtuel de Production  
Gestion de variables  
Gestion de programmes  
Gestion d'événements  
Communication opérateur  
Gestion de domaines  
Gestion de sémaphores  
Gestion de journal

et un sous- ensemble annexe :

Gestion de fichiers.

Deux modalités complémentaires peuvent compléter tout service en le liant à un sémaphore ou à une condition sur événement.

De nombreux projets, par exemple le profil européen CNMA, prévoient de n'implanter dans un premier temps qu'un sous-ensemble des 84 fonctions, utilisées le plus fréquemment dans les équipements industriels actuels : automates, robots, systèmes de convoyage.

Nous ne pouvons détailler ici toutes les fonctions qui ne seront parfois qu' énumérées. Les noms des fonctions sont en général très explicites et indiquent l'objet de celles-ci. Une description de chaque fonction est donnée dans les spécifications d'interface.

## 12.3 Sous-ensembles fonctionnels

### 12.3.1 Modalités

Elles permettent de conditionner l'exécution des fonctions ci-dessous à la disposition d'un sémaphore ou à la réalisation d'une condition liée à un événement :

Ce sont :

attachToSemaphore  
attachToEventCondition

### 12.3.2 Gestion de contexte

La Gestion de Contexte permet d'**initialiser et de terminer** un dialogue entre entités MMS paires et d'en **négocier les options**. Elle utilise le service ACSE. Les options négociables portent en particulier sur la liste des services élémentaires à mettre en oeuvre et les paramètres utilisés.

Primitives utilisées :

initiate	initialisation
conclude	terminaison normale
cancel	annulation du service
abort	terminaison sur anomalie
reject	refus du service

Ce sous-ensemble doit obligatoirement être implanté.

### 12.3.3 Gestion de l'Equipement Virtuel de Production

La notion d'Equipement Virtuel de Production (EVP ou VMD Virtual Manufacturing Device) est très importante. Elle ne s'applique qu'aux serveurs MMS. Coté client, une interface permettant de demander l'exécution de services est en cours de spécification (MMS Application Interface Specification) .

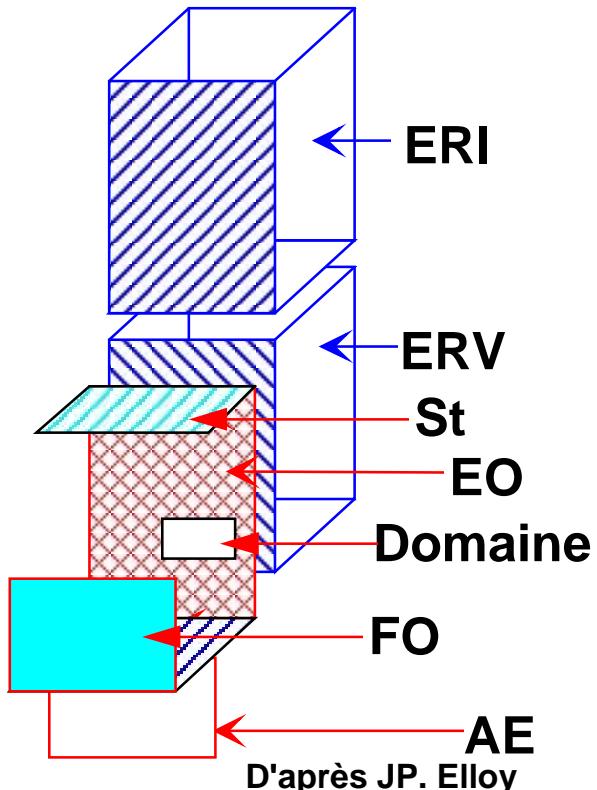
L'équipement virtuel de production permet de définir de manière logique un appareil (automate, robot, commande numérique de machine outil, etc.) ou une fonction de l'**équipement réel de production (ERP)**. L'EVP comporte un ensemble d'objets visibles, objets logiques, représentés dans la mémoire de l'équipement réel (ou d'un système frontal pour des équipements réels trop peu puissants) et mis en correspondance avec des objets réels par un logiciel spécifique, créé par l'équipementier (par exemple une procédure de conversion).

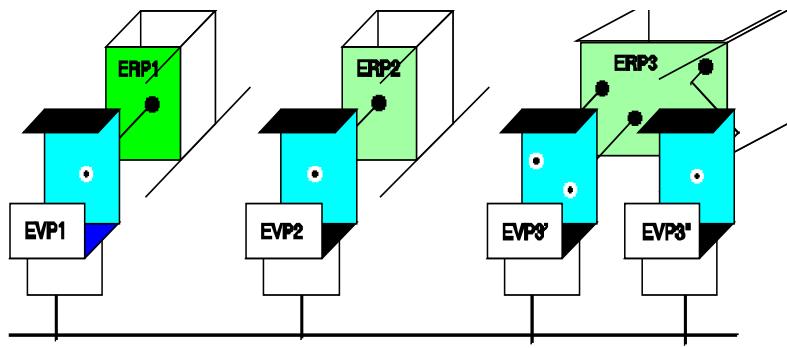
Certaines parties de l'équipement réel peuvent ne pas être traitées dans l'Equipement Virtuel et n'avoir qu'une portée locale. Elles sont notées ERI sur le schéma ci-contre. La partie visible de l'ERP est notée ERV.

Ainsi la communication n'est pas établie directement avec l'appareil réel, mais à travers une sorte de "sas" dans lequel MMS dépose ses requêtes (commandes, lecture de variables, etc.) ou vient lire les informations dont a besoin l'autre application.

L'appareil réel met à jour les "objets visibles" (variables,... notés EO) partagés ou vient retirer les commandes dans ce sas; la communication est ainsi désynchronisée du fonctionnement du processus réel.

Le schéma ci-dessous montre des exemples de correspondance entre équipement réels de production (partie "visible") et équipement virtuel correspondant. Cette correspondance n'est pas forcément biunivoque et un équipement réel complexe peut être "représenté" par plusieurs EVP correspondant à des groupes de fonctionnalités.





Les objets de l'EVP sont répartis en une dizaine de classes :

Variable

Type

Sémaphore

Condition événementielle

Action événementielle

Enveloppe événementielle

Journal

Domaine

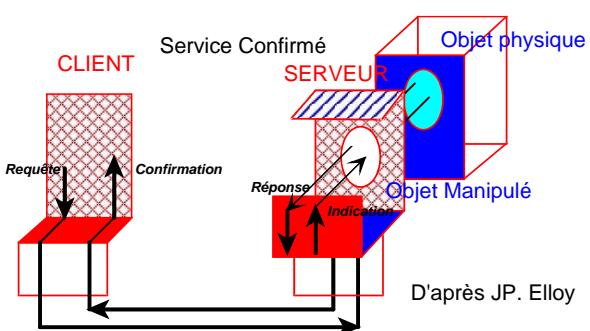
Invocation Programme

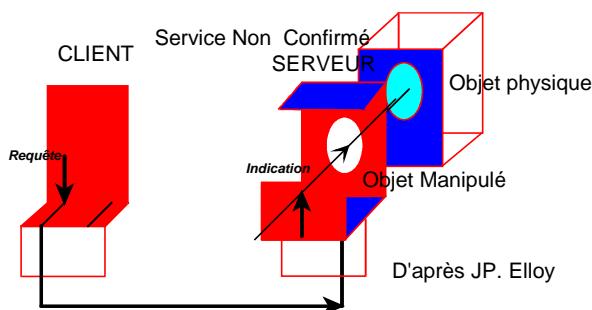
Station opérateur

Un **domaine** est un regroupement d'objets et/ou un ensemble d'informations qui peuvent être **télé-chargées** et **télé-sauvegardées**.

Ces objets, qui sont mis en correspondance avec des ressources de l'équipement réel, sont traités par la **Fonction Opératoire** (notée FO sur le schéma ci-dessus). Cette Fonction Opératoire réalise l'accès (en lecture ou en écriture) aux objets de l'EVP sur une demande de service de la part du système client ou sur des requêtes du système local.

Le service est la plupart du temps confirmé. Le schéma ci-contre illustre le mécanisme qui en permet la mise en oeuvre.





Quelques services ne sont pas confirmés (eventNotification, abort, reject, informationReport, unsolicitedStatus). Le mécanisme mis en oeuvre est plus simple.

Le schéma ci-contre illustre une coopération plus complexe entre quatre systèmes : un client, deux serveurs et un système client-serveur mettant en oeuvre des services confirmées ou non-confirmées.

Les noms de ces objets ont une portée variable :

Portée EVP

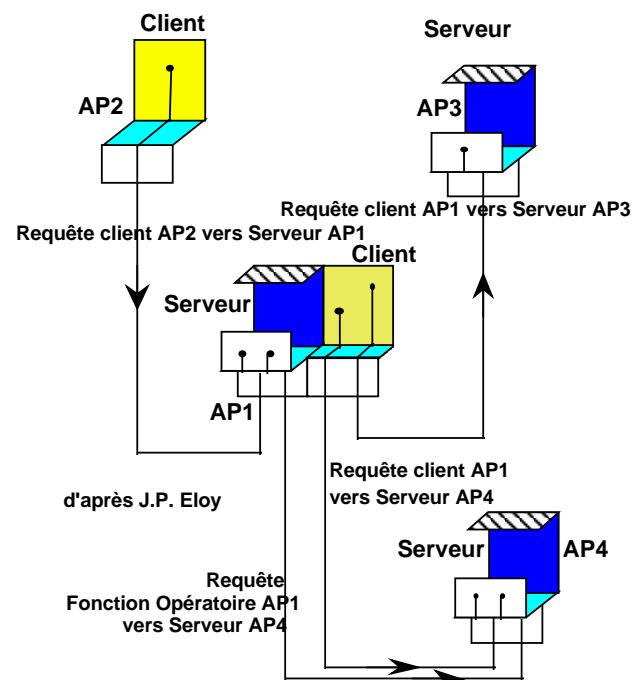
Portée Domaine (sous-ensemble de l'EVP)

Portée Association d'Applications (ensemble client-serveur associés pour traiter une application)

Tous les classes peuvent avoir l'EVP comme portée. Le journal, un domaine, une invocation de programme ou la station opérateur ne peuvent être réduit à un domaine. Un sémaphore, de même qu'un domaine, une invocation de programme ou une station opérateur, ne peuvent avoir une portée dépassant l'EVP.

Primitives utilisées :

- status
- getNameList
- identify
- rename
- getCapabilityList
- informationReport
- unsolicitedStatus



#### 12.3.4 Gestion de variables

Une variable correspond à un ou plusieurs éléments de données qui sont référencées pour le partenaire par un nom ou une description . Ces variables peuvent être prédéfinies par l'équipement réel; elles ne peuvent alors pas être renommées ou supprimées.

Un certain nombre de services élémentaires de ce sous-ensemble est nécessaire dans tous les types d'équipements réels:

La gestion de variables permet ainsi d'envoyer des commandes à un équipement réel, robot ou automate par exemple, ou de lire des informations sur un appareil. Ces variables peuvent être repérées par leur emplacement (adresse mémoire) comme dans les systèmes réels courants mais aussi par un nom logique.

Ce sous-ensemble permet en particulier :

- d'assigner un nom symbolique à un type de données
- d'obtenir le type associé à un nom symbolique
- de dissocier nom symbolique et type

Primitives associées :

defineNamedType  
getNamedTypeAttributes  
deleteNamedType

- d'assigner un nom symbolique à un emplacement distant
- d'obtenir l'emplacement associé à un nom symbolique
- de dissocier nom symbolique et emplacement

Primitives associées :

defineNamedVariable  
getVariableAccesAttributes  
deleteVariableAcces

- de lire ou de changer le contenu de une ou plusieurs variables d'une entité paire
- d'envoyer des informations à l'entité paire sans qu'elles soient demandées.

Primitives utilisées :

read  
write

- de manipuler des listes de variables

Primitives utilisées :

defineNamedVariableList  
getNamedVariablesList  
AttributesdeleteNamedVariableList

Autres primitives :

defineScatteredAcces  
getScatteredAccesAttributes  
deleteVariableAcces

### 12.3.5 Gestion de programmes

La gestion de programmes, associées à des fonctions de gestion de fichiers, permet le **lancement** de programmes ou leur arrêt. Il permet aussi de les suspendre ou de les relancer.

Leur **téléchargement** ou leur **sauvegarde** ressort de la **gestion de domaines** ou des fonctions annexes de transfert de fichiers.

Primitives associées :

```
createProgramInvocation  
deleteProgramInvocation  
start  
stop  
resume  
reset  
kill  
getProgramInvocationAttributes
```

### 12.3.6 Gestion d'événements

Cette unité fonctionnelle permet de gérer des événements sur une machine distante en créant, manipulant, inspectant ou supprimant ces événements.

Il est possible de **tester une condition** liée à un événement, de **lancer une action** ou de **Notifier une alarme sur un événement**.

Primitives utilisées :

```
defineEventCondition  
alterEventConditionMonitoring  
deleteEventCondition  
getEventConditionAttributes  
reportEventConditionStatus  
triggerEvent  
defineEventAction  
deleteEventAction  
getEventActionAttributes  
reportEventActionStatus  
defineEventEnrollment  
deleteEventEnrollment  
alterEventEnrollment  
reportEventEnrollmentStatus  
getEventEnrollmentStatus  
getEventEnrollmentAttributes  
acknowledgeEventNotification
```

```
getAlarmSummary  
getAlarmEnrollmentSummary  
eventNotification
```

Avec la gestion de sémaphores, la gestion d'événements fournit les fonctions de base pour la constitution d'un système temps réel distribué.

### 12.3.7 Gestion de sémaphores

Ce service permet, sur une machine distante, de définir ou de supprimer des sémaphores, de réserver, relâcher ou tester un sémaphore ou une ressource, enfin, de conditionner d'autres services selon la disponibilité d'une ressource (par un service modificateur) .

Primitives utilisées :

```
takeControl  
relinquishControl  
defineSemaphore  
deleteSemaphore  
reportSemaphoreStatus  
reportPoolSemaphoreStatus  
reportSemaphoreEntryStatus
```

### 12.3.8 Gestion de domaines

Ce sous-ensemble permet divers modes de **téléchargement de programmes** (ou de données) dans une zone mémoire spécifiée :

Chargement du local vers le distant (download)  
Chargement du local depuis le distant (upload)  
ou le stockage d'un programme sur la machine distante (load) .

Primitives utilisées :

```
initiateDownloadSequence  
downLoadSegment  
terminateDownloadSequence  
initiateUploadSequence  
uploadSegment  
terminateUploadSequence  
requestDomainDownLoad  
requestDomainUpLoad  
loadDomainContent  
storeDomainContent  
deleteDomain  
getDomainAttributes
```

### **12.3.9 Gestion de journal**

Le sous-ensemble "Gestion de journal" permet de créer un **journal des événements** avec le contenu des variables associées et de lire ou écrire dans ce journal .

Primitives utilisées : readJournal

- writeJournal
- initializeJournal
- reportJournalStatus
- createJournal
- deleteJournal

### **12.3.10 Communication opérateur**

Ce sous-ensemble permet de lire ou d'écrire sur les appareils d'entrée-sortie distants (console opérateur, clavier de commande, écran d'affichage, etc).

Primitives utilisées :

- input
- output

### **12.3.11 Annexe : Gestion de fichiers**

MMS fournit un minimum de fonctions pour gérer et utiliser des fichiers. Pour un traitement complet on utilisera le service FTAM (File Transfert Acces and Management) qui fournit les primitives pour un système de gestion de fichiers distribué. (Voir choix MAP ou TOP) .

Par MMS, les fichiers ne peuvent qu'être lus ou transférés en lecture. Ils peuvent être supprimés ou renommés. Enfin il est possible d'obtenir des informations sur un répertoire distant.

Primitives utilisées :

- fileOpen
- fileClose
- fileRead
- obtainFile
- fileRename
- fileDelete
- fileDirectory

## 12.4 Eléments de protocole.

La plupart de ces services sont des services confirmés. Seuls cinq d'entre eux ne le sont pas et sont simplement transmis du client au serveur :

informationReport  
unsolicitedStatus  
abort  
reject  
ou du serveur au client :  
eventNotification

L'automate correspondant ne comporte alors qu'un seul état et le service ne peut être annulé quand il a été lancé.

Les services confirmés demandent une réponse de la machine serveur distante.  
L'automate reste assez simple. Il permet toutefois d'**annuler** une requête (par cancel) qui peut ou non avoir déjà été exécutée; si le service demandé est déjà exécuté l'annulation n'a pas été prise en compte et on reçoit une confirmation négative au "cancel".

# 13 Couche 7/OSI : APPEL DE PROCÉDURES DISTANTES (RPC)

Un système d'appel de procédures distantes (RPC: Remote Procedure Call) apporte une technique simple pour concevoir des applications distribuées, caractérisées par des temps de réponse courts. Il est à la base des systèmes **client-serveur**. De tels systèmes existent en milieu homogène depuis le début des années 1980 (notamment sur des systèmes Unix fournis par Sun , Xerox puis DEC).

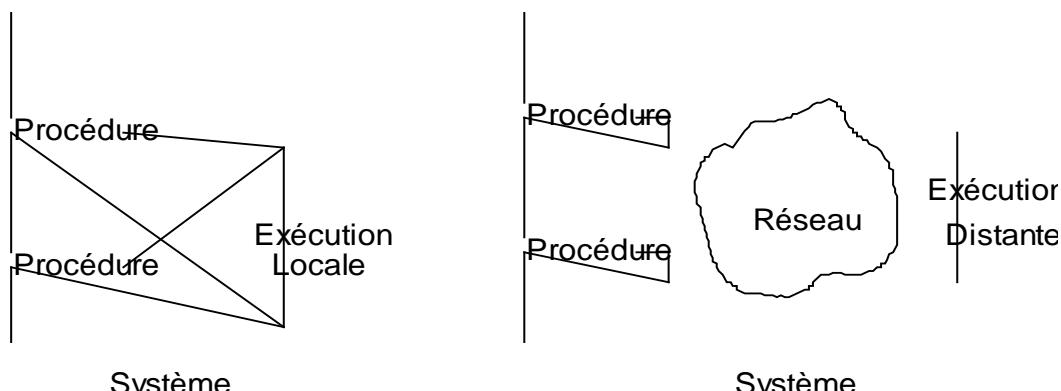
Pour les systèmes hétérogènes, un standard européen ECMA (ECMA127) a été spécifié. Un standard OSI est en cours d'élaboration depuis 1990; il reprend les concepts développés par l'ECMA (avec quelques adaptations). Il se situe au niveau 7/OSI (application) et utilise le service d'opérations distantes ROSE.

Dans le monde Unix, il est appelé à un essor important. Il doit constituer la base des applications réparties. Les projets de normalisation d'Unix prévoient de l'utiliser directement au dessus de la couche Transport : Transport OSI classe 4, TCP ou XTP à travers l'interface d'accès au service Transport XTI. Dans ce cas, on travaille en milieu homogène. Les RPC existants travaillent normalement en milieu homogène. Une certaine hétérogénéité est permise en utilisant le protocole XDR (eXternal Data Representation) pour coder les données transférées . (Ce produit joue le rôle de syntaxe de transfert).

## 13.1 Le modèle du standard RPC

### 13.1.1 Le concept de RPC ouvert

Le concept d'appel de procédures distantes consiste à étendre le mécanisme classique d'appel de procédures, existant dans la plupart des langages et s'exécutant avec des procédures dans des bibliothèques locales, à des systèmes ouverts où ces procédures sont exécutées sur un autre système accessible à travers un réseau.



Le programme appelant n'est normalement pas modifié. Sur le système local, des procédures "agents", représentant les procédures distantes qui effectuent réellement les traitements, sont invoquées par le programme appelant . Elles réalisent l'appel réel aux procédures distantes à travers le réseau. Ces procédures réelles sont installées sur un "serveur de procédures".

Les objectifs principaux sont :

- offrir au programmeur d'applications un mécanisme d'appel de procédures classiques prenant en charge les complications liées aux communications
- permettre l'utilisation de plusieurs langages de programmation, utilisés sous des systèmes d'exploitation différents, pour écrire et exécuter les différents composants d'une application distribuée
- permettre le développement d'une application distribuée dans un environnement local et de la rendre "distribuée" avec pas ou peu de changements.

Un système de RPC définit le moyen d'exécuter une procédure sur un (ou des) serveur(s) de procédures à travers une **interface standard D1**. Cette interface **définit les règles d'utilisation des procédures distantes** et permet une certaine **indépendance entre leur conception et leur appel pour exécution**.

### 13.1.2 Principe d'une application distribuée utilisant le RPC

Le composant distant d'une application est généralement construit comme un **service complet et nommé**. Les primitives qui composent ce service sont implantées sous la forme d'une **famille de procédures distantes**. Cette famille constitue le composant distant de l'application.

Ce service peut être soit spécifique d'une application soit un partage de ressources (à travers des procédures d'appel à ces ressources).

L'**appel, pour exécution**, des procédures constituant ce service distant depuis le service client doit être **transparent**, c'est-à-dire constitué d'un **appel normal de procédure**, écrit dans le langage de l'hôte (client).

Cet appel a deux destinations : une immédiate dans l'hôte, l'autre, finale, dans le serveur de procédures. La destination immédiate de l'appel est une procédure locale, "agent" de la procédure distante, qui prend en charge les opérations de communication avec le serveur distant, dans lequel se trouve les procédures à exécuter, destination finale de l'appel.

