Institutional Repository - Research Portal
Dépôt Institutionnel - Portail de la Recherche
researchportal.unamur.be
University of Namur

THESIS / THÈSE

MASTER EN SCIENCES INFORMATIQUES

Étude des couches supérieures du modèle OSI et d'une implémentation particulière : le logiciel ISODE 4.0.

Dave, M.

Award date:
1990

Awarding institution:
Universite de Namur

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Download date: 23. May. 2024

Etude des couches supérieures du modèle OSI et d'une implémentation particulière : le logiciel ISODE 4.0.

Mémoire présenté par M. DAVE

Namur, septembre 1990

Je tiens ici à remercier Monsieur VAN BASTELAAR (FNDP), pour sa guidance et sa pertinence sans lesquelles ce mémoire n'aurait jamais pu naître. Mes pensées vont également vers Messieurs DETREMBLEUR (FNDP) et GEURTS (BIM) pour leur disponibilité, leurs conseils et pour les informations qu'ils m'ont aidé à trouver.

TABLE DES MATIERES.

1. INTRODUCTION.

Les problèmes traités par l'informatique varient fortement en complexité. A côté de cas simples comme trouver les racines d'un polynôme de degré 2, nous pouvons en trouver d'autres relativement compliqués comme la gestion d'une usine entièrement automatisée. Face à une telle situation, une manière pour l'informaticien de concevoir une solution gérable en termes de clarté, flexibilité, maintenabilité consiste à diviser le problème compliqué en sous-problèmes plus simples : des modules.

A un niveau relativement bas, cela revient, par exemple, à diviser un programme en plusieurs blocs ou en plusieurs procédures. A un niveau plus élevé, le problème sera divisé en plusieurs programmes s'exécutant sur des sites parfois différents. Nous appellerons ce découpage "Modularisation" et "Tâche" l'unité fonctionnelle représentée par un module.

Cependant, cette décomposition amène une nouvelle difficulté : la coopération nécessaire entre ces tâches. Elles doivent en effet communiquer entre elles pour réaliser la fonctionnalité globale de l'application qu'elles composent.

A un bas niveau, les échanges se font via des variables, des tableaux, des paramètres de procédures. A un niveau supérieur, les programmes sur des sites différents s'échangent parfois des fichiers relativement complexes et volumineux.

De plus, pour communiquer valablement, il faut être d'accord sur l'interprétation des données échangées.

A un bas niveau, cela revient à s'entendre sur le type des variables, l'ordre des arguments d'une procédure. A un haut niveau, cela peut impliquer des accords sur la structure des fichiers échangés.

Enfin, dans le cas d'application distribuée où un problème est divisé en plusieurs tâches s'exécutant conjointement sur des sites différents, il faut également coordonner leur dialogue. Gérer cette dernière situation est très complexe : il faut s'entendre sur beaucoup de choses et ces accords constituent de volumineuses et complètes séries de règles appelées "protocoles".

La téléinformatique étudie les solutions à apporter à ce problème de tâches éloignées devant communiquer. Le but de ce mémoire comprend 2 points de vue. D'une part, nous nous attacherons à décrire la solution théorique proposée par l'ISO pour résoudre les problèmes posés par la téléinformatique. Nous verrons que cette solution passe par une modularisation hiérarchique ou encore en une découpe en couches fonctionnelles. Nous nous intéresserons plutôt aux couches supérieures de cette découpe (mais pas les couches applications spécifiques). D'autre part, à coté de cette vue toute théorique, nous décrirons une application pratique du modèle, à savoir le logiciel ISODE dans sa version 4.0.

Plus précisément, le chapitre 2 abordera de façon très générale les éléments théoriques relatifs à la téléinformatique. Ensuite (chapitre 3), nous présenterons ISODE dans les grandes lignes. Les chapitres suivants (4, 5, 6, 7, 8, 9) exploreront les couches supérieures du modèle proposé par ISO et leur implémentation dans ISODE. Nous verrons au chapitre 10 quelques exemples. Le chapitre 11 abordera les bases de données utilisées par ISODE, tandis que TCP/IP sera évoqué au chapitre 12. Enfin, le chapitre 13 sera consacré à quelques considérations pratiques relatives à l'installation d'ISODE.

2. LES PRINCIPES DU MODELE OSI.

2.1. LES GRANDES LIGNES.

Le transfert de données est un problème informatique compliqué. Pour le résoudre, ISO propose de le simplifier par une modularisation hiérarchique. La solution repose sur une découpe à 7 niveaux, chacun d'entre eux se voyant attribuer une fonctionnalité qui sera utilisée par les niveaux supérieurs. Notons au passage que certains systèmes (noeuds de réseau) ne comportent pas toutes les couches (voir figure 2.1).