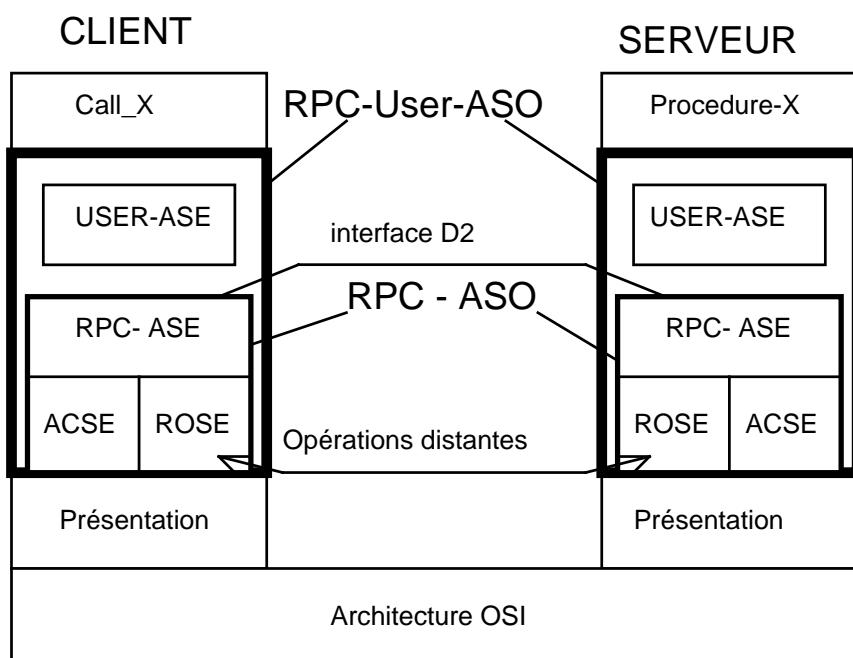
La procédure locael ("agent", "stub") attend, pour retourner au programme d'application, le résultat d'exécution, généré par la procédure appelée, ou une condition d'exception en cas d'anomalie d'exécution ou de communication. Comme dans un système localisé l'appel de procédure est "synchrone" (bloquant).

### 13.1.3 Architecture du modèle

Le schéma ci-dessous donne l'architecture d'un système RPC dans le cadre du Modèle de Référence OSI. Dans un système homogène, il peut être simplifié.

Pour s'exécuter, dans le cadre du modèle OSI, le RPC s'appuie sur un service d'opérations distantes ROSE et le service commun d'association ACSE. Il est mis en oeuvre dans des entités du service application (ASE) spécialisées. L'ensemble de ces entités est regroupé dans des "objets de service application" (ASO) selon l'architecture illustrée ci-dessous :

L'Objet de Service Application RPC (RPC-ASO) est composé de trois éléments de service d'Application : RPC-ASE, ROSE et ACSE. L'entité RPC-ASE traite le protocole RPC essentiellement en créant et en codant des opérations dont le transfert est demandé au service d'opérations distantes ROSE, après qu'une

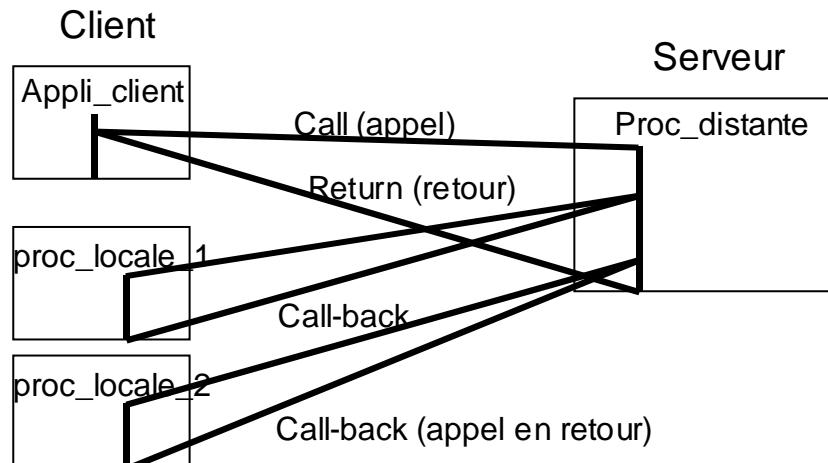


association ait été établie par ACSE. Chaque appel de procédure et chaque retour de procédure est traité par une demande d'opération distante. Cette opération comporte de nom de la procédure et ses paramètres.

Les appels du système client sont pris en compte par un élément de service d'Application spécifique (User-ASE) qui utilise le service RPC proprement dit à travers une interface standard D2.

Cette entité constitue avec l'objet de service Application RPC-ASO un objet de Service Application composite ; RPC-User-ASO. C'est l'interface avec ce service qui est accessible au programme utilisateur.

Une architecture symétrique est réalisée coté serveur. Cet ensemble



modulaire permet de réaliser l'indépendance vis à vis du système d'exploitation et du langage aussi bien coté serveur que coté client, ainsi que du système de communication.

Le RPC doit aussi supporter de manière transparente à l'utilisateur le mécanisme de call-back qui permet à une procédure exécutée sur le serveur distant de faire, pour son propre compte, des appels à des procédures de second niveau exécutées sur le système client. Un call-back peut être récursif jusqu'à un niveau de profondeur arbitraire.

### 13.1.4 Le service RPC OSI

Un RPC ouvert présente un degré important d'indépendance vis à vis de l'environnement hôte et rend à l'utilisateur un service spécifié à l'interface D2. Ce service est caractérisé par :

- son indépendance vis à vis du langage.

Le service RPC (et le serveur d'opérations distantes) doivent être indépendants du ou des langages de programmation choisis pour l'application. Les procédures sur le serveur de procédures distantes sont aussi écrites dans un langage quelconque.

Cependant on devra observer une restriction dans les paramètres d'appels ou de retour (quelque soit le langage ...) : ceux-ci doivent être effectivement transmis; il n'est bien sûr pas possible de ne passer qu'un pointeur sur ces données, celui-ci ne pouvant avoir qu'une signification locale. Dans le cas de tables ou de structures de données importantes, ceci peut éventuellement diminuer les performances du système, le temps de transfert pouvant parfois être assez important.

- son indépendance vis à vis de la machine

La représentation des données manipulées par le RPC doit être indépendante de la machine hôte (et du serveur). Ceci est réalisé en codant ces données d'appel ou de retour en syntaxe ASN.1 (OSI 8825).

Remarque : dans les RPC non-OSI on utilise souvent le langage XDR (extension des structures de données du langage C à cet effet).

- son indépendance vis à vis du réseau de télécommunications

Elle est assurée par l'utilisation de protocoles standards. Cependant les performances du systèmes sont très dépendantes du réseau de télécommunications traversé et de l'efficacité des ensembles logiciels de communication. On trouvera le RPC sur des réseaux offrant un débit suffisant : réseaux locaux (Ethernet, Token-Ring, FDDI) ou réseaux métropolitains ou étendus à haut débit (DQDB ou ATM).

Le service RPC comprend **sept primitives de service** disponibles à l'interface D2. Nous utiliserons les notations OSI en notant entre parenthèses les notations ECMA.

- **RPCBind (BeginRPCBinding)** :

Etablit un lien entre les entités RPC en proposant des valeurs d'attributs. Cette primitive n'implique aucune interaction directe entre des procédures. Ce **service, toujours invoqué par le client** (architecture client-serveur) retourne à l'utilisateur une référence "Refbind" qui identifie le lien (Bind) unique géré par les RPC-ASE communicants.

- **RPCUnBind (End RPCBinding) :**

Permet aux utilisateurs de détruire le lien de référence "RefBind", établi par la primitive ci-dessus, entre les RPC-ASE

- **(GetRPCAttributes)**

Cette primitive n'a qu'une portée locale. Elle permet aux utilisateurs du service de prendre connaissance de certains paramètres associés au lien "RefBind" en cours.

- **RPCInvoke (RPCCall) :**

Permet de demander l'exécution d'un procédure distante particulière. C'est la primitive de base pour le client. Le serveur réservera l'utilisation de cette primitive à l'appel en call-back.

- **(RPCProcedureOffered) :**

Permet au serveur de donner la liste des procédures offertes à l'exécution distante (offre aux clients). Les procédures offertes en appel call-back sont aussi précisées dans cette liste.

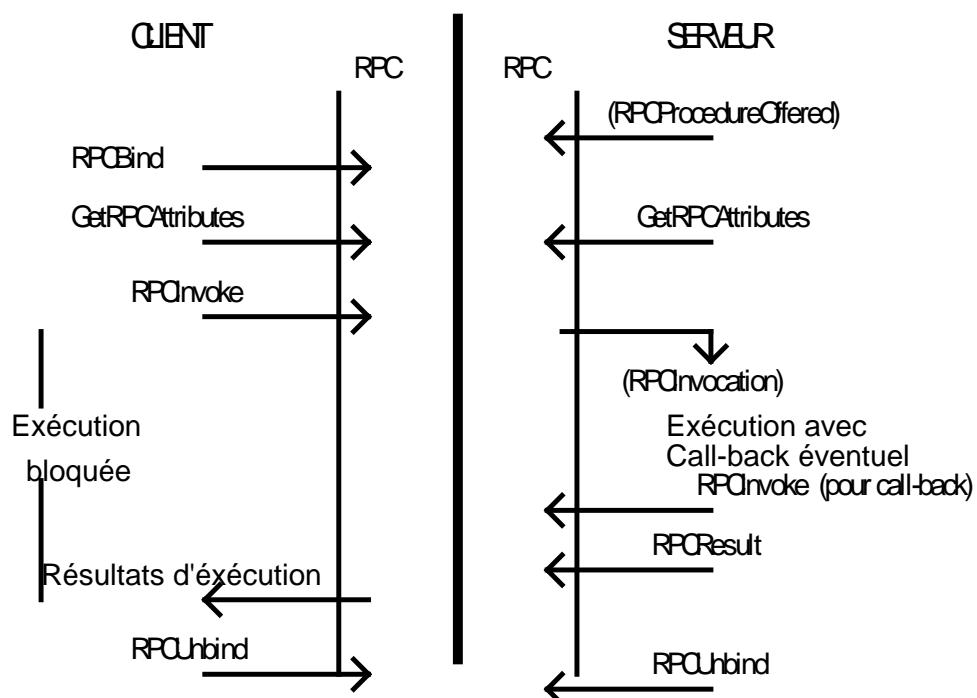
- **(RPCInvocation) :**

Cette primitive constitue l'indication d'appel, pour exécution, d'une procédure. Elle permet au RPC-ASE du serveur de demander l'exécution d'une procédure qui figure sur la liste des procédures offertes à exécution distante.

- **RPCResult (RPCReturn) :**

Permet au serveur de confier les résultats d'exécution d'une procédure à son RPC-ASE. Les paramètres de retour sont fournis à l'utilisateur dans l'échange de paramètres de la procédure appelante qui se termine.

Le schéma ci-dessous montre l'enchaînement de ces primitives.



### 13.1.5 Service d'opération distante (ROSE) nécessaire

Pour s'exécuter, le RPC défini par l'OSI ou l'ECMA s'appuie sur le serveur d'opérations distantes ROSE. RPC est caractérisé par

- son asymétrie : système client-serveur dans lequel l'initiative revient toujours au client
- le mécanisme synchrone d'exécution. Chaque appel de procédure est bloquant jusqu'à réception des résultats d'exécution. Les appels en retour (call-back) sont eux aussi bloquant pour l'exécution de la procédure distante et se déroulent sur le système client pendant que l'initiateur de l'appel original attend la fin de la procédure demandée.

Dans ces conditions, le service ROSE utilise une **classe 1 d'association** (seul l'initiateur de l'association peut invoquer des opérations) et une **classe 1 d'opérations (opération synchrone avec réponse en cas de succès ou d'échec)**. Les appels en retour (call-back) sont supportés par le mécanisme d'**opérations liées, de type parent-enfant**. L'opération résultant de l'appel originel est parent, les autres sont des opérations enfants.

## 13.2 Architectures réelles pour un RPC et exemples d'utilisation

### 13.2.1 Structure modulaire

Un élément de service Application RPC (RPC-ASE) se décompose en plusieurs modules différents sur le système client et le système serveur.

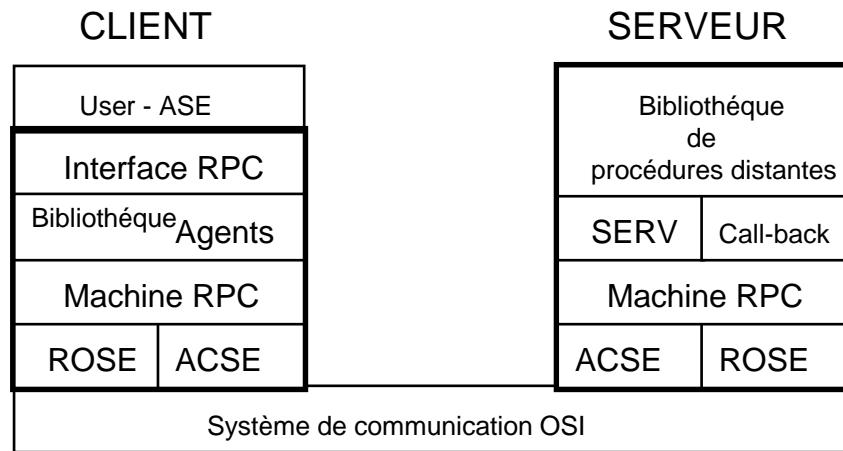
Coté client on doit trouver un module d'interface avec l'application client (User-Ase) de manière à faciliter l'accès au système depuis des applications très diverses. Ce module permet d'accéder à la bibliothèque des procédures "agents" ("Stub"). En effet à chaque procédure exécutable sur le serveur correspond une procédure locale spécifique et très brève chargée de récupérer les paramètres d'appel, de lancer le module RPC supportant le protocole proprement dit et de transmettre à l'appelant les résultats.

Sur le client et le serveur se trouvent les modules RPC traitant le protocole proprement dit. Ils doivent en particulier coder les messages en langage ASN.1 et les placer dans des opérations distantes soumises à ROSE.

Sur le serveur, le destinataire final des appels est le système d'exécution des procédures. Celles-ci sont stockées dans une bibliothèque de procédures offertes. Elles peuvent être chargées et exécutées en fonction des demandes.

Un module logiciel reçoit les demandes d'exécution du RPC-ASE serveur et réalise l'appel réel de procédure. L'interface D1 peut n'être qu'implicite si le serveur est un système spécialisé. Un module spécial doit être implanté pour gérer les appels en retour; appelé par la procédure distante il sert d'interface avec le module supportant le protocole RPC proprement dit.

Le schéma ci-dessous, donné à titre indicatif, illustre ce type d'architecture.



### 13.2.2 Exemple d'utilisation

Le serveur offre un ensemble de procédures, notamment une procédure (EigenV) permettant le calcul des valeurs et vecteurs propres d'une matrice et un procédure (Print) permettant l'impression d'un fichier d'un système client sur une imprimante supportée par le serveur.

Des primitives, non standards, ont été introduites pour synchroniser les modules lors de l'exécution. Ce sont RPCReponse, RPCWaitBind, RPCWaitInvoke.

#### 13.2.2.1 Sur le client

##### **Programme\_client**

```
"  
"  
EigenV (matrice_entrée, valeurs_propres, vecteurs_propres, status)  
"
```

```
Print (Fichier_matrice, status)  
"
```

##### **User-ASE**

```
"  
"  
Interface_RPC (EigenV, matrice_entrée)  
"  
Interface_RPC (Print, Fichier_matrice)  
"
```

##### **Interface RPC**

```
"  
Appel Agent_EigenV (matrice_entrée)  
"
```

```
Appel Agent_Print (fichier_matrice)
```

**Agent\_EigenV**

```
begin
RPCBind()
RPCInvoke (Operation1,...)
    RPCReponse ()
RPCUnbind
```

```
end
```

**Agent\_Print**

```
begin
RPCBind
RPCInvoke (Operation8,...)
    RPCReponse ()
While call_back
    Appel_opération_enfant_1      ; ouvrir fichier
    Appel_opération_enfant_2      ; lire fichier
    Appel_opération_enfant_3      ; fermer fichier
    RPCResult(...)
        RPCReponse ()
Endwhile
RPCUnbind ()
end
```

**Table des opérations locales**

Opérations enfants	Entrée
1	Open_in
2	Read_Buffer
3	CloseFile

### 13.2.2.2 Sur le serveur

**EigenV**

```
Calcul (matrice_entrée, valeurs_propres, vecteurs_propres)
begin
```

```
end
```

**Print**

```
Printfile (fichier)
begin
appel_call_back (Num_oper_enfant_1)
appel_call_back (Num_oper_enfant_2)
```

```

appel_call_back (Num_oper_enfant_3)
end
Serv
main ()
RPCOffer ()
While (en activité)
RPCWaitBind
RPCWaitInvocation
exécuter procédure associée à numéro d'opération transmis par ROSE
RPCResult
RPCEnd
EndWhile
RPCEndOffer

```

### Call-back

```

begin
RPCInvoke (Num_oper_enfant)
RPCReponse (...)

end

```

### Opérations demandées et procédures

Procédure	Opération	Entrée
EigenV	1	Calcul
S_fichier	2	Open_in
S_fichier	3	Open_out
Print	8	Printfile

Les modules sont en fait réalisés par des procédures dans lesquelles sont appelées les procédures de niveau inférieur.

#### 13.2.3 Implantation sous Unix sur un service de niveau 4/OSI

RPC, en milieu (quasi)-homogène, est implanté normalement au dessus du service Transport, que ce soit un transport OSI (classe 4) ou un service TCP. Cependant on utilise le plus souvent le **protocole sans connexion UDP** de l'ensemble Inet.

Dans les nouvelles versions d'Unix, il utilisera l'interface standard XTI.

L'ensemble RPC est constitué de 3 sous-couches. La sous-couche supérieure est complètement transparente au programmeur. Elle sert à administrer le RPC et par exemple à déterminer le nombre d'utilisateurs à un instant donné.

La couche intermédiaire est accessible directement depuis le programme utilisateur. Dans les bibliothèques Unix sont fournies des fonctions permettant de mettre en oeuvre le RPC. La plus simple est callrpc ( ) qui exécute un appel de procédure distante. La fonction registerrpc() fournit un numéro unique sur tout le système pour identifier la connexion client-serveur établie.

La couche inférieure est utilisée pour des applications plus sophistiquées qui demandent la modification de certains paramètres du RPC fixés par défaut.

Si les données doivent être converties en langage de transfert XDR, on utilise les processus de sérialisation appelés par des procédures du type `xdr_int ()`, `xdr_long ()`, `xdr_short ()`, `xdr_string ()`, etc. Ces procédures sont utilisées dans une procédure `xdr_simple ()` qui sera placée dans les paramètres d'appel de `rpccall ()`

Par exemple :

```
callrpc (hostname, PROGNUM, VERSNUM, PROCNUM, xdr_simple, &simple,...)
```

`xdr_simple` appelle dans son code `xdr_int`, `xdr_short` ou `xdr_long` par exemple.

Côté serveur s'effectue la désérialisation. Le logiciel lance le module de transport (par exemple UDP) et se met en attente d'attachements (Bind) d'un utilisateur. Suite à un `rpccall`, il enregistre la demande, fait un appel service pour réaliser la procédure demandée. Si la procédure a abouti normalement, qui renvoie la réponse à l'appelant.

Le serveur de ressources NFS (Network Files System) utilise RPC.

Dans un proche avenir, ce type de protocole devrait être utilisé de manière systématique pour traiter la majeure partie des applications réparties sous Unix.

# 14 COUCHE 7/OSI : PROTOCOLE d'APPAREIL VIRTUEL

NOTA : Le chapitre ci-dessous illustre les concepts d'appareil virtuel mais ne tient pas compte des développements récents dans ce domaine. Il est rédigé d'après une version préliminaire de la norme OSI. L'exemple fourni, terminal Vidéotex, est donné pour illustrer ces principes.

## 14.1 Applications interactives et Applications de transfert d'information pour restitution.

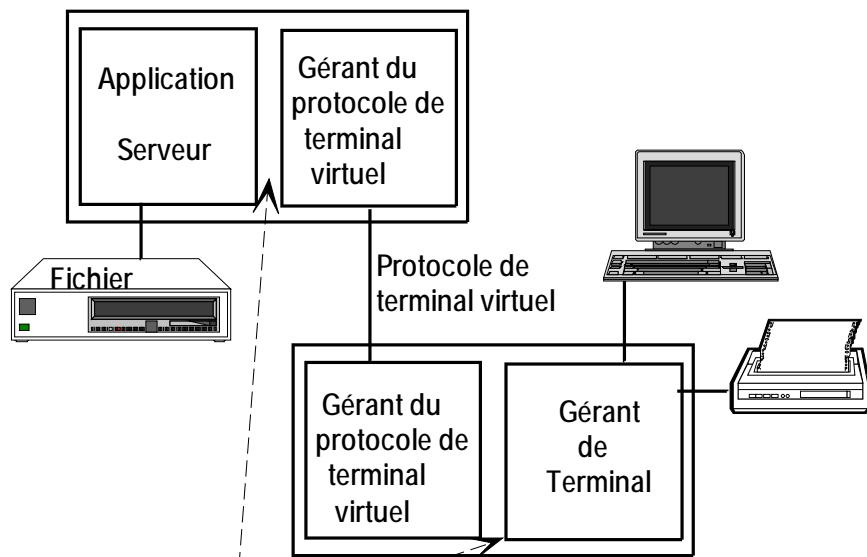
Les applications interactives sont constituées de transactions élémentaires composées chacune d'une "question" et d'une réponse. La question peut être composée en texte libre ou à l'aide d'une grille d'écran. Les échanges se font en temps réel.

Les applications de transfert d'information pour restitution s'échangent des quantités, parfois importantes, d'informations sans intervention de l'usager durant le transfert. Cette information est ensuite restituée sous une forme compréhensible par l'usager. Cette restitution peut se faire en temps différé quand le document est complètement transféré.

Ces deux types d'application ont en commun le problème de la restitution des données sur un terminal : écran ou imprimante. Il est nécessaire de **masquer à l'usager la diversité des terminaux** et de la manière avec laquelle ils réalisent une fonction donnée. Pour cela la Présentation définit un protocole d'appareil virtuel qui décrit le langage commun (syntaxe de transfert) aux entités de Présentation indépendamment des terminaux réels ou du système de stockage des données.

Les données échangées peuvent être **textuelles** et/ou **graphiques**. On pourra utiliser la syntaxe X409 pour décrire formellement la structure des données échangées.

Le schéma ci-dessous illustre ce service d'appareil



## 14.2 Modèle conceptuel

Ce modèle conceptuel définit une représentation théorique d'un terminal, représentation commune aux entités interconnectées. Chaque application traduit ce modèle dans ses propres, structures. Ce modèle fournit une représentation commune de l'image à visualiser, à stocker ou à transmettre ainsi que de l'état du terminal et des moyens de signalisation : curseur(s) ou réticule(s).

La syntaxe de transfert utilisée va permettre de décrire ce modèle conceptuel d'une manière identique pour les deux entités de présentation qui communiquent. Chaque entité d'application en aura une représentation particulière adaptée au terminal ou au système de fichiers.

### 14.2.1 Définitions

*Objet atomique* : Plus petite unité de données adressable séparément exemples : caractère, pixel, bit, objet graphique

*Attributs* : Propriétés associées à un champ de données ou un groupe de champs, éventuellement un objet atomique.

*Cellule* : Plus petit élément de **stockage** adressable séparément. Elle contient un objet atomique.

*Image conceptuelle* : Contenu d'information d'une zone de stockage de données tel qu'il est vu par les deux utilisateurs.

*Aire de communication conceptuelle* : CCA Zone de dépôt conceptuelle pour les deux entités qui échangent des données ou des commandes.

*Zone de données conceptuelle* : CDS

Sous-ensemble de la CCA où sont stockées (virtuellement) les données par les deux utilisateurs.

**Chaque utilisateur doit pouvoir "visualiser" l'image conceptuelle contenue dans la CDS et consulter ou modifier les différentes zones de la CDA.**

*Zone de signalisation* : SIDS

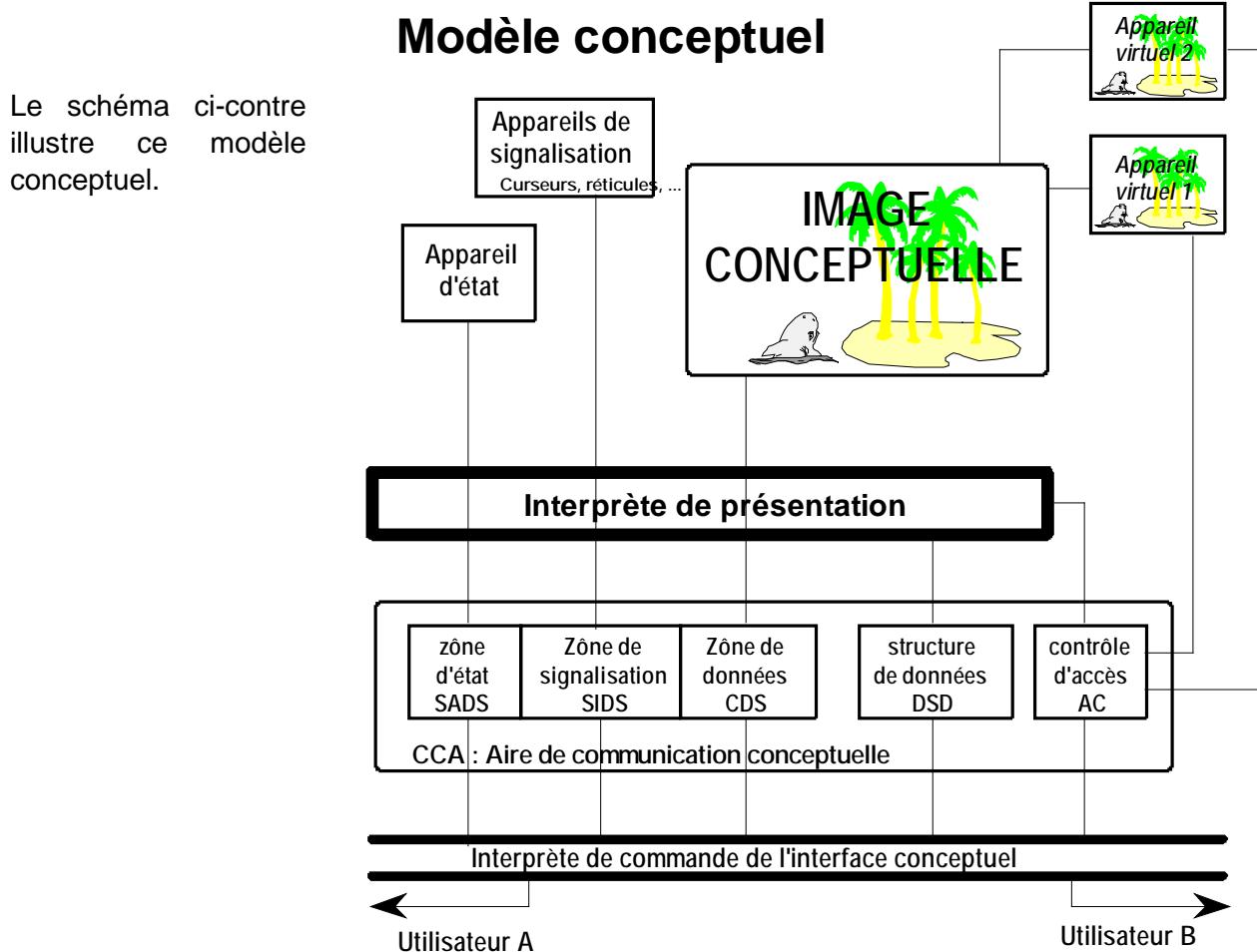
Sous-ensemble de la CCA où est consignée la position des moyens de signalisation : curseurs ou réticules

*Zone d'état* : SADS

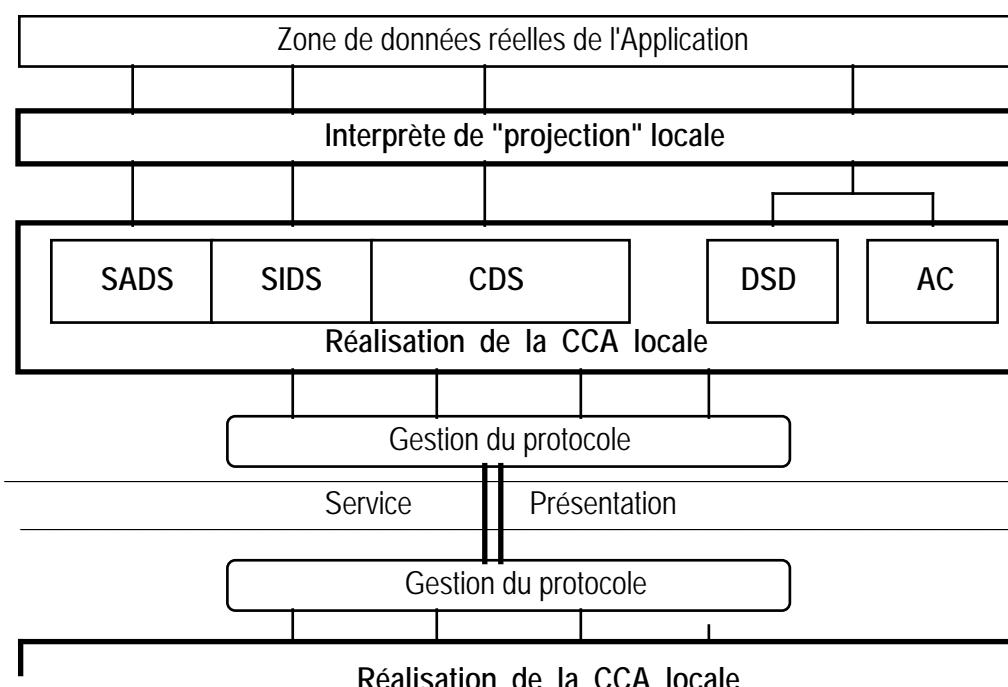
Sous-ensemble de la CCA qui contient les variables d'état du terminal virtuel.

Le modèle conceptuel contient aussi des fonctions gérant le *contrôle d'accès* (AC ) et un *ensemble de structures de données types* ( DSD ) servant de modèle de données.

### 14.2.2 Structure logique et implantation



Pour chaque application on aura une implantation spécifique selon le schéma de principe ci-dessous :



## 14.3 Exemple ancien : Appareil virtuel Architel

L'exemple ci-dessous illustre, pour un terminal simple "**Minitel**", la notion de protocole d'appareil virtuel et de modèle conceptuel. Pour plus de détails sur ce système on se rapportera au cours sur le Vidéotex.

Pour les applications de type Vidéotex ou Télétex, l'administration française des Télécommunications a défini un protocole d'appareil virtuel à implanter dans les serveurs ou dans les concentrateurs de terminaux.

Ce protocole d'appareil s'appuie sur un service Session OSI de type BAS.

Il permet une composition des images souple et **modulaire** qui facilite la construction d'une image par assemblage de sous-images et sa modification partielle.

Les applications traitent et échangent des informations logiques sous forme de chaînes de caractères qui seront interprétés avant visualisation. L'unité logique d'information à visualiser est le **champ**.

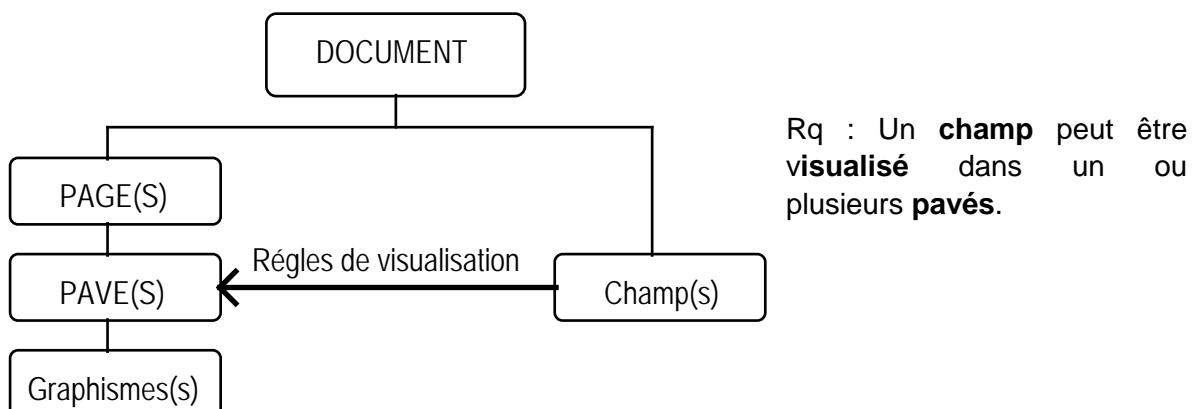
### 14.3.1 Structure de document

L'objet le plus global manipulé par le protocole est le **Document**.

L'image visualisée à un instant donné sur le terminal constitue une **page** de ce document.

Pour faciliter les traitements, les stockages ou les transferts, le document est découpé en **champs**. Le champ est l'**objet de base** de la structure logique. Il est modifiable par l'opérateur en saisie ou par l'application en réponse à une question.

Dans sa structure d'images, le document est découpé en pages; chaque page est construite à partir de **pavés** qui constituent les sous-images. Un pavé est une surface de visualisation rectangulaire, aux cotés parallèles à ceux de l'écran, et de dimensions quelconques. Le contenu d'un pavé est constitué de **graphismes** qui portent l'information utile.



### 14.3.2 Signalisation

Pour manipuler un objet il faut pouvoir le désigner soit dans sa structure physique (structure d'images) soit dans sa structure logique qui permet d'échanger des informations entre utilisateur et application.

Dans la structure physique :

on utilise un **curseur** composé de quatre éléments :

- pointeur de document
- pointeur de page
- pointeur de pavé
- pointeur de graphisme

Dans la structure logique :

on utilise un **pointeur de champ**.

A un instant donné, l'application peut accéder soit à la structure d'image par le curseur, soit à la structure logique par le pointeur de champ.

### 14.3.3 Objets et modèles

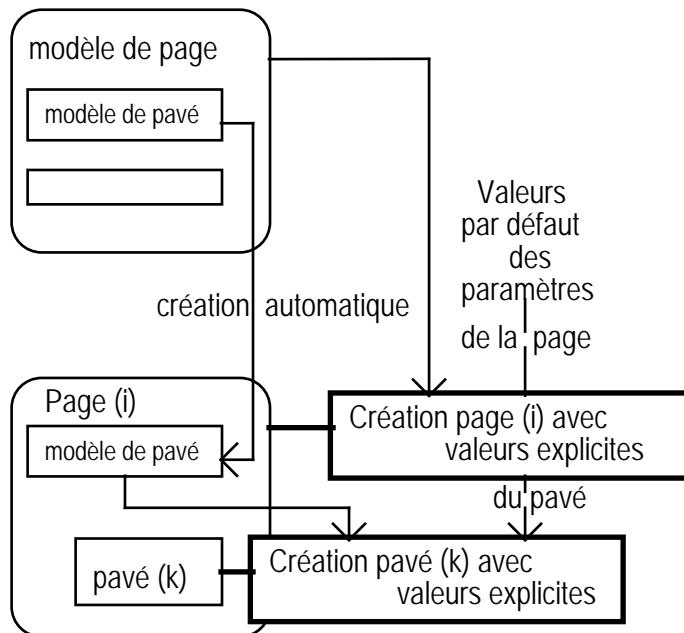
Habituellement, les terminaux, simples (vidéotex), sont connectés à un concentrateur de terminaux situé à l'entrée du réseau que l'on nomme "**point d'accès**".

Pour diminuer les volumes de données à transférer, le point d'accès dispose d'une bibliothèque d'**objets téléchargeables** par le Serveur.

- Un tel objet n'est transféré qu'une seule fois sous forme d'éléments de protocole
- Il peut être utilisé directement ou modifié plusieurs fois par des commandes envoyées depuis le Serveur.

Chaque objet manipulable ( page, pavé ) comporte un ensemble de paramètres qui lui sont propres. Ainsi un pavé est caractérisé par une couleur de fond, une couleur de graphisme, un modèle de graphisme, des dimensions. L'ensemble des valeurs les plus utilisées pour les paramètres d'un objet est regroupé dans un **modèle**. Pour créer un nouvel objet on pourra copier ce modèle ou en modifier un ou plusieurs paramètres (par exemple uniquement la valeur de fond). Il faudra aussi donner une valeur explicite aux graphismes pour visualiser l'information.

Le schéma ci-dessous illustre ces mécanismes.



Un modèle de document est constitué de pages modèles.

Un modèle de page regroupe des modèles de pavés. Ainsi à la création d'un objet, on hérite des modèles de ses objets constitutifs qui sont créés automatiquement avec leurs paramètres par défaut. On pourra éventuellement les modifier.

De même, la présence d'un modèle d'objet à un niveau de structure permet de créer automatiquement les objets de niveaux supérieurs s'ils ne sont pas déjà décrits explicitement.

La modification d'un document permet d'ajouter une page en fin de liste, de modifier les éléments variables de modèle de page et le contenu des pages de document. La modification d'une page permet d'ajouter un pavé en fin de liste, modifier les éléments variables du modèle de pavé et de modifier le contenu des pavés.

La modification du contenu d'un pavé ne porte que sur le modèle de graphisme et la suite des graphismes.

La modification du contenu d'un champ ne peut porter que sur la suite de graphismes répartis dans la suite de pavés qui constituent le champ.

#### 14.3.4 Programme de saisie

Pour diminuer la charge du serveur le programme de saisie est déporté au point d'accès. Il permet de rendre le traitement des informations indépendant de la conduite de la saisie sur le terminal.

L'objet de base de la structure logique étant le champ, la saisie se fait champ par champ ; chaque champ est traité par un sous-programme de saisie.

L'état global de la saisie (active ou non, rang du sous-programme en cours) est rangé dans un objet manipulable par la Présentation: le compteur de saisie.

Chaque sous-programme comporte des consignes générales et un pilote de saisie définissant les actions à effectuer selon les événements reçus (touches de fonctions). ces actions peuvent être locales ou provoquer le transfert d'information vers le serveur. Le programme de saisie devient alors inactif. (ainsi le transfert est semi-duplex :TWA au niveau Session).

---

## 15 Couche 7/OSI : SYSTEMES TRANSACTIONNELS REPARTIS OSI-TP

### 15.1 Introduction : le traitement transactionnel

Le traitement transactionnel en ligne (OLTP pour On Line Transaction Processing) est un mode d'organisation d'une application informatique qui permet à un grand nombre d'utilisateurs de soumettre, via leurs terminaux, des transactions à un système qui doit les traiter le plus rapidement possible et répercuter leurs effets sur une grande base de données.

Ce concept, apparu au début des années 70 est en cours d'évolution et doit intégrer désormais les nouvelles données du marché informatique à savoir :

- l'émergence des plates-formes matérielles autour des micro-ordinateurs ou de stations de travail performantes
- l'apparition de standard autour de normes OSI (Open System Interconnection) ou X/OPEN

#### Exemples de champs d'applications

Le premier domaine d'application concerné par le traitement transactionnel fût celui de la réservation aérienne, et par la suite, on a assisté à une croissance importante des applications "OLTP" qui couvrent aujourd'hui des domaines aussi variés que :

- la réservation dans les domaines du transport et de l'hôtellerie
- la banque et la finance pour les opérations clientèles, les ordres de bourse et la monétique
- l'assurance pour la gestion des contrats en temps réel
- les grandes administrations pour certaines applications caractéristiques comme l'interrogation de fichiers centraux
- la distribution pour les applications tournant autour de la prise de commande ou de la gestion des stocks
- l'industrie dans le cadre d'applications de pilotage de production

Cette attraction qu'exerce le traitement de transactions en ligne sur les entreprises est due aux possibilités d'accès et de modification en temps réel, permettant aux employés d'être au courant des changements constants qui affectent l'entreprise, ce qui augmente leur productivité.

On dit généralement que le système global (les gestionnaires de transactions et de base de données) doit permettre d'assurer l'intégrité transactionnelle des données modifiées. Dans ce contexte, le moniteur transactionnel s'occupe de l'acheminement de la transaction vers le processus qui doit la traiter d'une part et dialogue avec le gestionnaire de base de données qui doit répercuter le résultat de cette transaction dans sa base d'autre part. Le problème important à évoquer est le traitement des transactions réparties, c'est à dire celles qui impliquent l'accès et la mise à jour de plusieurs bases de données distribuées sur plusieurs machines. Pour résoudre les problèmes liés aux défaillances des systèmes ou réseau, lorsqu'une transaction répartie est en cours de traitement, le moniteur transactionnel s'appuie sur le protocole de validation en deux phases (two phases commit). Dans les pages suivantes nous donnons plus d'informations sur ce protocole.

### 15.1.1 Caractéristiques des applications transactionnelles.

Ce service est adapté à des applications présentant les caractéristiques suivantes :

- Traitements courts et répétitifs
- Grand nombre d'utilisateurs simultanés
- Nombre important de transactions à traiter simultanément, en parallèle
- Temps de réponses brefs et assez constants, de l'ordre d'une à quelques secondes
- Accès concurrent à des bases de données
- Volumes de données importants
- **Grande fiabilité et haute disponibilité**

Ces caractéristiques ne se retrouvent pas seulement dans des applications du secteur tertiaire mais correspondent aussi à des besoins de coordination dans les applications de productique, pour la coordination des tâches dans les systèmes intégrés de fabrication (CIM).

Les grands constructeurs puis les organismes de normalisation du monde Unix ont introduit ces concepts dans leurs architectures distribuées. Ainsi l'X/OPEN group a développé un modèle d'architecture pour le traitement transactionnel distribué : DTP (Distributed Transaction Processing). L'OSF a aussi introduit le traitement distribué dans le modèle DCE (Distributed Computing Environment) avec le système OLTP (On Line Transaction Processing)

### 15.1.2 La transaction

#### 15.1.2.1 Définition : propriétés ACID

Une **transaction** est un ensemble d'opérations apparentées caractérisé par quatre propriétés:

- Atomicité
- Consistance
- Isolation
- Durabilité

Ces propriétés sont appelées **propriétés ACID**.

L'**atomicité** implique que les opérations en rapport dans une transaction soient toutes exécutées ou qu'aucunes ne le soient.

La **consistance** implique que les effets des opérations apparentées dans une transaction soient effectuées correctement, précisément et **avec validité**, en respect avec la sémantique de l'application. Les données manipulées par une transaction passent d'un état consistant à un autre état consistant.

L'**isolation** implique qu'à part les opérations de la transaction considérée, aucune opération ou fonction d'une autre transaction ne puisse accéder aux résultats partiels obtenus. **Cette définition implique que des transactions qui travaillent sur des données communes** (à leur interface) **soient sérialisées**. Cette propriété est assurée par un contrôle de **concurrence** qui garantit qu'une action atomique ne peut être validée tant que d'autres actions atomiques qui en ont changé la valeur ne sont pas validées et qu'aucun changement

n'est fait sur une donnée lors de son utilisation par l'action atomique, excepté par les branches de celle-ci.

La **durabilité** implique que les résultats d'une transaction terminée (complète) ne peuvent en aucune manière être altérés, par exemple en cas de défaut ou de panne. Pour annuler l'effet d'une transaction il faut exécuter une transaction de compensation. Une restauration (Recovery) assure la progression correcte d'une action atomique en cas de panne. Elle est basée sur des mécanismes de synchronisation et de **sauvegarde (log)** des données.

Des **données sûres** sont des données qui survivent à une panne d'une application et sont disponibles à l'entité d'application (du logiciel de communications transactionnel) après que des procédures de reprise locales l'ait restaurée. Des données liées sont des données sûres qui existent durant que la transaction s'exécute. Les données d'une action atomique sont des données sûres qui maintiennent:

- la connaissance qu'une action atomique est en cours.
- l'état du contrôle de concurrence.
- l'information permettant de restaurer toute donnée dans son état initial.

La **journalisation (log)** est un événement correspondant à l'écriture, sur un support sûr, des données spécifiques à une action atomique. Elle permet la reprise consistante et la terminaison d'une action atomique interrompue par une panne.

### 15.1.2.2 Transactions réparties : arbre de dialogue

Une transaction qui se déroule sur plusieurs systèmes ouverts est appelée une transaction répartie. Pour maintenir les quatre propriétés d'une transaction répartie, une coordination est nécessaire entre les entités de chaque système chargées de son traitement. Cette coordination requiert des communications entre ces entités. Dans le service transactionnel OSI, ces entités sont nommées TPSUI : Transport Processing Service User Invocation. Ce sont des instances particulières d'une entité de service (TPSU), ensemble spécifique de capacités de traitement dans un processus d'application.

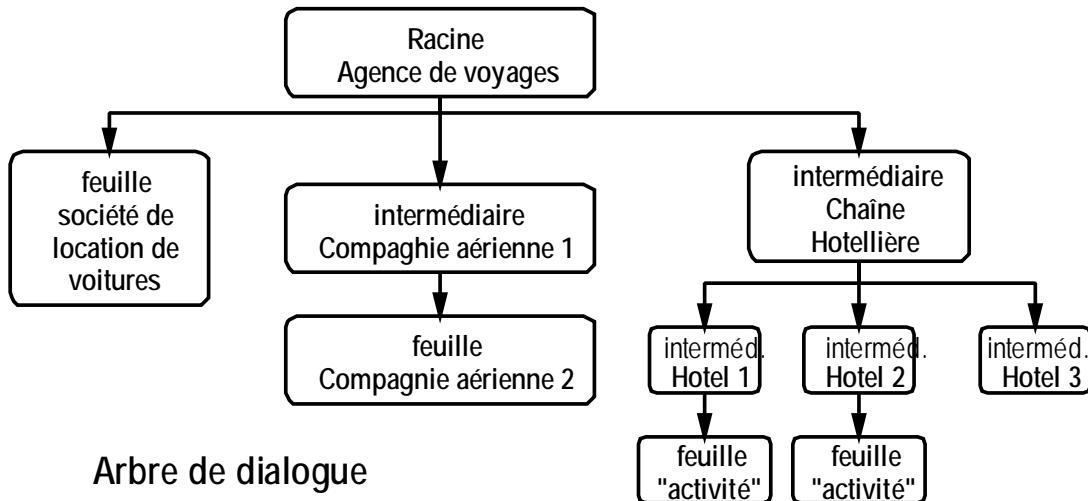
Ces entités interagissent par des communications point-à-point dénommées dialogues. Par ces dialogues les entités peuvent assurer :

- le transfert de données
- la notification des anomalies
- l'initialisation, la validation ou l'annulation (avec remise à l'état précédent : rollback) d'une transaction
- la terminaison ordonnée ou brutale du dialogue
- La synchronisation des activités pour atteindre un point de traitement que les entités peuvent mutuellement agréer. Ceci est supporté par un service d'acquittement disponible pour la durée d'un dialogue.

Un dialogue peut se dérouler selon deux modes de commande:

- polarisé (initiateur-répondeur), dans lequel seule une entité possède le contrôle du dialogue (sauf pour la notification des anomalies, l'annulation d'une transaction ou sa terminaison brutale)
- partagé, dans lequel les deux entités commandent simultanément le dialogue.

Une entité quelconque peut créer plusieurs dialogues avec des entités distinctes. Ces dialogues sont dits adjacents. Un arbre de dialogue est une structure arborescente de dialogues adjacents. Dans un arbre de dialogue, l'entité qui établit un dialogue est appelée "supérieur direct" et l'entité appelée "subordonné direct". L'entité qui n'a aucun supérieur est appelée "racine". Une entité n'ayant aucun subordonné est une "feuille". Les autres entités sont des "intermédiaires".



La partie d'une transaction distribuée réalisée par une paire d'entités partageant un dialogue est appelée "branche de transaction". Le niveau de coordination détermine si un service de validation des transactions est utilisé ou non, c'est à dire qui est responsable, de l'application ou du système de gestion de transactions, du maintien des propriétés ACID. Dans un système de niveau de coordination "validation" (commitment) on peut définir un arbre de transaction dont la racine assure le rôle de "coordinateur de validation" ou "coordinateur d'engagement".

Parfois enfin, notamment durant les phases de reprise (recovery) les systèmes utilisant les transactions peuvent avoir à communiquer directement, sans implication des entités supportant les transactions. Ceci est réalisé par les "canaux transactionnels".

### 15.1.3 Déroulement d'une transaction

Le système qui débute la transaction est le nœud racine de l'arbre de transaction ie. le coordinateur d'engagement. Un identificateur alloué à la transaction sera propagé sur tout arbre de transaction. Il peut être un critère d'allocation exclusive d'une ressource à la transaction. et ainsi permettre à chaque nœud de garantir l'**isolation** de la transaction.

Durant le déroulement d'une transaction distribuée, un des systèmes peut se trouver dans l'impossibilité de réaliser le travail qui lui est assigné. Cette *situation d'annulation* (Rollback) est communiquée aux autres noeuds pour qu'ils cessent leurs travaux afin de libérer les données liées à la transaction (données liées) dans leur état initial. Tant que la transaction n'est pas entrée dans sa phase terminale, une annulation peut être effectuée sans difficulté. La solution choisie (avortement présumé) entraîne la reprise à l'état initial. une solution plus sophistiquée, utilisant des points de reprise intermédiaires pourrait être ajoutée.

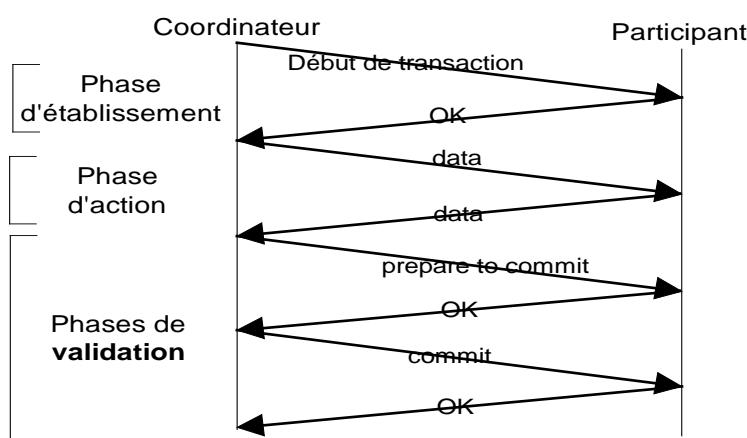
Pour assurer la **consistance** de la transaction un mécanisme de **validation à deux phases** (two phases commit) doit être implanté. Il se déroule de la manière suivante :

- le coordinateur envoie à ses partenaires une requête de préparation.
- chaque participant doit répondre en acceptant (*ready*) ou en refusant de poursuivre la transaction (*rollback*). Un système qui accepte la transaction est capable d'assurer l'écriture des données en mémoire stable (pour permettre la durabilité).
- Si tous les systèmes ont signalé au coordonnateur qu'ils acceptent la transaction celui-ci leur enjoint de libérer les données dans leur état final (*commit*).

Si l'un des participants a refusé (*rollback*) ou s'il a détecté une panne, le coordonnateur répercute cette requête vers tous les participants qui remettent leurs données dans l'état initial.

- Chaque système, quand il a reçu l'ordre d'engagement (*commit*) met ses données dans l'état final en lieu sûr (**durabilité**) et signale au coordinateur l'achèvement de son rôle.

Suivant le rôle joué par le système les 2 phases : "prepare" et "commit" ne se situent pas au même moment. Durant la seconde phase **tout système doit stocker dans un journal (/log)**



**Figure 1.** L'échange protocolaire dans la validation à deux phases

les diverses informations concernant l'état des données, les décisions prises, les partenaires avec lesquels il coopère. Ce journal ne doit pas être affecté par une panne; enfin de cette seconde phase, les informations devenues inutiles seront éliminées.

Un participant qui a accepté la transaction (*ready*) ne peut plus avorter car cela pourrait entrer en collision avec la réception de la requête

d'engagement venant du coordinateur. Durant la première phase, s'il détecte une rupture de la communication, il doit initialiser une reprise (nouvelle association) et répéter son acceptation (*ready*). Il attend alors la décision du coordonnateur.

Enfin des défaillances peuvent survenir durant la transaction, en particulier durant la (seconde) phase d'engagement.

Différentes défaillances ont été répertoriées: défaillance d'un programme d'application, défaillance des communications (rupture), faute dans le protocole, crash système, défaillance des moyens de stockage, verrous mortels.

Une défaillance se produisant durant la première phase provoque une annulation avec une clause d'avortement présumé.

Une défaillance durant la seconde phase pose des problèmes beaucoup plus graves et complexes.

On distingue 3 étapes dans la reprise:

- Lancement du processus de reprise après détection d'une défaillance

On peut décider l'arrêt du dialogue, une annulation ou une reprise.

- Reprise de la transaction après redémarrage du composant défaillant

On cherche à rétablir un canal avec l'entité distante et à terminer l'engagement.

- Reprise du dialogue quand la consistance est rétablie

On cherche à rétablir le dialogue.

Il est alors possible de reprendre l'opération normale.

#### 15.1.4 Validation à deux phases

Un des problèmes majeurs dans les systèmes répartis est de distribuer les données et les ressources sur l'ensemble des machines connectées et de pouvoir assurer l'intégrité de ces données. Si, accéder à une base de données unique à partir d'une machine et assurer sa cohérence est assez simple à réaliser, accéder, dans une même opération de mise à jour, à plusieurs fichiers situés dans des bases différentes, est plus délicat. La raison en est que l'opération de mise à jour doit être atomique. A ce jour, le seul algorithme reconnu qui garantit la cohérence de toutes les données, avant et après l'opération de mise à jour, même dans le cas d'arrêt brutal, panne d'une machine ou du réseau est l'algorithme de "two-phase-commit" (validation à deux phases).

Lorsque plusieurs SGBDs sont impliqués dans une transaction répartie, chaque SGBD doit assurer la cohérence des données et ceci même en cas de défaillance de la machine ou du processus qui exécutait la transaction. Pour cela, il utilise un journal de transactions où il conserve l'identifiant unique de la transaction (le numéro unique est attribué par l'initiateur de la transaction), la valeur d'item avant et après modification, les participants, le coordinateur de transaction, etc, ... En pratique, on dispose soit d'un journal "avant" soit d'un journal "après" suivant le choix qui a été fait par le concepteur. Les valeurs des items avant et après modification sont écrites seulement dans le journal qui appartient à la base où se trouvent les items (AFUU 89). A ce niveau, on introduit le concept de validation en deux phases avec les règles suivantes :

- Une transaction ne peut pas être écrite dans la base tant que le SGBD n'a pas reçu l'ordre de validation (commit).
- Une transaction ne peut pas être validée tant qu'on n'a pas enregistré toutes les modifications des items dans le journal (dans le cas d'une seule base de données, l'organisme de standardisation X/OPEN suggère pour des raisons d'optimisation, la validation en une seule phase).

Ainsi, la première phase est toujours l'écriture dans le journal et la deuxième phase est toujours l'écriture dans la base.

Cette procédure répartie requiert qu'un des participants (l'initiateur) à la transaction coordonne les échanges avec tous ses partenaires. Ce participant particulier, le coordinateur, et tous ses partenaires sont chacun doté d'un journal, comme mentionné ci-dessus. Pour y conserver des informations utiles pour la reprise éventuelle, quatre vagues successives (BARBOT 91) d'informations sont échangées :

- 1- Le coordinateur signale à ses partenaires le début de validation en deux phases en envoyant l'ordre de "prepare to commit" à ces derniers et appelle à voter.
- 2- Chaque partenaire peut voter "oui" ou "non". Un vote positif indique que le partenaire s'est assuré qu'il pourra, sur décision finale du coordinateur, annuler l'opération effectuée ou entrer en phase terminale (mise à jour effective de la base). Un vote

négatif est notifié dans le cas contraire. Le coordinateur note le résultat positif du vote dans son journal.

- 3- Les votes sont collectés par le coordinateur. Si ses partenaires et lui-même votent "oui" à l'unanimité, le coordinateur ordonne la mise à jour effective de la base en leur envoyant l'ordre de "commit". Dans le cas de réponse positive des participants, le coordinateur note la décision dans son journal. Un seul vote négatif provoque l'annulation de la transaction et dans ce cas, le coordinateur demande à ses partenaires d'annuler les opérations effectuées jusqu'alors en leur envoyant l'ordre "abort".
- 4- Chacun des participants s'assure de la fin de transaction, en notifie le coordinateur, et note dans son journal le résultat. Le coordinateur, sur réception de toutes les confirmations de validation note également la fin de la transaction.

La procédure de reprise éventuelle s'applique après qu'une panne ait rompu la communication entre deux partenaires. Elle se présente différemment pour le coordinateur et ses partenaires.

Pour le coordinateur, la procédure de reprise est déclenchée sur détection de panne après la première phase. Le coordinateur trouvant dans son journal la trace du vote positif sans que la fin de transaction ait été enregistrée, entre en contact avec des participants pour lesquels il n'a pas reçu la notification finale. Le coordinateur informe ces derniers et la validation à deux phases reprend en 4).

Pour un participant, la procédure de reprise s'applique lorsqu'après avoir voté il s'aperçoit d'une rupture de communication avec le coordinateur, il trouve dans son journal la trace de son vote sans celle de la fin de transaction. Le participant prend alors contact avec le coordinateur afin que ce dernier lui communique le résultat du vote. Si le journal du coordinateur ne comporte aucune trace de cette transaction, c'est que le résultat du vote est négatif et les données ne doivent pas être mise à jour. Si non, le coordinateur et le participant, après communication du résultat du vote reprennent la validation en phase 4).

Pour que le protocole "two phase commit" s'exécute correctement dans tous les cas, il doit être associé avec le protocole "two phase locking" (verrouillage en deux temps). Le verrouillage en deux temps impose qu'aucun déverrouillage n'intervienne avant que tous les ordres de verrouillage soient exécutés après la validation de transaction. Ainsi aucune transaction n'aura accès à des données modifiées mais pas encore validées issues de transaction en cours ou annulées. Ainsi le protocole "two phase commit" garantit la cohérence des données dans tous les cas qui peuvent se présenter.

## 15.2 Modèles d'échanges entre applications

Trois grands modèles d'échanges entre applications permettent de bâtir des systèmes distribués:

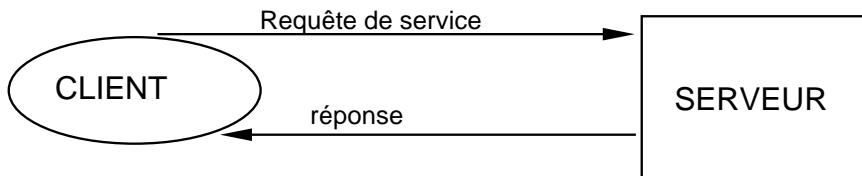
- Conversationnel (APPC)
- Client-serveur (RPC)
- Queue de messages (MQ)

### 15.2.1 Modèles client-serveur

Ces modèles sont basés sur des échanges "Requête-Réponse" et sont supportées, en général, par un système d'appel de procédures distantes (RPC : Remote Procedure Call)

### 15.2.1.1 Modèle client / serveur de base

Un modèle qui peut s'intégrer dans le traitement transactionnel en ligne est celui du client/serveur. Dans cette architecture, un client demande un service qui est assuré par un serveur. Puisque le client et le serveur sont des modules logiciels, une même machine peut les traiter. La multiplication des réseaux (locaux ou étendus) permet d'envisager le développement des applications distribuées sur les différentes unités de traitement connectées.



**Figure 2.** . Le modèle client/serveur

Les avantages liés à ce modèle sont multiples. Parmi ceux-ci on peut citer :

#### **Modularité:**

L'application peut avoir une forme modulaire. Dans ce cas, pour le développeur d'applications, le travail est plus simple, et la maintenance plus facile.

#### **Facilité de distribution:**

La facilité de distribution permet de placer les processus clients, au plus près des utilisateurs, et les serveurs peuvent être distribués sur les différents noeuds du réseau.

#### **Extensibilité:**

L'architecture modulaire permet d'étendre l'application. Dans ce cas, l'extension de l'application est possible sans perturber les parties existantes.

### 15.2.1.2 Le Modèle client / serveur enrichi

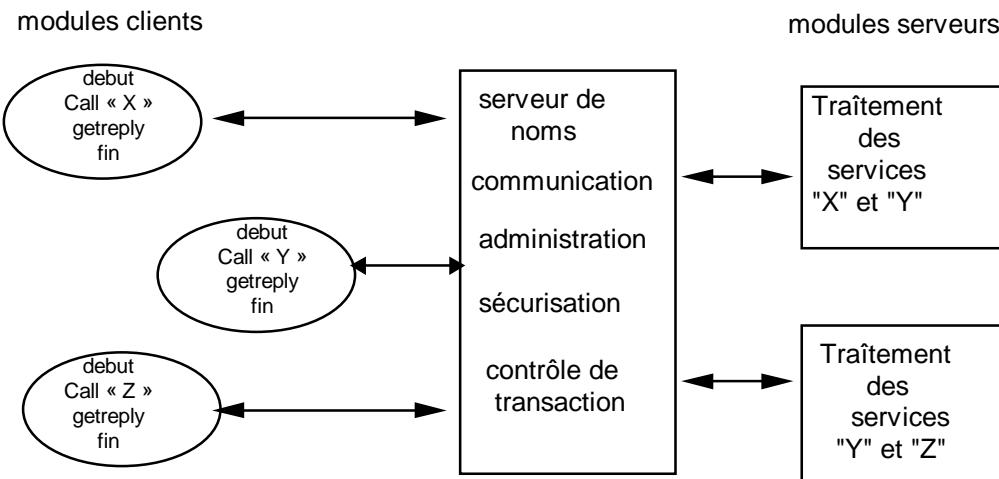
Le modèle client/serveur ci-dessus suggère une relation "un pour un" entre les processus client et serveur. Ceci ne représente pas une architecture optimale. Il entrave la capacité de partage des ressources.

Le processus client doit connaître les informations concernant le serveur et oblige ce dernier à avoir la capacité de satisfaire toutes les requêtes du client. Pour surmonter ces faiblesses (BERSON 94), les systèmes transactionnels proposent une version améliorée : le modèle client-serveur enrichi. Ce modèle permet de placer un serveur de noms entre le client et le serveur.

Cette nouvelle architecture présente les caractéristiques suivantes :

- Un serveur est constitué d'un ensemble de services (1 à n); un service est un module écrit pour exécuter une tâche unique de l'application
- Les serveurs sont des processus actifs en attente de traitement des requêtes des clients
- Un serveur peut offrir plusieurs services et un service peut être offert par plusieurs serveurs
- Au lieu d'activer un processus serveur, le client s'attache au serveur de noms avec une requête pour un service donné
- Le serveur du nommage met en correspondance le nom logique et l'adresse physique du serveur qui peut exécuter la requête de service

- En contrôlant les services assemblés dans le serveur de nommage, l'administrateur du système peut étendre ou diminuer dynamiquement les ressources disponibles
- La séparation nette des modules clients par rapport aux modules serveurs peut poser des problèmes d'accès non autorisés, l'intégration d'un serveur d'authentification et de contrôle d'accès permet de faciliter la protection des données et des programmes.



**Figure 3. . Le modèle client/serveur enrichi**

Parmi les avantages de ce modèle, on peut citer :

- **l'indépendance**: des processus clients par rapport aux processus serveurs
- **la transparence de localisation**: serveurs et clients sont indépendants géographiquement
- **la performance élevée**: les serveurs sont continuellement actifs
- **la sécurité des données**: le gestionnaire de transactions assure la cohérence des bases de données
- **la sécurité d'accès**: le serveur d'authentification et de contrôle d'accès permet de contrôler l'accès aux ressources partagées
- **la souplesse**: on peut ajuster le nombre de serveurs pour équilibrer l'application
- **l'efficacité**: les ressources partagées réduisent le coût par utilisateur
- **la flexibilité**: de nouvelles fonctionnalités peuvent être ajoutées sans perturber les parties existantes.

### 2.1.3 Le modèle DCE (Distributed Computing Environment) de l'OSF

Open Software Fondation est un consortium constitué des constructeurs informatiques tels que IBM, BULL, HP, DIGITAL et autres. Le but d'OSF est de construire un ensemble d'outils et de règles pour permettre l'interopérabilité entre les systèmes hétérogènes dans un environnement distribué (ROSENBERG 93). L'interopérabilité permet aux utilisateurs d'exploiter et de partager les possibilités et les ressources des différents systèmes connectés à travers le réseau et ainsi de fournir aux utilisateurs de meilleures performances et une utilisation plus effective des ressources informatiques. Un des choix majeurs d'OSF en matière d'architecture répartie est le DCE.

Le DCE est un ensemble de services étendus et intégrés qui supporte le développement, l'utilisation et la maintenance des applications distribuées. La possibilité d'avoir un ensemble uniforme de services sur des machines différentes, réparti dans le réseau, rend capable les applications d'exploiter réellement la puissance qui tend à rester inutilisée dans les systèmes informatiques. Cette architecture représente une opportunité pour transformer un groupe

d'ordinateurs interconnectés en une ressource informatique unique et cohérente. Il est constituée d'une couche logicielle qui masque les différences entre plusieurs types d'ordinateurs.

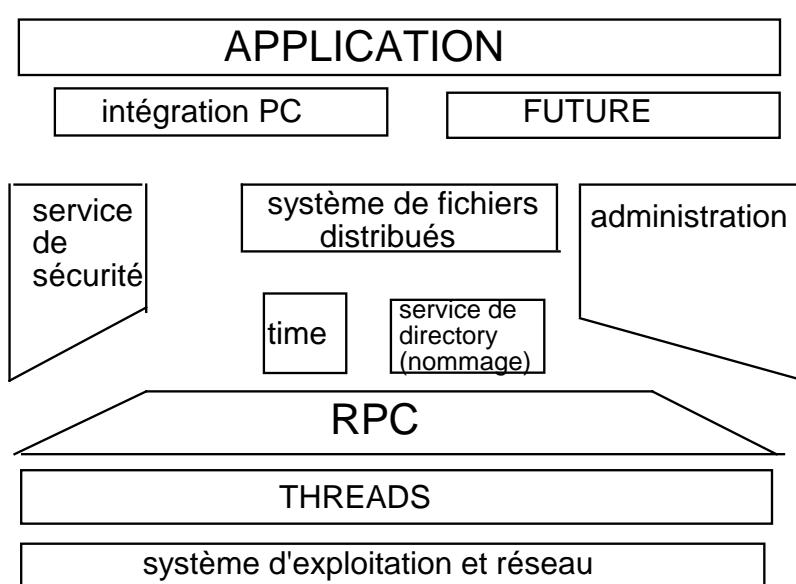
Ainsi le Distributed Computing Environment d'OSF permet aux utilisateurs et aux concepteurs de logiciels d'exploiter les ressources informatiques réparties sur un réseau. Cet environnement logiciel et intégré fait apparaître un réseau de systèmes de différents constructeurs comme un seul système. Dès lors, les utilisateurs peuvent, de manière transparente, bénéficier d'une multitude de ressources informatiques distribuées : applications, informations stockées dans des bases de données, service d'impression et d'archivage et la puissance de traitement. Il est bien évident que l'informatique distribuée soulève un certain nombre de problèmes spécifiques de mise en oeuvre. Comment protéger les données qui doivent être partagées entre plusieurs ordinateurs? Comment synchroniser des événements qui se produisent sur des ordinateurs distincts? Comment des systèmes utilisant des formats de données et des structures de fichiers différents vont-ils pouvoir coopérer ?

L'architecture DCE, en fournissant un ensemble d'outils et de services permet de répondre à ces différents problèmes. Cette architecture est basée sur un modèle en couche et intègre un ensemble de sept technologies (IBM 92) qui présente l'environnement comme une unité logique de système plutôt qu'une collection de services disparates.

Les services de DCE sont organisés en deux catégories :

- Les services distribués fondamentaux offrant des outils pour développer des applications distribuées comme service d'appel de procédures distantes, service de sécurité, service de thread, service de distribution de temps et le service de nommage.
- Les services de partages de données construits sur ces services fondamentaux. Il inclut le système de fichiers distribués et le support des PCs.

La plupart des constructeurs propose le produit DCE comme la base de leur architecture distribuée et l'intègre dans leur environnement comme DCM de BULL et SAA d'IBM.



**Figure 4. . Les composants de DCE**

En résumé, le DCE essaie d'apporter une solution aux besoins des entreprises qui portent essentiellement sur les points ci-dessous :

- protéger l'investissement informatique
- élargir le choix des solutions
- optimiser l'utilisation des ressources
- améliorer la productivité des utilisateurs

## Présentation des composants de DCE

### L'appel de procédures distantes (RPC pour Remote Procedure Call)

Ce mécanisme permet de distribuer une application sur plusieurs machines interconnectées. Il représente le traitement d'une requête d'un programme appelant pour utiliser une procédure placée dans un programme séparé dans l'une des machines.

### Le service d'annuaire (Directory Service)

Au coeur de DCE se trouve le service d'annuaire, le composant qui permet aux utilisateurs ainsi qu'aux applications de décrire et localiser les objets (personne, application, service, ...) participant à l'environnement distribué. Ce service de nommage ou localisation est très important, car il aide les usagers à gérer ou utiliser les informations disponibles dans le réseau.

### Le service de sécurité (Security Service)

Un environnement informatique distribué peut avoir à supporter des centaines d'utilisateurs accédant aux ressources situées sur n'importe quel ordinateur dans le réseau. Comme il s'avère difficile de maintenir les informations de sécurité de chaque utilisateur sur chaque machine de l'environnement, DCE rassemble ces données dans une base de données qui est logiquement centralisée, mais physiquement dupliquée sur les noeuds du réseau.

### Les threads

Le service des threads de DCE permet aux programmeurs d'exploiter la puissance de calcul et le parallélisme inhérent disponible à travers le réseau. Pour atteindre ce but, le DCE fournit des facilités pour le support de la programmation concurrente, permettant à une application d'exécuter plusieurs actions de façon simultanée.

Pendant qu'un thread exécute une procédure distante, un autre peut permettre la saisie des données de l'utilisateur. Le service de threads est utilisé par la plupart des composants de DCE tels que RPC, service de sécurité, service d'annuaire, service de temps et le système de gestion de fichiers distribués.

### Les services de synchronisation d'horloge (Time Service)

La synchronisation du temps est un problème majeur dans l'environnement distribué. Cette synchronisation permet à des applications distribuées de déterminer l'enchaînement, la durée et le calendrier prévisionnel des événements, quelque soit l'endroit où ils se produisent. Aussi longtemps que les horloges qui contrôlent les événements sont correctement synchronisées soit avec une base de temps, soit avec d'autres événements, les applications peuvent s'exécuter sans problème.

Le Distributed Time Service de DCE essaie d'apporter une cohérence au niveau du temps en synchronisant les horloges des ordinateurs interconnectés par des réseaux locaux ou à grande distance.

## **Le service des fichiers distribués (DFS)**

Le DFS permet d'avoir une vue globale sur les fichiers manipulés par les systèmes hétérogènes. En fournissant une interface cohérente, le DFS permet des accès globaux aux fichiers distribués sur les différents nœuds du réseau aussi facilement qu'en local. Pour réaliser cette tâche, le DFS d'OSF utilise le modèle commun client/serveur basé sur le RPC.

## **Le service d'intégration des PCs**

Le service d'intégration des PCs d'OSF donne aux utilisateurs de mini-ordinateurs, "mainframes" et PC la capacité de partage de fichiers, périphériques et applications dans un environnement distribué. En utilisant ce service, les utilisateurs des PCs sous DOS ou OS/2 peuvent accéder aux services offerts par DCE sur d'autres machines, visualiser, copier et transformer des fichiers vers ou à partir du monde UNIX. Ils peuvent partager des serveurs d'impression pour imprimer les fichiers stockés localement ou sur la machine serveur.

Le service d'intégration de PC utilise le service de NFS de SUN et le Server Message Bloc standardisé par X/OPEN et s'appuie sur TCP/IP ainsi que le protocole OSI pour la communication au niveau transport.

### **15.2.2 Le modèle Distributed Transaction Processing (DTP) de l'X/OPEN**

Historiquement, le système de gestion de TP (Transaction Processing) a démarré à partir des besoins spécifiques et pour des traitements particuliers des clients. Il était implanté uniquement sur des "mainframes" propriétaires. De cette manière, les applications développées pour l'utilisation d'un moniteur transactionnel ne pouvaient en aucun cas tourner sous un autre moniteur transactionnel.

Pour pallier cet inconvénient majeur, les clients et fournisseurs ont essayé de travailler ensemble pour assurer la pérennité des applications développées et réduire le coût de ces développements. Ces efforts ont été manifestés sous forme de standard "national" ou "international" par les organismes tel que ANSI (American National Standards Institute), ISO (International Standard Organization) et le groupe X/OPEN. Ces organisations ont essayé de casser le mur des systèmes transactionnels propriétaires (entre autres) en proposant des solutions standards sous forme de protocoles et API (Application Program Interface) pour accès aux services des moniteurs transactionnels. Un système de gestion des transactions est nécessaire pour gérer une large variété de ressources à travers des plates-formes différentes.

Pour répondre aux besoins des clients et pour permettre la pérennité de leur investissement, le groupe X/OPEN a proposé un modèle transactionnel distribué en définissant le protocole et les APIs correspondants.

#### **15.2.2.1 Le groupe X/OPEN**

X/OPEN est une organisation indépendante, mondiale et acceptée par bon nombre de fournisseurs de système, utilisateurs et les sociétés de logiciels. Sa mission est d'apporter une valeur ajoutée aux utilisateurs à travers des implémentations pratiques des systèmes ouverts. La stratégie de l'X/OPEN pour atteindre ce but est de combiner l'existant et fusionner les standards dans un environnement intégré (DTP 91), utilisable et compréhensif appelé "Common Application Environment".

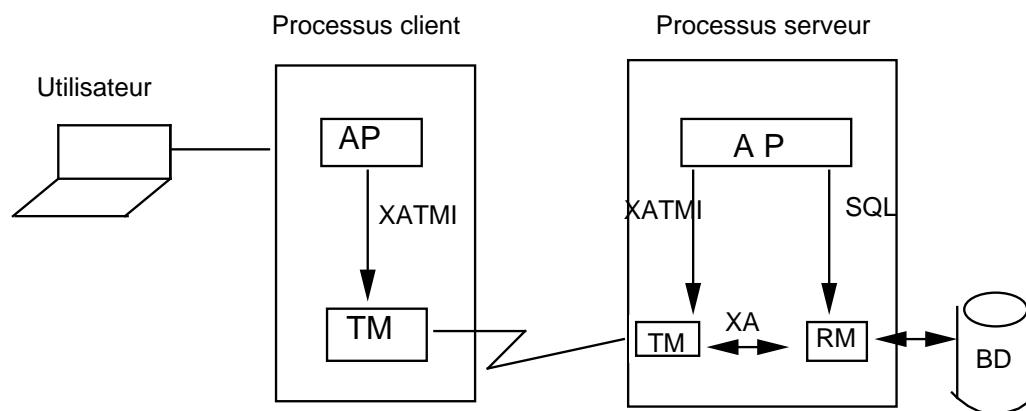
Les APIs définis par X/OPEN permettent d'améliorer de façon significative la portabilité des programmes d'applications au niveau de code source.

Une des plus importantes contributions de l'X/OPEN a été de spécifier un modèle pour le traitement transactionnel distribué (Distributed Transaction Processing). Ce modèle a l'avantage d'offrir une découpe organique et modulaire des différentes entités impliquées dans le traitement transactionnel et ainsi de proposer une implémentation simple dans un environnement local ou distribué selon le principe de modèle client/serveur.

### 15.2.2.2 Le modèle DTP

Le Distributed Transaction Processing (DTP 91) définit un modèle pour le traitement transactionnel dans un environnement distribué. Ce modèle prévoit trois composants logiciels :

- Un programme d'application (AP) définit les limites d'une transaction et spécifie les actions que constitue une transaction
- Un gestionnaire de ressources (Resource Manager comme un gestionnaire de base de données ou un système de gestion de fichiers ou toute autre ressource) fournit l'accès aux ressources partagées.
- Un gestionnaire de transactions (Transaction Manager) prend la responsabilité de routage, validation de transaction et recouvrement des données en cas de panne.



**Figure 5. . Le modèle DTP de l'X/OPEN**

La figure ci-dessus illustre une instance locale d'un système DTP où AP appelle un TM pour structurer une transaction. Il peut exister plusieurs systèmes DTP sur le même processeur. Les interfaces utilisées par les composants de DTP sont l'interface XATMI (Application Transaction Manager Interface) entre AP et TM et permettant à AP de dialoguer avec TM, l'interface bidirectionnel entre TM et RM permet de gérer les transactions réparties en se basant sur le protocole de validation à deux phases et enfin l'interface entre AP et RM est en général ISO - SQL. Il faut noter que le protocole de validation en deux phases d'X/OPEN s'appuie sur la définition des services et protocole de CCR (Commitment, Concurrency and Recovery) d'ISO dans le cadre d'OSITP (voir ci-dessous).

Ce modèle s'appuie sur l'architecture client/serveur enrichi et ses composants sont les processus client, serveur et le gestionnaire de transactions.

## **Le processus client**

C'est la partie de l'application qui s'interface avec l'utilisateur final (via un terminal, un lecteur de code à barres, ...). Comme celui-ci est le processus "frontal", il est responsable de :

- L'acquisition des données en entrée
- Le lancement de(s) requête(s) de service
- La réception de(s) réponse(s)
- La gestion de l'affichage des résultats

## **Le processus serveur**

C'est la partie de l'application qui exécute des services spécifiques. Souvent, elle interface un gestionnaire de ressources (RM), typiquement un SGBD mais peut aussi bien être un autre type de gestionnaire de ressources, un spooler d'imprimante par exemple. Selon le modèle DTP, ce processus comme le processus client est formé d'une partie applicative (AP) et d'un gestionnaire de transactions (TM) et souvent, il comprend un gestionnaire de ressources.

### **Le gestionnaire de transactions (TM)**

Il assure le routage de requêtes et la réception de réponses pour des services nommés et prend en charge la gestion de la communication TM à TM, quand elle se fait à travers un réseau. Il a pour principales tâches de :

- Router les transactions vers le processus serveur concerné où qu'il soit dans le réseau à partir de requêtes de service par nom, ce qui assure une indépendance totale du client par rapport au serveur.
- Assurer l'atomicité des transactions réparties (une transaction répartie correspond à plusieurs transactions locales au niveau de plusieurs serveurs et ce par le biais d'une interface/protocole de "pilotage" du gestionnaire de ressources, c'est à dire l'interface XA).
- Permettre à l'administrateur de contrôler et d'adapter l'application aux conditions d'exploitation.

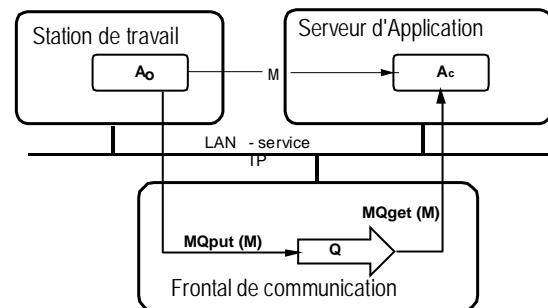
### **L'interface XA**

C'est l'interface entre le TM et RM dans le modèle DTP de l'X/OPEN. Dans le cadre des transactions réparties, cette interface garantit l'intégrité transactionnelle. Le protocole XA se base essentiellement sur le protocole de validation à deux phases pour réaliser cette tâche. Cette interface est complémentaire aux services de CCR (Commitement, Concurrency and Recovery) d'ISO puisqu'elle gère la relation entre le gestionnaire de transactions et le gestionnaire de ressources dans le cadre de ce protocole. Dans un système transactionnel basé sur le modèle DTP, le Moniteur transactionnel fournit une partie des fonctions de cette interface, pour permettre au gestionnaire de ressources de récupérer des informations concernant la transaction répartie. La deuxième partie des fonctions est fournie par le gestionnaire de ressources qui permet au TM de manipuler la transaction (initialisation, validation, annulation,...).

### 15.2.3 Echanges inter-applications par files de messages : modèle MQ

Le **modèle MQ, Message Queueing**, est le plus récent et sans doute le moins connu alors qu'il offre une grande souplesse et une facilité d'utilisation qui en font tout l'intérêt. Fonctionnant typiquement sur un mode asynchrone, il permet de désynchroniser les applications et n'exige même pas que les deux applications qui interagissent s'exécutent simultanément. Par ailleurs il peut être aussi bien implanté entre des applications s'exécutant sur un seul système ou réparties. Enfin il peut, en utilisant un système de double files, répondre aussi aux besoins d'échanges client/serveur de type Requête/Réponse.

Ce système de communication **repose sur des transactions réparties**; il est en cours de normalisation à l'OSI comme 7ème partie du standard OSI TP (OSI Transaction Processing) avec la référence ISO/IEC CD 10026-7.



#### 15.2.3.1 Principes

Le modèle Message/Queue MQ repose sur le **transfert de messages entre deux applications à travers des files d'attentes sécurisées**.

Une application origine  $A_o$  envoie un message  $M$  à une application collecteur  $A_c$  par l'intermédiaire d'une file d'attente  $Q$ . Ce message est déposé par une commande  $MQput$  par  $A_o$  et retiré de la file par une commande  $MQget$  de  $A_c$ .

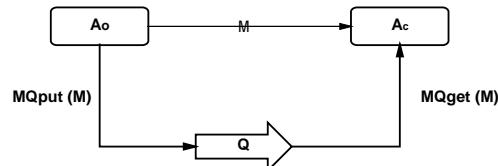


Figure 7

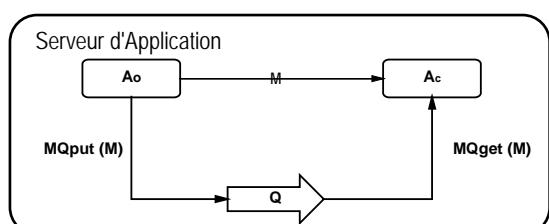
Les entités  $A_o$ ,  $A_c$  et  $Q$  peuvent être situées sur le même serveur d'applications ou être réparties entre un serveur d'application, une station de travail et éventuellement un frontal de communication qui ne supporte que les files. Ces architectures sont illustrées sur les schémas ci-dessous.

**La queue apparaît comme un service dans une architecture client-serveur.**

Les applications en sont des **clients source ou collecteur** des messages qui y sont déposés.

Le gestionnaire de file d'attente est un **serveur** qui rend, à distance, les services suivants:

- établissement du dialogue (avec le serveur)
- création et/ou ouverture des queues
- écriture ou lecture d'un message dans une queue
- fermeture et/ou destruction d'une queue
- terminaison du dialogue



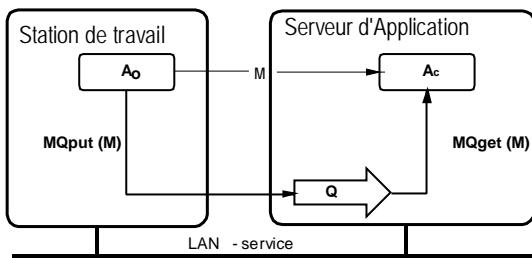


Figure 9

Les communications entre les applications clientes et le serveur de files d'attentes s'appuient sur un service (de communications) transactionnel qui assure l'intégrité et la durabilité des messages. Un **message** n'est considéré **écrit dans une queue** qu'après qu'une **validation à deux phases** (commit) ait été réalisée. De même, à la suite d'une lecture par un client, un message n'est effacé d'une queue qu'après qu'une validation à deux phases ait prouvé la fin du traitement.

Le serveur (gestionnaire) de files d'attente peut être physiquement installé soit sur un frontal de communications comme dans le schéma ci-dessus, soit réparti sur les systèmes qui supportent les applications comme illustré par le schéma ci-contre :

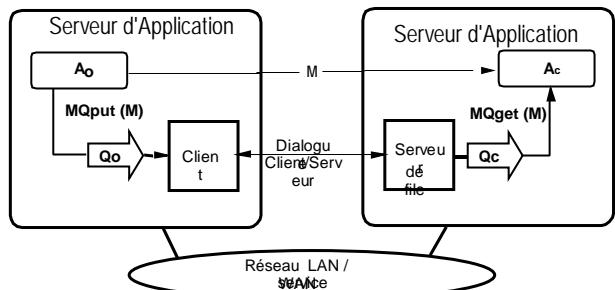
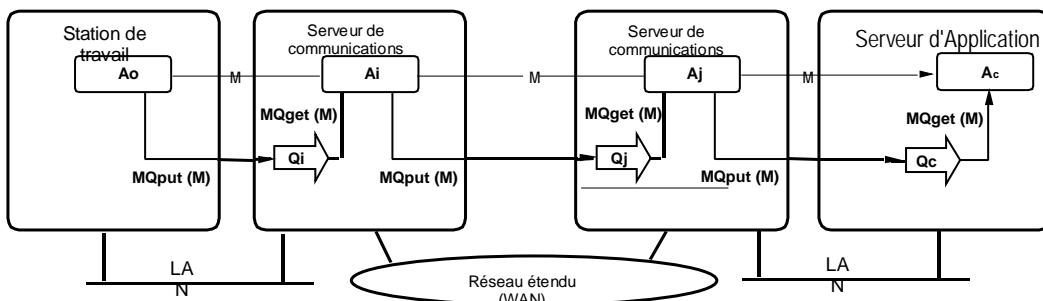


Figure 10

Le programme Client de l'application source lit un message dans la file Qo selon le modèle MQ et l'écrit dans la file Mc de l'application collecteur selon le même modèle. Dans chaque serveur d'application on peut aussi utiliser ce modèle pour écrire (resp. lire) les messages dans les files Qo et Qc.

On peut aussi ajouter des fonctions permettant de consulter sélectivement les files, de les purger sélectivement ou d'y modifier l'ordre des messages.

Enfin le serveur de files est identifié et chaque file qu'il contient est aussi identifiée par un nom logique et un type caractéristique des objets contenus.



Ce système de files réparties peut être généralisé en utilisant des systèmes de communications intermédiaires qui supportent ces queues comme illustré ci-dessous.

L'utilisation des systèmes relais peut être rendue complètement transparente grâce aux noms logiques et à des fonctions d'annuaires. Cette technique permet aussi d'utiliser des protocoles de communications différents entre les systèmes par exemple un ensemble ISO pour le réseau étendu et des ensembles Inet (TCP/IP) sur les réseaux locaux (ou toute autre architecture spécifique comme SNA). Des fonctions supplémentaires d'archivage, de

diffusion, de chiffrement, de compression peuvent être placées sur ces systèmes intermédiaires.

#### **15.2.3.2 Communications asynchrones et synchrones**

Le modèle MQ est spécialement bien adapté aux communications asynchrones; il ne demande pas aux applications de s'exécuter simultanément, les files permettant de stocker les messages temporairement de manière sûre.

Un système de communication symétrique avec des files permettant des échanges dans les deux sens permet de travailler en mode asynchrone avec réponse ou en mode synchrone (attente de la réponse pour continuer l'application cliente) si les messages sont correctement identifiés pour pouvoir mettre en correspondance une réponse dans la file de retour avec une requête de la file de commande. Ce service de réponse peut permettre de sécuriser les échanges sans écriture systématique des messages sur disque; dans ce cas une réponse de l'application serveur (positive ou erreur) permet de connaître le sort de la requête (une non-réponse dans un délai donné doit être alors considéré comme un incident qui demande une reprise ...). La charge système pourrait en être allégée. Toutefois cette technique n'est pas régulière vis à vis d'une application asynchrone et demande un système spécifique pour traiter les anomalies. L'utilisation d'un seul service et d'une seule interface constitue un avantage non négligeable.

### **15.3 La normalisation du traitement transaction réparti : OSITP**

Dans le cadre de l'interconnexion des systèmes ouverts, l'OSI a entrepris la normalisation d'un protocole de communication pour les applications transactionnelles distribuées. Dans le contexte de l'Interconnection des Systèmes Ouverts, l'objectif d'un tel effort a été de définir un modèle à partir duquel les services et le protocole pourraient être dérivés pour supporter les transactions distribuées à travers une collection de systèmes ouverts. Ce protocole doit permettre le traitement des données réparties de façon fiable et cohérente. Les travaux de normalisation ont débuté en Avril 1986 et enregistré sous forme de DIS (Draft International Specification) en novembre 1989 en définissant les services d'ISO attendus ainsi que les spécifications de protocole. Dans ce cadre, de 12 à 14 pays membres de l'OSI, CCITT et l'ECMA ont suivi les travaux, soit environ une cinquantaine d'experts. Au niveau français, l'AFNOR s'est montrée particulièrement active dès le début des travaux et a proposé le modèle, services et protocole dont de très larges parts sont retenues dans la version de DIS (BARBOT 91).

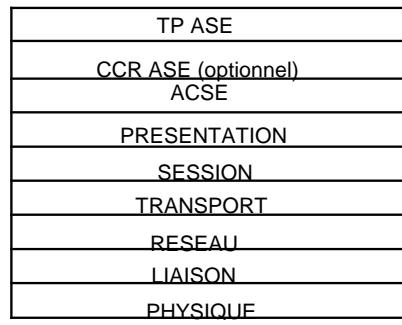
La norme définitive (IS) a été adoptée en avril 1991 et les documents détaillés ont été publiés.

#### **15.3.1 Présentation de la norme TP (Transaction Processing)**

Dans le contexte d'ISO, le traitement transactionnel est pris en charge par un ensemble de systèmes ouverts exécutant la transaction. Deux ou plusieurs systèmes coopèrent pour réaliser une tâche donnée appelée transaction. L'objectif premier de la norme est de spécifier un protocole qui par encapsulation de la communication entre processus d'application confère à cette communication des caractéristiques transactionnelles. Ces

caractéristiques résident principalement dans le traitement résistant aux pannes et le traitement réparti hiérarchisé. Aux côtés des buts strictement transactionnels, on retrouve les exigences de performances, conditions essentielles à la pratique des transaction réparties. Le cahier des charges souligne le besoin d'exécuter des séquences de transactions, avec un temps de réponse minimum et d'un traitement efficace de transactions courtes ou longues, et cela même dans une situation de trafic élevé.

Vis-à-vis du Modèle de Référence ISO, la norme TP se place dans la couche application (DRAFT-ISO 90), c'est à dire au niveau des entités d'application rendues visibles au monde des communications OSI.



**Figure 12.** . Le modèle architectural d'OSITP

La norme TP répond également aux principes d'intéropérabilité et d'économie développés par la normalisation OSI. Pour capitaliser sur l'investissement des normes OSI déjà développées, les services définis par la norme TP sont construits aussi souvent que possible, à l'aide des services offerts par la couche présentation, ACSE et CCR.

### 15.3.2 Terminologie et concepts de base

Le terme "transaction" utilisé correspond à une unité de travail caractérisée par les propriétés ACID. Dans le cadre de l'OSI, ce terme recouvre les notions de "données liées" et propriétés ACID. Les données liées (bound data) sont ainsi appelées car leur sort est lié à la destinée de la transaction. Une transaction OSI acquiert les données liées, les traite et enfin les libère. Du point de vue de la transaction, les données sont acquises dans l'état initial et sont libérées dans l'état final (leur valeur après traitement) ou l'initial, ce dernier cas est appliqué en cas d'échec.

La transaction OSI se définit au moyen de quatre propriétés, dites "ACID", vis à vis des données liées.

- La transaction doit être considérée comme une unité indécomposable de traitement ou encore atomique (le "A")
- La transaction OSI a pour objectif d'assurer que les données liées sont libérées dans le même état sur tous les sites de traitement, dans l'état final si la transaction se termine avec succès, sinon dans l'état initial ("C").
- Les données liées sont acquises par une transaction et une seule. Il ne peut y avoir ni partage de données entre transactions OSI, ni accès aux résultats intermédiaires d'une transaction OSI par une autre transaction OSI ("I").
- Les effets de la transaction OSI au niveau des données liées sont rendus durables vis-à-vis des situations de panne, bien que la transaction soit terminée ("D").

En résumé, la transaction OSI transforme les données liées d'un état cohérent à un autre état cohérent (GHERNAOUTI 90), même en cas de panne. C'est sur cette base que la norme OSITP a été établie. Ce protocole transactionnel vise à garantir les quatres propriétés ACID.

Pour garantir l'intégrité transactionnelle, OSITP se base sur le protocole de validation à deux phases assuré par les services de CCR (Commitment, Concurrency and Recovery).

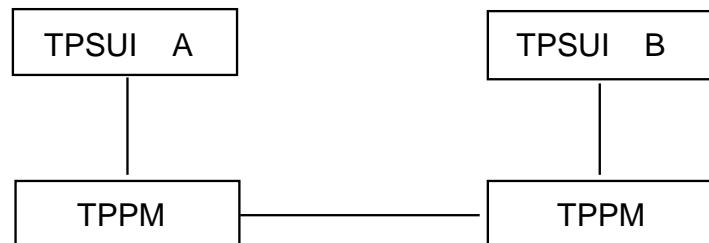
Pour atteindre cet objectif, l'OSITP définit un modèle de communication dans lequel les différentes entités sont organisées en arbre de dialogue et transaction répartie.

### 15.3.3 Dialogue

Lorsque deux systèmes ouverts veulent communiquer, ils le font en établissant une relation entre eux. En terme d'OSI, cette relation est établie à différents niveaux. Par exemple, une connexion réseau doit être établie pour permettre une connexion transport, etc . . . jusqu'à la couche application, dans laquelle selon la terminologie d'architecture OSI, on établit une association entre les processus d'application.

Pour permettre la relation entre processus de traitement des transactions, et réduire également l'"overhead" de traitement, OSITP en établissant une association, définit un niveau abstrait de relation entre systèmes ouverts appelé "dialogue".

Lorsqu'après l'établissement d'association entre deux systèmes, une invocation de processus transactionnel (TPSUI pour Transaction Processing Service User Invocation) veut demander un service particulier de transaction d'un autre processus (TPSU), si la TPSUI demandée est active, un dialogue va être établi entre ces deux TPSUIs.



**Figure 13.** . Le modèle de dialogue dans OSITP

En d'autre terme, le dialogue se définit comme la relation de communication entre deux TPSUIs appartenant à deux processus d'applications différentes. Pour la transmission des éléments sémantiques, le dialogue utilise une association qui peut être établie et gérée dans un "pool" commun.

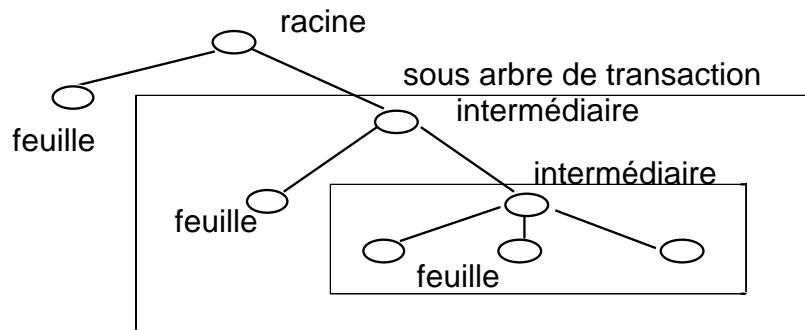
### 15.3.4 Arbre de transaction

Une TPSUI peut établir un ou plusieurs dialogues, chacun avec une TPSUI distante, ce qui peut signifier la création d'une nouvelle instance de TPSU demandé. A son tour, une TPSUI distante peut établir un ou plusieurs dialogues avec d'autres TPSUIs, etc...

Cette possibilité pour une TPSUI de créer une instance d'une TPSU donne naissance à une structure dont la nature est un arbre (arbre de dialogue).

OSITP fournit deux modes pour assurer le dialogue entre TPSUIs. Dans un mode, il fournit le support minimal à ces derniers (GHERNAOUTI 90) et leur permet la coordination entre systèmes ouverts pour garantir la bonne exécution de transaction distribuée. Le second mode est plus riche et fournit aux utilisateurs de services OSITP le support complet de

maintenance des propriétés ACID de la transaction répartie (le choix de ces modes s'effectue suivant l'attribut "coordination" d'un dialogue).



**Figure 14.** . L'arbre de transaction dans OSITP

Dans le cas où ces deux modes sont disponibles, une TPSUI peut établir une branche de transaction avec son partenaire dans le cadre d'un dialogue. Cette branche peut être ouverte à l'ouverture de dialogue ou plus tard. La fin de la branche de transaction se produit soit par la fin de la validation à deux phases soit par un retour arrière; c'est à dire la libération des données liées dans l'état initial. En d'autres termes, à l'intérieur d'un arbre de dialogue, on peut avoir un ou plusieurs sous arbres ayant demandé le support complet des propriétés ACID à OSITP. Comme ces sous-ensembles de l'arbre ont également une structure en arbre, ils sont appelés arbres de transaction.

Un arbre de transaction consiste en une racine, des noeuds intermédiaires et des feuilles comme un arbre de dialogue.

### 15.3.5 Services et protocole d'OSITP

La machine protocolaire OSITP utilise l'ASE (Application Service Element) de Association Control Service Element (ACSE) pour établir et libérer les associations d'application.

Lorsqu'une TPSUI demande l'établissement d'un dialogue, il spécifie si le TPPM devra fournir les services pour assurer les propriétés ACID.

Les services d'OSITP sont regroupés sous formes d'unités fonctionnelles (GHERNAOUTI 90). L'unité fonctionnelle noyau (Kernel) supporte le service de base de traitement transactionnel permettant d'ouvrir un dialogue entre 2 TPSUls. D'autres unités fonctionnelles sont définies pour assurer les modes de contrôle polarisés et partagés pour fournir les services de synchronisation, ceux de coordination et de chaînage ou de déchaînage.

#### 15.3.5.1 Noyau

L'unité fonctionnelle noyau supporte les services pour établir un dialogue entre deux TPSUls dans lesquels les U-ASEs peuvent être invoquées. Cette unité fonctionnelle comprend les services pour ouvrir un dialogue, le terminer, l'interrompre, en refuser l'ouverture et traiter les erreurs.

#### 15.3.5.2 Contrôle partagé

Cette unité fonctionnelle permet à deux TPSUls d'avoir le contrôle de dialogue, charge à ces derniers de s'entendre pour exercer leur contrôle de façon cohérente. Par exemple, les données peuvent être transférées simultanément par les deux invocations de TPSUs.

### **15.3.5.3 Contrôle polarisé**

Avec ce mode, seule une TPSUI possède le contrôle de dialogue à un instant donné. Les données sont transmises par la TPSUI qui possède le contrôle de dialogue.

A noter que les unités fonctionnelles de contrôle partagées et polarisées sont mutuellement exclusives. Pour un dialogue, une seule doit être choisie.

### **15.3.5.4 Synchronisation**

L'unité fonctionnelle "handshake" permet aux TPSUIs de synchroniser leur traitement. La synchronisation correspond à la possibilité pour les TPSUIs de vérifier qu'ils ont atteint un point de traitement convenu à l'avance. La sémantique d'une telle synchronisation est totalement définie par ces derniers.

### **15.3.5.5 Commit**

L'unité fonctionnelle commit (coordination) rend possible la validation (commitment) fiable ou le retour en arrière (rollback) des transactions. La coordination correspond à la possibilité de déclencher la phase terminale de la transaction avec validation à deux phases et reprise possible, sous contrôle du fournisseur de service TP. C'est cette possibilité de coordination qui permet de garantir les propriétés ACID. Si le dialogue n'est pas établi avec cet attribut, la garantie des propriétés ACID incombe à l'application. La validation à deux phases comprend la coordination de l'arbre de transaction, la journalisation, la gestion des données liées et le pilotage de CCR.

### **15.3.5.6 Le chaînage et le déchaînage des transactions**

Ces modes permettent le regroupement ou l'exclusion des TPSUIs de même branche dans une transaction.

unités fonctionnelles	primitives de service
noyau	TP-BEGIN-DIALOGE TP-P-REJECT TP-END-DIALOGUE TP-DATA TP-U-EROR TP-P-ERROR TP-U-ABORT TP-P-ABORT
contrôle partage	pas de primitive
contrôle polarisé	TP-GRANT-CONTROLE TP-REQUEST-CONTROLE
synchronisation	TP-HANDSHAKE TP-HANDSHAKE-AND-END TP-HANDSHAKE-AND-GRANT-CONTROL
COMMIT	TP-DEFERRED-END-DIALOGUE TP-DEFERRED-GRANT-CONTROL TP-COMMIT TP-CONTINUE-COMMIT TP-COMMIT-RESULT
	TP-DONE TP-COMMIT-COMPLETE TP-PREPARE TP-READY TP-ROLLBACK TP-ROLLBACK-COMPLETE
Transactions non chainées	TP-DEFER-NEXT-TRANSACTION TP-BEGIN-TRANSACTION

**Figure 15.** . Les unités fonctionnelles et primitives de services d'OSITP

## 15.4 Les moniteurs transactionnels

Le rôle d'un moniteur transactionnel est de permettre la communication et la mise en relation des processus clients et serveurs. Cette communication est complètement transparente pour ces processus quant à leur localisation et le média de communication (réseau) utilisé. En offrant une interface programmatique et des outils de supervision, il simplifie le rôle des développeurs d'application ainsi que celui de l'administrateur.

Comme il a été évoqué dans les pages précédentes, le moniteur transactionnel et le SGBD sont les deux composants indissociables d'un vrai système OLTP.

L'importance de ce composant dans les offres actuelles ou futures de tous les grands constructeurs (PIMONT 92) comme IMS/CICS d'IBM, TDS et TP8 chez Bull, ACMS de Digital,... montre à l'évidence le caractère stratégique du moniteur TP dans les architectures des systèmes distribués.

Dans le monde UNIX, et en voyant son importance comme système ouvert, le manque d'un moniteur TP a conduit les fournisseurs à proposer des produits pour répondre à ce besoin.

En effet, si pendant de nombreuses années le traitement transactionnel a été pratiquement le domaine réservé des grands systèmes propriétaires (IBM/MVS, BULL/GCOS, UNISYS,...), la situation est en train de changer pour différentes raisons, parmi lesquelles :

- le rapport prix/performance du système UNIX par rapport à celui des grands systèmes
- la montée en puissance des processeurs RISC et des systèmes UNIX construits autour de ceux-ci
- l'arrivée de nouvelles architectures de systèmes ouverts telles que les machines multiprocesseurs et à tolérance de panne
- la baisse des coûts des réseaux locaux et des stations de travail
- l'effet d'entraînement des standards et des systèmes ouverts
- l'amélioration de la fiabilité des systèmes UNIX
- la disponibilité de nombreux SGBD comme ORACLE, INGRES, INFORMIX, SYBASE,...
- les interfaces graphiques évoluées offertes par les stations de travail
- l'existence aujourd'hui d'une offre UNIX chez tous les constructeurs

Actuellement, on assiste à l'émergence sur le marché des moniteurs transactionnels sous UNIX des produits tels que:

- TUXEDO d'USL
- ENCINA de TRANSARC
- CICS/6000 d'IBM

Nous donnons dans les pages suivantes une description générale du produit TUXEDO de Unix System Laboratories ainsi qu'une présentation succincte des produits ENCINA et CICS/6000. Nous décrirons aussi un moniteur distribué utilisant le Modèle MQ.

## **15.4.1 TUXEDO**

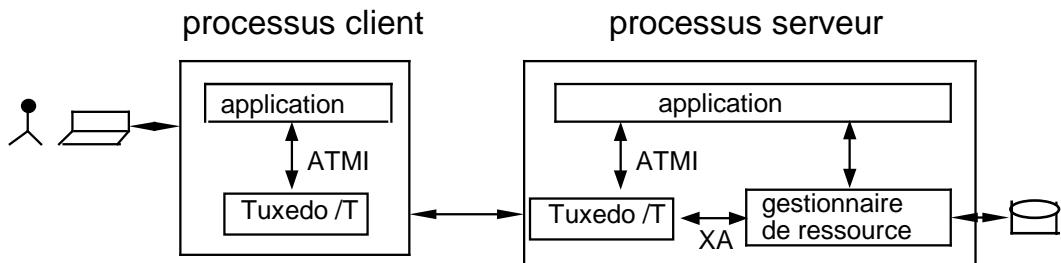
### **15.4.1.1 Présentation**

Ce produit a été développé et commercialisé par Unix System Laboratories d'ATT. TUXEDO a été conçu pour fournir un environnement transactionnel efficace sous UNIX avec une séparation nette des modules clients et serveurs sur les principes décrits précédemment. Ce produit a été porté sur de nombreuses plates-formes UNIX (USL 95) et également sous d'autres systèmes d'exploitation (VMS, OS2,...).

Il constitue la base de l'"offre transactionnelle" de plusieurs constructeurs comme BOS/TP de BULL ou OPEN OLTP d'UNISYS, mais est commercialisé également sur les plates-formes UNIX de DEC, HP, SUN, IBM,...

Il est sans aucun doute le moniteur le plus utilisé aujourd'hui. TUXEDO est conforme au modèle général DTP et correspond au composant TM tel que défini par l'X/OPEN. Il dispose de l'interface XA et peut donc communiquer avec les principaux SGBD du marché.

Son interface programmatique ATMI entre les applications et le moniteur a servi de base (USL 95) à l'X/OPEN pour définir XATMI (qui reste très proche d'ATMI). L'interface ATMI offre aux programmeurs d'application un accès transparent aux services offerts par TUXEDO, tels que les services de dialogues entre applications, conversion et manipulation des données et des opérations de contrôle d'une transaction. ATMI contient notamment les interfaces Tx, TxRPC et XATMI. L'interface Tx permet de délimiter la notion de transaction. L'interface TxRPC permet d'utiliser des appels de procédure distante (RPC) tel que défini par X/OPEN (compatible DCE) et enfin l'interface XATMI contient les primitives permettant de prendre en charge le dialogue client/serveur.



**Figure 16.** . L'architecture et les interfaces de TUXEDO

TUXEDO définit un intermédiaire logiciel entre les applications et les ressources en général. Il est capable de fédérer à l'aide d'interfaces standards (définis dans le modèle DTP de l'X/open) des plates-formes matérielles, des systèmes d'exploitation (DOS, UNIX, . . .), des différents SGBDs (Informix, Oracle, Sybase), le tout dans un ensemble cohérent et transparent.

Le modèle OLTP trouve dans TUXEDO le moniteur transactionnel qui assure le lien entre client et serveur; le modèle client/serveur est ainsi enrichi car TUXEDO introduit un serveur de nom (Name server) entre les deux, pour jouer le rôle de "dispatcher" entre n clients qui demandent m services de p serveurs. Ce produit est écrit en "C" ANSI et est conforme à SVID (System V Interface Definition), POSIX et XPG3 (guides de portabilité de l'X/OPEN).

Le moniteur TUXEDO comporte deux composants bien séparés :

- le SYSTEM /T transaction manager : ce module est basé sur l'architecture client/serveur enrichi et gère toutes les communications client et serveur
- le system /D DBMS : C'est le module de gestion de base de données à haute performance et conforme à l'interface XA. Il offre un ensemble d'outils pour construire et administrer des applications transactionnelles.

La figure précédente montre les processus clients connectés aux serveurs à travers le "Tuxedo /T" et le "Front End". Les avantages de cette interface sont :

- La transparence de la localisation du client par rapport au system /T
- La transparence du terminal : le type du terminal utilisé est sans importance pour le system /T. Il n'y a pas de restriction sur le type de client dans une application donnée
- La transparence de l'échec d'un client : si un client tombe en échec, cela n'affecte pas le reste de l'application
- L'entrée multiplexée : on peut combiner l'entrée de plusieurs clients en une seule entrée si le processus de Front End est programmé pour la manipuler.
- La figure montre également les processus serveurs connectés à Tuxedo /T à travers l'interface offerte par le "Transaction manager". Cette architecture offre les fonctionnalités suivantes :
- La localisation du processus serveur est transparent pour l'utilisateur ou le processus client
- Un serveur peut offrir plusieurs services et un même service peut être offert par plusieurs serveurs
- On peut attribuer des niveaux de priorités aux services. Les serveurs traitent en premier les requêtes avec les plus grandes priorités
- Un groupe de serveurs peut être activé (déplacé) sur une autre machine en cas d'échec d'un noeud et les applications continuent à tourner. Le recouvrement après Un échec est accompli sans perte de transactions

- Les serveurs d'application sont linkés avec les librairies de TUXEDO, ils sont donc accessibles pour le contrôle par le system /T
- On peut équilibrer les charges du système (Load balancing) en attribuant un facteur de charge à chaque service. Dans ce cas, le system /T choisit le serveur le moins chargé pour satisfaire la requête, ou bien attribuer une même queue à plusieurs serveurs et dans ce cas, la requête est prise en compte par le premier serveur disponible. En fonction de la configuration, ce choix peut améliorer la performance globale du système
- L'administration de TUXEDO est centralisée à travers un fichier de configuration. Ainsi, toutes les parties de l'application comme les ressources, machines, serveurs, services, etc. sont complètement définies dans ce fichier central
- Les ressources de l'application sont protégées contre les accès non autorisés. Les mécanismes de sécurité offerts par TUXEDO utilisent les services d'authentification de Kerberos et ainsi permettent de contrôler les permissions et les droits d'accès aux ressources de l'application.

#### 15.4.1.2 Les composants de TUXEDO

TUXEDO comprend différents composants parmi lesquels le moniteur transactionnel proprement dit appelé /T ainsi qu'un gestionnaire de données /D. Le système /T de TUXEDO offre une extension pour les PC (/WS) sous DOS ainsi que des extensions (/HOST, /CONNECT et /CONNECT REVERSE) pour les ordinateurs centraux (USL 95) en s'appuyant sur le protocole LU 6.2 (SNA). Les autres composants de Tuxedo sont /DOMAIN, /QUEUE, /RPC et /OSITP.

Le module NET/T de TUXEDO s'appuie sur l'interface socket pour TCP/IP et l'interface TLI (Xti) pour accéder aux fonctions de la couche transport du Modèle OSI pour l'utilisation des services du réseau.

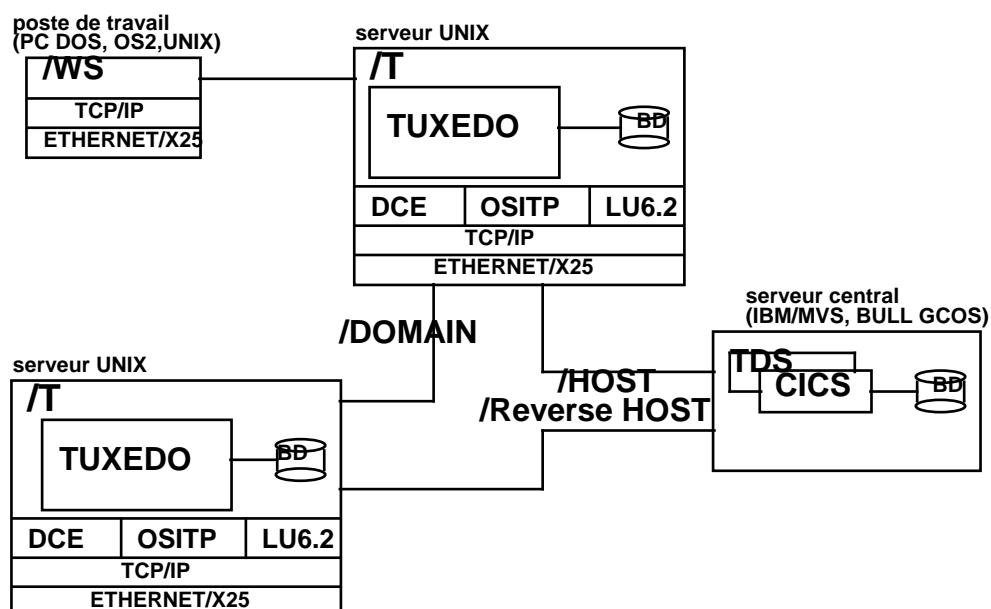


Figure 17. . Les différents composants de TUXEDO

#### - /WS

Cette extension de TUXEDO permet côté client à une application transactionnelle de s'exécuter entièrement ou partiellement à partir d'une station du travail. De cette manière, les modules clients peuvent être déplacés sur des machines intelligentes tournant sous UNIX, MS-DOS, OS2, Windows et Macintosh et réduire la surcharge du système. Cette facilité

permet également d'utiliser les possibilités des gestionnaires d'écran existant pour les stations de travail et les PCs.

Le TUXEDO /WS fournit les fonctionnalités nécessaires à l'établissement d'une connexion réseau entre le client et le serveur (interface TLI et socket) et réalise l'encodage/décodage des messages.

#### **- /HOST, /CONNECT et /Reverse HOST**

Ces modules étendent le modèle client/serveur dans l'environnement hôte et permettent aussi bien à un client TUXEDO d'accéder aux services et aux données d'un gestionnaire de transactions non TUXEDO en l'occurrence CICS d'IBM qu'à un module s'exécutant dans l'environnement mainframe d'appeler des services applicatifs de TUXEDO..

Ce logiciel fournit une passerelle qui permet la communication entre un noeud du SYSTEM /T et une application tournant sous CICS via le protocole de communication LU 6.2 de SNA en utilisant l'interface normalisée CPI-C.

#### **- /QUEUE**

Ce module permet à un client ou à un serveur TUXEDO de poster un message dans une file d'attente sécurisée (sauvegarde sur le disque) pour un traitement ultérieur (traitement asynchrone). Un serveur particulier retire les requêtes de service de ces files d'attente et effectue la demande de service.

#### **- /DOMAIN**

Ce module permet de décomposer une application TUXEDO en un ensemble d'applications à travers différents lieux géographiques et de les gérer sous forme de domaines de point de vue administratif.

#### **- /DCE**

S'appuie principalement sur TxRPC qui est une extension des RPCs de DCE en mode transactionnel. Dans la version 6, TUXEDO annonce l'intégration des autres services de DCE.

#### **- /OSITP**

TUXEDO dans sa dernière version permet l'utilisation du protocole OSITP par son extension /OSITP. De cette manière, TUXEDO se conforme aux spécifications d'ISO dans le cadre du système transactionnel normalisé.

### **15.4.2 ENCINA**

Le moniteur TP ENCINA de la société TRANSARC est un produit conçu spécifiquement pour la mise en place d'applications transactionnelles distribuées entre des systèmes ouverts.

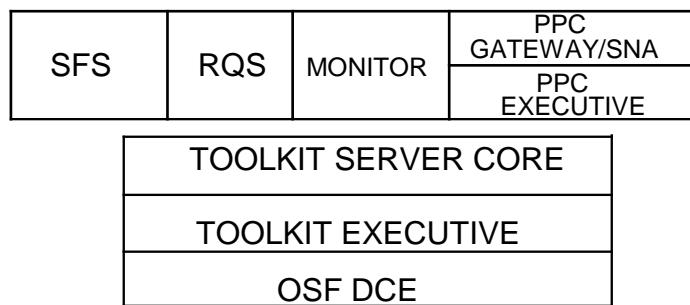
Il répond complètement aux concepts d'"open OLTP" et est conforme au modèle DTP de l'X/Open (ENCINA 92).

Transarc a été créé en 1989, par d'anciens chercheurs de l'université de Carnegie Mellon et des anciens ingénieurs d'IBM, et a annoncé son premier produit ENCINA en 1991.

L'importance du produit ENCINA, en plus de sa conformité au modèle DTP de l'X/open est le fait qu'il est construit sur la technologie DCE de l'OSF (Transarc est par ailleurs à l'origine du composant DFS de DCE) et de ce fait, lui confère un rôle important dans les années à venir au sein des systèmes ouverts. Les RPC de DCE et le service d'annuaire distribué CDS sont les deux briques de base, permettant d'assurer l'interopérabilité entre les composants d'ENCINA supportant une transaction distribuée entre plusieurs systèmes ouverts.

La caractéristique architecturale majeure du moniteur TP de Transarc est donc son intégration profonde avec DCE, dont il constitue une extension fonctionnelle.

Ce produit est modulaire et simple à mettre en oeuvre et a été choisi par différents constructeurs comme base de leur offre transactionnelle sous UNIX, en particulier IBM, HP et STRATUS.



**Figure 18.** . L'architecture générale d'ENCINA

#### 15.4.2.1 Les composants d'ENCINA

Encina est basé sur deux concepts stratégiques :

1- il étend le concept de DCE pour inclure les services de traitement transactionnel réparti, et la gestion de recouvrement des données.

2- il propose un produit conçu sur un modèle en couches fournissant dans chacune d'elles, des services utilisables par les couches supérieures à travers des APIs bien définies. L'architecture générale d'Encina comprend deux composants appelés respectivement Encina Toolkit Executive et Encina Toolkit Server Core.

**Encina TOOLKIT executive** étend les services de DCE d'OSF avec l'univers des technologies qui utilisent l'architecture client/serveur.

Il inclut transactional-C, une API qui fournit l'accès aux services TP et résout les problèmes liés aux accès concurrents, enrichit la technologie RPC de DCE pour y inclure la notion d'intégrité transactionnelle, et pour la gestion des données réparties sur plusieurs sites s'appuie sur le protocole de validation à deux phases. Ensemble DCE d'OSF et Encina, permettent à une station de travail d'initialiser une transaction, de localiser les services (à travers le service d'annuaire CDS de DCE et par le mécanisme de RPC) et valider les transactions réparties. Les programmes serveurs sont activés sous forme de "thread" DCE et accèdent à des bases de données externes via SQL.

**Encina Toolkit server core** étend les services de l'Executive pour supporter le stockage et la maintenance des données recouvrables. Il inclut une bibliothèque d'accès aux données, un système de journalisation basé sur des fichiers log et l'interface XA de l'X/OPEN, pour permettre l'interopérabilité des bases de données impliquées dans la transaction.

Transarc a développé sa propre famille de produits TP construite sur DCE de base et Encina Toolkit. Ces produits incluent :

- **Encina Monitor** représente le moniteur TP fournissant un environnement d'administration, de développement et d'intégration d'applications réparties TP.

Encina Monitor fournit également un environnement d'exécution fiable qui gère l'équilibrage des charges (load balancing) ainsi que des services d'ordonnancement (scheduling) à travers des ordinateurs hétérogènes dans le but d'assurer la haute performance et l'intégrité transactionnelle. Il utilise les services d'authentification de DCE pour gérer les problèmes de sécurité dans un environnement réparti et enfin fournit un ensemble d'outils pour la configuration et la gestion de tous les objets (client, serveur, machine, ...) d'un système réparti comme une unité cohésive.

- **Structured File Server (SFS)** est un système de gestion de fichiers orienté "record". Il fournit un accès rapide et sécurisé aux données et permet de respecter la cohérence de celles-ci en cas de problème. SFS est conçu spécifiquement pour le transactionnel sous UNIX et permet de pallier les faiblesses de ce système d'exploitation dans ce domaine en offrant le mécanisme de verrouillage des enregistrements dans le cas d'accès multiples par les serveurs par exemple. En outre, SFS se base sur les mécanismes d'indexation, clustérisation, Btree, pour accéder de façon performante aux données et fournit également les interfaces de type ISAM de l'X/OPEN et VSAM.

- **Peer - to - Peer communication service (PPC)**

Ce module offre la possibilité de communication programme à programme au-dessus de TCP/IP ou transport SNA par le biais de l'interface CPI-C de LU 6.2 (unité logique "conversationnelle" dans l'architecture SNA). De cette manière, un module client se trouvant sur une station de travail peut avoir un dialogue de type programme à programme avec un serveur tournant sous CICS sur une machine host d'IBM.

- **Recoverable Queuing Service (RQS)** fournit un mécanisme d'enqueuing/dequeueuing des données des transactions. Il permet de mettre en file d'attente les transactions pour être traitées ultérieurement.

#### 15.4.3 CICS/6000

CICS/6000 d'IBM comme TUXEDO d'USL est conçu pour réaliser et exploiter des applications transactionnelles à deux ou à trois niveaux (PIMONT 95) dans un environnement de systèmes ouverts interconnectés par un réseau hétérogène.

Ce produit offre un environnement relativement complet permettant le développement, l'exécution et l'administration d'applications réparties sur le modèle client/serveur.

Il est conforme aux principaux standards du marché et interopère avec les autres produits nécessaires pour mettre en oeuvre ce type d'architectures distribuées (SGBD, gestionnaires d'interface homme/machine,...).

CICS/6000 est un produit assez récent et basé sur des technologies modernes comme DCE. Ce produit a été développé à l'origine sur les systèmes AIX (RISC 6000) et a été porté par la suite sur des plates-formes HP-UX, ULTRIX et SNI (Simens-Nixdorf) et récemment sur SOLARIS de SUN (PIMONT 95).

Ce moniteur transactionnel respecte le modèle DTP de l'X/OPEN mais a choisi les APIs de la "famille CICS" et vise en priorité les clients d'IBM ayant déjà ce produit sur leurs systèmes centraux. De cette manière, ce produit permet d'enrichir la famille CICS en fonctionnant dans l'environnement RISC 6000 (existe déjà sur MVS, VSE , OS/2 et OS/400).

Toutefois, la spécification du produit montre qu'il s'appuie sur les briques de base du moniteur transactionnel ENCINA de TRANSARC (dont IBM est par ailleurs propriétaire). Ce choix s'explique largement par la stratégie "OSF/DCE" d'IBM qui a annoncé cette infrastructure sur toutes les plates-formes majeures (MVS, AIX, OS/400 et OS/2).

En proposant CICS/6000 dans l'environnement UNIX, IBM a pu commercialiser un moniteur transactionnel bénéficiant de l'ensemble des services de DCE (RPC, annuaire CDS, sécurité KERBEROS,...) et présentant les mêmes interfaces programmatiques que les autres produits de la famille CICS.

Par le fait que le CICS/6000 dispose des mêmes APIs que les autres moniteurs de la famille CICS (ce qui rend les applications portables d'une plate-forme MVS à une plate-forme UNIX) et qu'il interopère avec tous ceux-ci (en protocole SNA/LU6.2), IBM espère convaincre ses grands clients de l'intérêt de son propre moniteur par rapport à des produits comme TUXEDO ou ENCINA.

## 15.4.4 Moniteur MQM pour mise en œuvre du modèle MQ

Etude réalisée dans le cadre d'une convention de recherche "Marben" - INSA de Lyon - Ministère de l'Industrie.

### 15.4.4.1 Propriétés

Un moniteur MQM est un logiciel permettant la mise en œuvre du Modèle Message/Queue dans un environnement hétérogène. Il permet les échanges inter-applications en mode asynchrone à travers une interface programmatique (API) simple d'emploi. Cette interface peut, par exemple être l'interface MQI (Message Queue Interface) définie par IBM.

Un moniteur MQM permet de réaliser simplement des applications transactionnelles ou Client/Serveur et en facilite la portabilité. Il doit disposer d'une fonction d'annuaire ou avoir accès à un tel service pour assurer la transparence de localisation des applications (identifiées par des noms logiques).

Il doit être utilisable sur le système serveur qui le supporte ou accessible à distance grâce à des modules clients sur des stations de travail.

Un moniteur MQM doit pouvoir être utilisé par des applications situées sur le même systèmes ou sur des systèmes distants. Dans ce cas il utilise un protocole transactionnel OSI/TP pour sécuriser les échanges. Pour une implantation purement locale une sécurisation par un module local spécifique est utilisé pour sécuriser les messages dans les files.

Le moniteur doit assurer **l'intégrité complète et la durabilité de tous les messages** placés dans ses files d'attente. Pour cela ils doivent être écrits sur des mémoire non volatiles (disques en général) et cette écriture est contrôlée par un mécanisme de **journalisation** (logging) qui permet de connaître leur état et d'effectuer les reprises en cas d'incident sur les systèmes (crash machine par exemple).

### 15.4.4.2 Architecture

Un moniteur MQM doit comporter:

- Un gérant de files d'attentes
- Une fonction d'annuaire
- Un (ou des) modules Serveur
- Des modules clients
- Une fonction de journalisation/reprise

Ces composants logiciels sont indépendants et peuvent s'exécuter en parallèles. Ils seront réalisés par des processus ou de threads.

Le **module de gestion des files d'attente** constitue le noyau du moniteur. Ces files sont placées en mémoire partagées. Il permet de créer, d'ouvrir, de fermer, de supprimer ces files ainsi que d'y lire ou y écrire des messages. Il dispose pour cela de mécanismes de

verrouillages (sémaphores) permettant des accès concurrents à ces files de la part des modules serveurs, clients ou de journalisation.

La **fonction d'annuaire** permet de déterminer l'adresse à utiliser et le protocole nécessaire pour établir un dialogue avec le gérant de la file d'attente ainsi que le nom de la queue à lire ou à écrire dans celui-ci. Pour cela on utilise des identificateurs associés à chaque application et chaque file d'attente. L'annuaire peut aussi fournir des paramètres additionnels sur la qualité du service requis. On peut utiliser une base de données spécifique ou consulter un annuaire externe (X500 ou CDS dans l'architecture DCE de l'OSF).

Le moniteur MQM attend les requêtes de service (lecture ou écriture) des applications clientes. Pour cela il comporte un ou plusieurs modules **Serveurs** qui sont chargés de l'établissement des dialogues et de l'accueil de ces requêtes. Ces serveurs sont spécifiques d'un support de communications (OSI, Inet, SNA, ...).

Pour supporter les échanges directs entre des moniteurs, pour des applications en mode Requête/Réponse ou pour utiliser des systèmes intermédiaires, le moniteur MQM doit disposer de **modules Clients** chargés d'établir les communications avec les autres moniteurs et d'écrire ou lire dans leurs files.

Pour sécuriser les messages et en assurer l'intégrité, ils sont écrits, à la suite d'une requête MQput, dans un fichier journal et identifiés par un nom unique. Lorsqu'ils sont extraits de la file (Requête MQget) un enregistrement de suppression, comportant l'identificateur du message, est écrit dans un fichier journal secondaire. Une fonction de réorganisation, indépendante des fonctions précédentes, permet d'éliminer du fichier journal les messages obsolètes, en utilisant les informations du fichier journal secondaire. Ces fonctions sont réalisées par le composant de **journalisation (logging)**. A la suite d'un incident, après relance du moniteur, la lecture des fichiers journaux permet de reconstruire en mémoire partagée les différentes files d'attente et d'y réinscrire les messages qu'elles comportaient lors de l'incident; c'est la fonction de **reprise (recovery)** du composant de journalisation.

#### 15.4.4.3 Communications entre composants

##### Echanges au sein d'un même système

Les communications entre modules passent par les moyens d'échange offerts par le système d'exploitation : tubes, tubes nommés, streams, mémoire partagée, etc..

Chaque application est munie d'une interface MQI et d'un module client. Le moniteur dispose d'un module Serveur chargé de l'accueil des requêtes des clients (et des autres composants décrits ci-dessus)

Figure 19

L'application cliente (initiatrice) ouvre un dialogue avec l'un des serveurs du moniteur MCM en utilisant le module client le mieux adapté à ses besoins(tube, tube nommé, socket, stream, etc.) Puis elle **ouvre une file d'attente** par un appel de primitive MQ-OPEN et **écrit un message dans cette queue** par à une primitive MQ-ENQUEUE - Message (MQPUT avec l'API MQI) (1). Si la queue ne préexiste pas elle est créée par les serveurs (queue dynamique).

Pour effectuer ces opérations le serveur utilise des primitives internes (2) iMQOPEN ou iMQPUT vers le gérant de queues.

**Si la qualité "messages sécurisés" est demandée, le message est enregistré** dans le fichier journal principal LOG (3). Dans ce cas, l'activité MQ-ENQUEUE se termine (4) seulement après écriture du message dans le fichier journal.

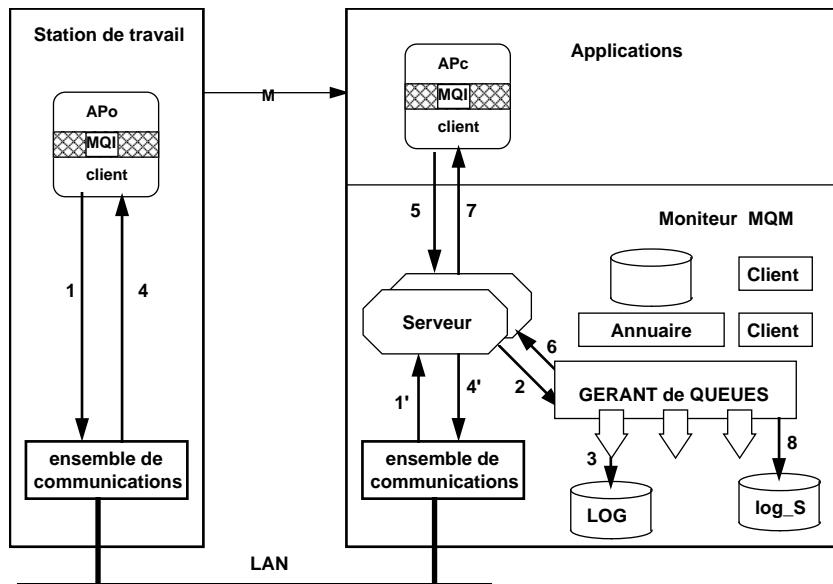
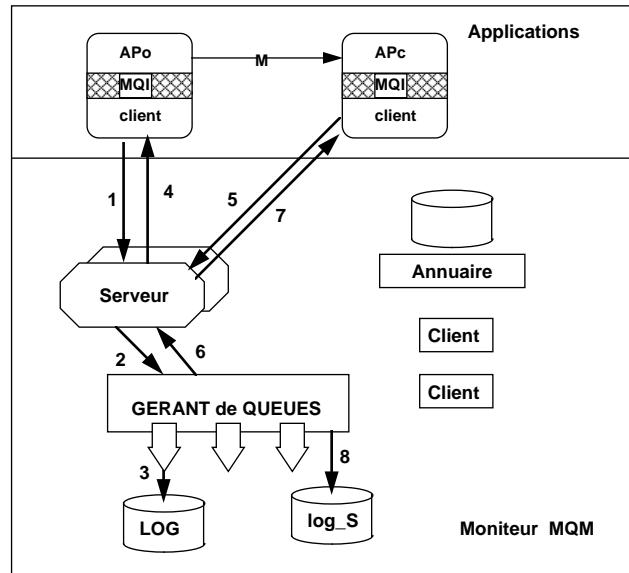


Figure 20

L'application serveur (répondeur) émet une primitive MQ-OPEN-DIALOGUE pour ouvrir un dialogue avec le serveur puis une primitive MQ-OPEN-QUEUE pour ouvrir la file d'attente à travers laquelle elle traite ses échanges. Elle demande la lecture d'un message par un appel de la primitive MQ-DEQUEUE-Message (5) (MQGET avec l'API MQI). Si un message est

stocké dans la queue il est lu grâce à une primitive interne iMQGET adressée au gérant de queues (6). Ce message est alors transmis au client (7) au moyen du dialogue établi auparavant. Si le message est sécurisé, une demande de suppression (8) est enregistrée dans le fichier journal secondaire log\_S.

Si il n'y a pas de message en attente, le comportement du moniteur MQM dépend de l'option choisie lors de l'appel de la primitive de lecture MQ-DEQUEUE-Message : synchrone (attente bloquante) ou asynchrone (attente non bloquante). Dans le premier cas le serveur ne fait rien d'autre que de noter la demande de lecture en attente. Dans le second il envoie à l'application une réponse signalant que la queue est vide.

## Echanges entre plusieurs systèmes

D'autres architectures sont susceptibles d'être utilisées. Le schéma ci-dessus montre le dépôt de l'application cliente initiatrice sur un terminal connecté par un réseau local.

Le schéma suivant montre l'installation du serveur de queues et de son moniteur MQM sur un frontal de communication. Les applications client et serveur sont reliées à ce frontal par un réseau. Les protocoles de communication vers ces systèmes peuvent être différents (par exemple OSI et TCP/IP)

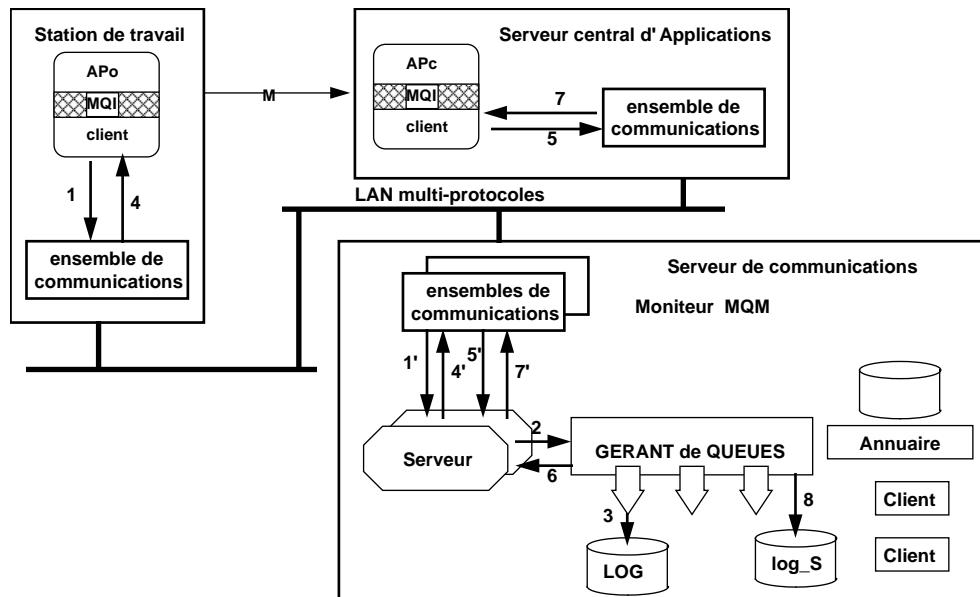


Figure 21

Le dernier schéma illustre une architecture plus générale. Elle met en jeu des systèmes qui peuvent être seulement client ou serveur, des systèmes intermédiaires qui ne sont que serveur de queue et des systèmes qui peuvent être serveur de queue et serveur d'application (en utilisant soit leur propre service de queue soit un service de queues distant).

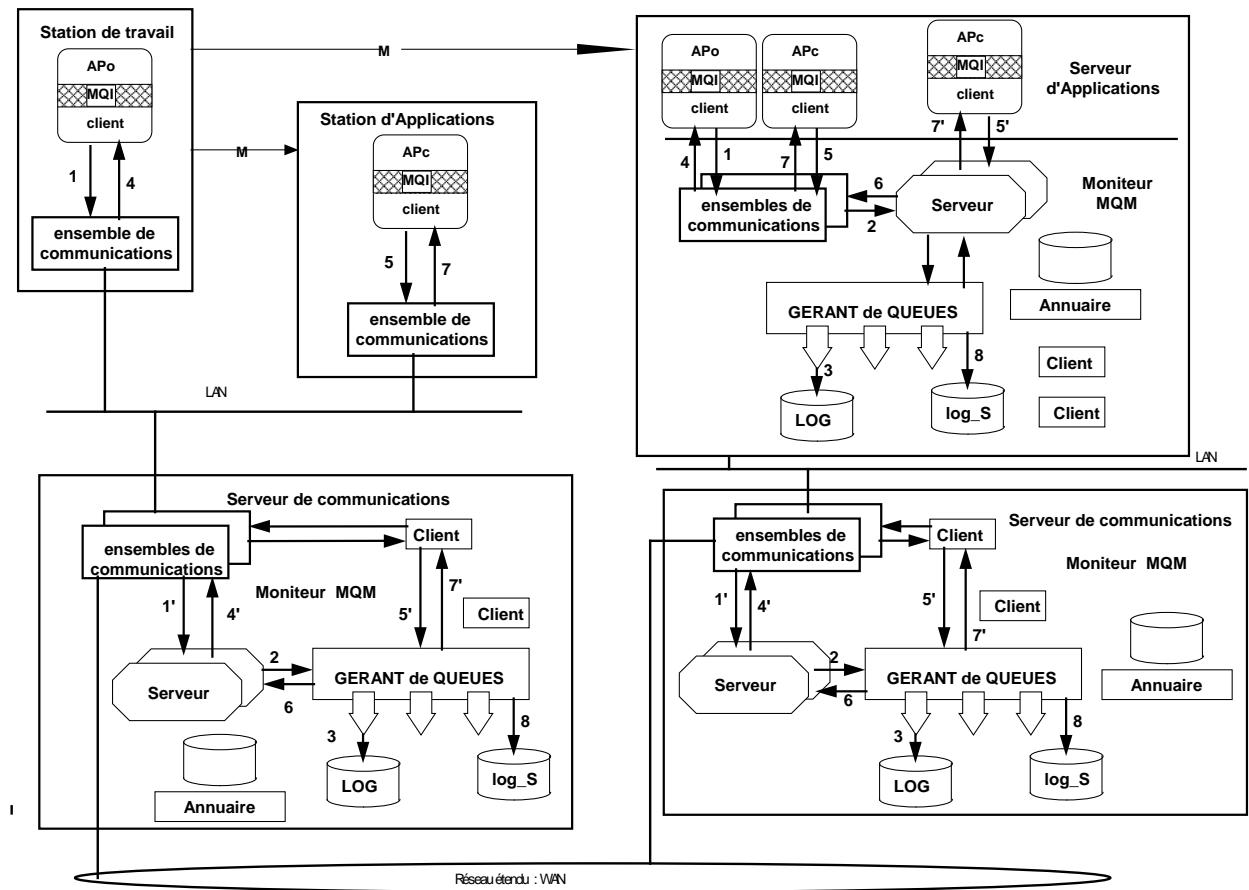


Figure 22

## 15.5 Conclusion

La décennie 90 a vu l'apparition et la généralisation du modèle client/serveur. Ce modèle adapté aux réseaux informatiques d'entreprise favorise la fédération des réseaux et des systèmes (avec leur applications) et l'accès par l'intermédiaire d'une station de travail ou d'un PC aux informations de différents systèmes.

Les modèles d'environnement distribué proposés par les grands constructeurs comme SAA d'IBM, NAS de DIGITAL ou DCM de BULL mettent en évidence la généralisation de l'architecture client/serveur dans un environnement ouvert et communicant basée sur les normes et standards. Ce modèle s'oriente vers une interface commune d'accès et de programmation et permettent aux utilisateurs de travailler dans un monde de continuité, de pérennité et d'efficacité.

La tendance actuelle des constructeurs et des utilisateurs par le biais des organismes de normalisation comme OSI ou X/OPEN est d'atteindre des objectifs de "transparence" du système d'information pour l'utilisateur. Cette "transparence" ne peut être réalisée qu'à travers un poste de travail donnant accès parallèlement à un grand nombre d'applications et de bases de données locales ou éloignées.

L'implémentation par les constructeurs des protocoles OSI (FTAM, X.400, OSITP,...) ainsi que les interfaces proposées par l'X/OPEN à un rythme soutenu ainsi que l'interopérabilité de ces environnements hétérogènes montrent la volonté de ces constructeurs d'ouverture vers le monde extérieur.

Dans ce contexte, du fait des contraintes administratives et géographiques, les données et les traitements sont répartis sur de nombreux systèmes de tous types (personnels, intermédiaires et centraux), dans des lieux souvent différents.

Le défi, par conséquent, est d'être capable de fournir des solutions applicatives (basées sur le modèle client/serveur et/ou coopératif) pouvant être mises en oeuvre dans un grand nombre d'environnements, tout en les rendant accessibles à tout le personnel de l'entreprise. L'évolution dont nous venons de retracer les grandes lignes, va profondément modifier les conditions d'utilisation des grands ordinateurs traditionnels ainsi que leur bases de données centralisées.

Ces ordinateurs vont devoir se comporter désormais comme de puissants serveurs, mis au service de projets dont ils ne constituent plus le cœur. De la même manière, avec la répartition des traitements et des données obligeant à éclater les informations sur les différents systèmes, les bases de données centralisées (DECISION 96) seront de moins en moins en mesure de répondre aux exigences transactionnelles des entreprises, et céderont donc la place aux SGBD distribués ou répartis.

## 15.6 Annexes :

### 15.6.1 Terminologie

ACID	Atomicité, Cohérence, Isolation et Durabilité, propriétés attribuées à une transaction
ACSE	Association Control Service Element, protocole défini par ISO pour l'établissement d'une association
ANSI	American National Standards Institute
AP	Application Program, un des "composants" d'un processus client ou serveur
API	Application Program Interface, interface de programmation
ASE	Application Service Element, éléments de service définis par ISO au niveau de la couche application
ATMI	Application Transaction Manager Interface, interface programmatique offerte par le moniteur transactionnel TUXEDO
CCR	Committement, Concurrency and Recovery, protocole de validation et reprise de transactions défini par l'ISO
DCE	Distributed Computing Environment, architecture d'environnement distribué définie par l'OSF
DDA	Distributed Data Access, accès aux bases de données distribuées
DFS	Distributed File Service, service de fichiers distribués
DIS	Draft International Specification
DTP	Distributed Transaction Processing, modèle de référence de l'organisme X/OPEN pour le traitement transactionnel réparti
EDI	Exchange Data Interchange, Echange de Données Informatiques
FTAM	File Transfer, Access and Management, la norme pour le transfert de fichiers définie par l'ISO
IPM	Inter Personal Message, messagerie interpersonnelle
IS	International Specification (norme)
ISO	International Standard Organization, organisme de normalisation
MHS	Message Handling System, système de messagerie
OLTP	On Line Transaction Processing, traitement transactionnel en ligne
OSF	Open Software Foundation, consortium créé par les constructeurs comme IBM, BULL, HP,...
OSITP	OSI Transaction Processing, norme pour le transactionnel réparti définie par ISO

POSIX	Interface système définie par IEEE
RM	Resources Manager, gestionnaire de ressources (gestionnaire d'une base de données par exemple)
RPC	Remote Procedure Call, procédure d'appel distant
SC	Serveur Central
SGBD	Système de Gestion de Base de Données
SL	Serveur Local
SNA	System Network Architecture, architecture de réseau IBM
SQL	Structured Query Language, le langage standard d'accès à une base de données
ST	Station de Travail
SVID	System V Interface Definition, l'interface définie par AT&T
TWO PHASE COMMIT	protocole de validation en deux phases
TM	Transaction Manager, gestionnaire de transactions
TP	Transaction Procession, traitement transactionnel
TPPM	Transaction Processing Protocol Machine, machine protocolaire d'un système transactionnel
TPS	Transaction Par Seconde, unité de mesure de la performance d'un système transactionnel
TWO PHASE LOOKING	verrouillage en deux temps
TPSU	Transaction Processing Service User, processus offrant un service transactionnel
TPSUI	Transaction Processing Service User Invocation, une invocation de processus transactionnel
XA	Interface/protocole définie par X/OPEN pour le dialogue entre le gestionnaire de transactions et le gestionnaire de ressources
XATMI	Application Transaction Manager Interface, interface programmatique proposée par X/OPEN dans le cadre d'un système transactionnel
X/OPEN	organisme de standardisation créée par les utilisateurs, les éditeurs de logiciels et les constructeurs
XPG3	guides de portabilité définis par l'X/OPEN

## 15.6.2 Bibliographie

- (UNISYS 89) Traitement transactionnel en ligne sur systèmes ouverts (UNISYS France 1989)
- (AFUU 89) Tribunix Bulletin de liaison de L'AFUU no 24 Fevrier 1989
- (GHERNAOUTI 90) S. Ghernaouti : Réseaux : applications réparties normalisées 1990 Eyrolles
- (DRAFT-ISO 90) Distributed Transaction Processing (DRAFT ISO /IEC 1990)
- (ATT 90) TUXEDO system transaction manager (document AT&T 1990)
- (DTP 91) Distributed Transaction Processing (X/OPEN 1991)
- (DESAINTQUENTIN 91) J.M DESAINTQUENTIN : L'informatique éclatée 1991 MASSON
- (OSF 91) Documentation technique de DCE (document OSF 1991)
- (BARBOT 91) The OSI - TP protocol : H. Barbot (BULL), G. Lacoste (IBM), S. sedillot (INRIA)
- (MARTIN 92) Daniel Martin : La performance transactionnelle 1992 Masson
- (CLAYBROOK 92) B. Claybrook : On Line Transaction Processing Systems 1992 Wiley
- (PIMONT 92) Systèmes distribués et traitement transactionnel J. L. Pimont, MARBEN 1992

- (IBM 92) The OSF DCE for The IBM MVS /ESA SYSTEM (document IBM 1992)
- (BULL 92) Distributed Computing Model (document Bull 1992)
- (ENCINA 92) ENCINA from TRANSARC (document Transarc corporation 1992)
- (GRAY 93) J. GRAY : Transaction Processing:concepts and techniques 1993 M.Kaufmann
- (ROSENBERG 93) W. Rosenberg, D. Kenney : Comprendre DCE 1993 Addison - vesleg
- (MARBEN 93) OSI Transaction Processing (document Marben 1993)
- (BERSON 94) A. BERSON : Client/Serveur Architecture 1994 McGraw-Hill
- (USL 95) TUXEDO system 5, Description technique d'ensemble document USL 1995
- (PIMONT 95) Le client/serveur avec les technologies Internet J. L. Pimont MARBEN 1995
- (DECISION 96) Decision Micro & Réseaux no 266 9 septembre 1996

### Articles sur Internet

- client-serveur goes business critical, Standish group int., webmaster@beasys.com
- On Line Transaction Processing, Simens Nixdorf & informations system, franz.dielmann@mch.sni.de, /public/mr/oltp2/oltphome.htm, 29 March 96
- Oracle powers world's largest OLTP client/serveur for Europcar, http://www.uk.oracle.com, /europe/emea/info/news/europcar.html, Jul 1996
- OLTP : Tuxedo conference 96 Overview, BEA systems Inc. http://www.beasys.com, /Oltp/tux96e3.htm, 23 May 96
- OLTP : Tuxedo conference 96, BEA systems Inc. http://www.beasys.com, /Oltp/tux96e2.htm, 21 June 96
- Transarc revises ENCINA OLTP monitor, Unix News Int. no 8, 5 octobre 1995, http://apt.usa.globalnews.com, /UNI/iss8/news2.htm
- On Line Transaction Processing, Data Management & Transaction Processing, http://www.mch.sni.de, /public/mr/oltp2/oltphome.htm, 10 June 96
- Encina Base Services, Introduction the Open Software Foundation (OSF) Distributed Computing Environment (DCE) provides a set of powerful, http://www.transarc.com, /afs/transarc.com /public/www/public/ProdServ/Product/Encina/base\_s, 6 June 96

∞

### 15.6.3 L'environnement distribué

#### 15.6.3.1 Le Modèle client/serveur généralisé

L'étude comparative des différents modèles proposés par les grands constructeurs (DCM chez BULL, NAS chez DEC, SAA d'IBM , NEWWAVE chez HP,...), pour faciliter la mise en oeuvre des systèmes distribués, fait apparaître un modèle commun appelé "**modèle client - serveur généralisé**" (PIMONT 92) et qui se caractérise de la manière suivante :

Au niveau le plus bas, le réseau d'entreprise permet d'assurer l'interconnectivité physique des différents systèmes (en mettant en oeuvre différentes technologies : 802.3, 802.5, FDDI pour les réseaux locaux ou X25, RNIS, Frame Relay,... pour les réseaux étendus).

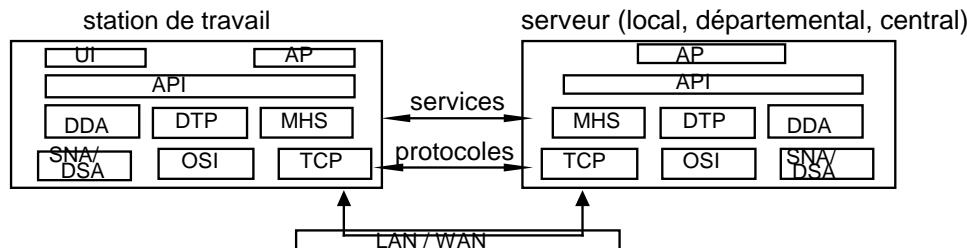
Chaque système, (c'est-à-dire une station de travail ou un serveur) comporte un service de communication supportant différents protocoles (principalement OSI, SNA ou TCP/IP) qui lui

permet de communiquer à travers le réseau d'entreprise avec la couche homologue d'un système distant.

Au dessus de cette "**couche protocole**", une "**couche service**" (couche au sens du terme informatique) permet d'offrir à travers des interfaces programmatiques standards (API) les différents services dont ont besoin les applications distribuées. Ces services, appelés globalement "services de distribution et d'intégration d'application" sont principalement :

- le partage de fichier dans un réseau local (DFS : Distributed File Service)
- l'accès transparent à des bases de données (DDA : Distributed Data Access)
- le traitement transactionnel coopératif (DTP : Distributed Transaction Processing)
- la messagerie et l'échange électronique de données (MHS/EDI:Message Handling Service/Electronic Data Interchange)
- les transferts de fichiers (FT : File Transfer)
- l'émulation des terminaux (TE : Terminal Emulation)

Ces services de distribution et d'intégration utilisent bien entendu les services fournis par la couche de protocole pour faire communiquer, à travers le réseau physique, une application répartie entre un client et un serveur et plus généralement entre un client et plusieurs serveurs.



**Figure 19.** . Le modèle client/serveur généralisé

### 15.6.3.2 Le modèle conceptuel du réseau informatique

La conception de l'architecture générale rend nécessaire la définition d'un modèle simple permettant de schématiser l'ensemble du réseau informatique et de faire apparaître d'une part les principaux composants, d'autre part les relations existant entre ceux-ci. Le réseau d'entreprise permet les types d'échanges suivants :

- une station de travail (ST) d'un réseau local peut communiquer avec :
  - le serveur local (SL de son réseau d'établissement)
  - le serveur local (SL d'un autre établissement)
  - le serveur départemental
  - le serveur central (SC)

Ces échanges peuvent s'effectuer selon les besoins des applications en mode :

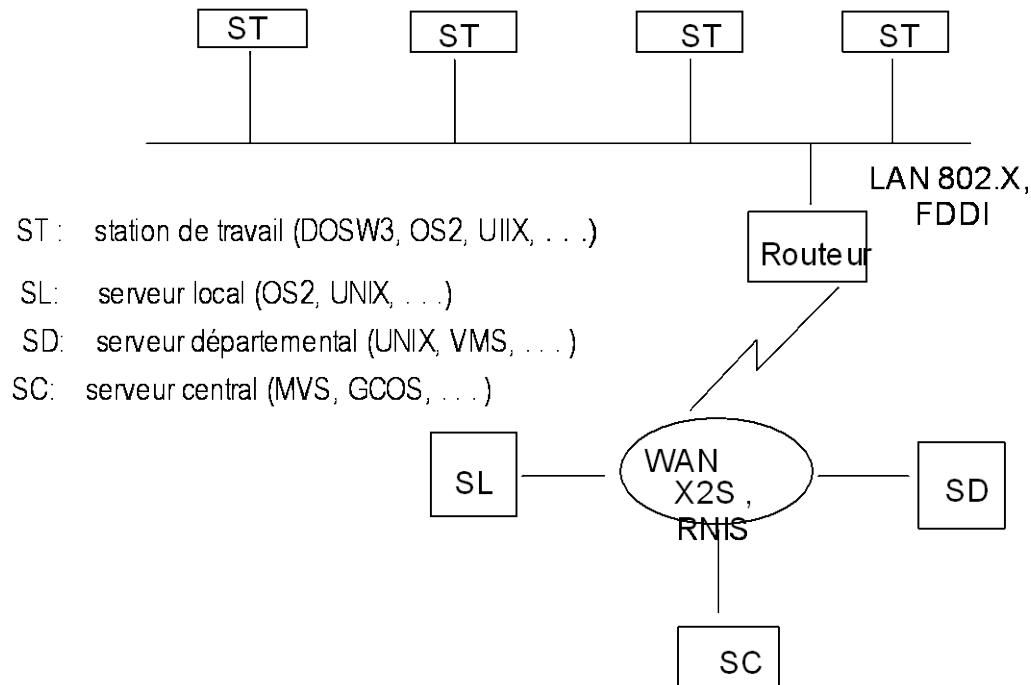
- émulation de terminal
- transfert de fichiers
- communication de programme à programme (DDA, DTP)
- messagerie (MHS) de personne à personne (IPM) ou entre applications(EDI)

Ainsi, une station de travail (ST) peut être utilisée :

- en émulation de terminal 3270 pour accéder à une application CICS d'un serveur central écrit pour ce type de terminal
- comme poste de travail pour accéder à une application transactionnelle distribuée (DTP) ou à une base de données distantes (DDA) ou encore à la messagerie interpersonnelle (MHS IPM).

Un serveur peut communiquer avec un autre serveur en utilisant des services de :

- transfert de fichiers
- communication d'applications à applications en mode
- transactionnel coopératif synchrone (DTP) ou
- messagerie (transactionnelle asynchrone)



**Figure 20.** . Le modèle architectural à trois niveaux

En fonction du modèle cité ci-dessus, aux architectures transactionnelles centralisées héritées des années 70/80, se substituent déjà depuis quelques années des architectures TP "client/serveur" à 2 ou à 3 niveaux (PIMONT 95) mettant en oeuvre :

- soit des postes de travail et des serveurs centraux UNIX ou propriétaires (TP à 2 niveaux),
- soit des postes de travail, des serveurs locaux/ et/ou départementaux et des systèmes centraux (TP à 3 niveaux).

Dans ces nouvelles architectures transactionnelles, les fonctions applicatives sont réparties entre les plates-formes:

- les postes de travail intelligents assurent l'interface homme/machine et une partie de traitement,
- les serveurs applicatifs effectuent le reste des traitements et l'accès aux bases de données.

Les postes de travail sont le plus souvent des "PC" (Windows ou OS2) connectés par un réseau local (Ethernet ou Token Ring) à un serveur d'établissement (NT, OS2 ou UNIX) et ce serveur local est lui-même relié à un système central UNIX ou propriétaire (IBM/MVS, BULL/GCOS,...) par un réseau généralement longue distance.