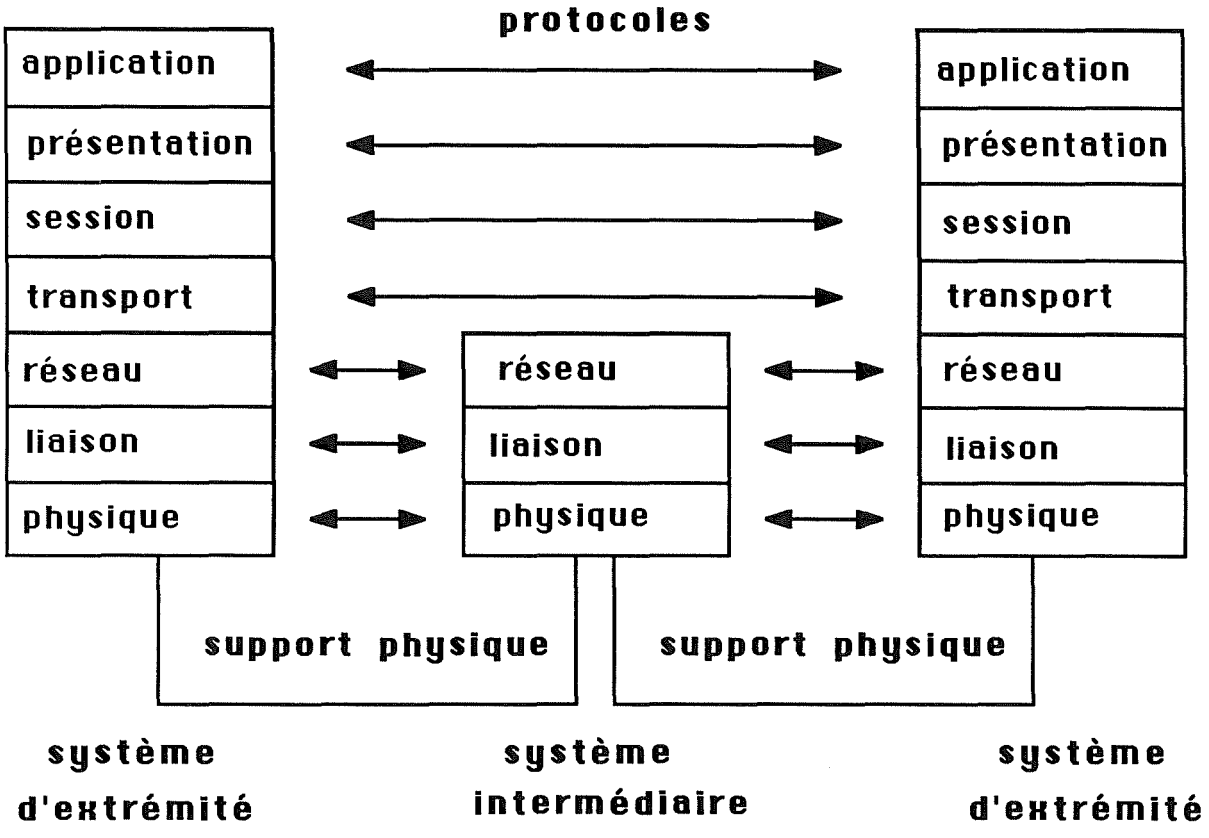

figure 2.1. le modèle OSI.

Voyons à présent les différentes couches et leur fonction.

Niveau 1 ou couche physique : elle a en charge l'usage correct de la ligne de transmission. Exemple : X21 est un protocole de couche physique.

Niveau 2 ou couche liaison : elle a pour rôle de transmettre des bits sans erreur, sans duplication ni perte entre systèmes adjacents. Exemple : le protocole HDLC.

Niveau 3 ou couche réseau : la couche réseau se charge des problèmes d'adressage et de routage. Exemple : X25.

Niveau 4 ou couche transport : le rôle de la couche transport est d'assurer le transport de bout en bout de l'information. Notons que cette couche se situe comme intermédiaire entre les couches orientées transmission (1,2,3) et entre celles orientées traitement (5,6,7).

Niveau 5 ou couche session : elle est responsable de la gestion du dialogue entre 2 applications en vue d'une coopération harmonieuse.

Niveau 6 ou couche présentation : cette couche gère les problèmes de compréhension entre les machines de même que ceux de compactage et d'encryptage des données.

Niveau 7 ou couche application : cette dernière couche offre aux utilisateurs les services de communication.

2.2. QUELQUES CONCEPTS IMPORTANTS.

2.2.1. SYSTEME OUVERT.

Un système est dit **ouvert** lorsqu'il permet la communication entre logiciels et équipements de types différents pourvu que ces derniers assurent leurs fonctions en conformité avec les conventions internationales. Par opposition, un système **fermé** ne permet de mettre en relation que des équipements en provenance d'un même constructeur, selon des protocoles qui lui sont propres et qui peuvent évoluer dans le temps sous son seul contrôle.

2.2.2. ISO-OSI.

L'ISO (International Standards Organization) est un organisme international qui a entrepris de définir l'architecture stratifiée à 7 niveaux pour les systèmes ouverts. Ce modèle de normalisation est appelé modèle de référence de base pour l'interconnexion de systèmes ouverts plus connu sous le sigle **OSI** (Open System Interconnection Reference Model). On peut donc parler à son sujet du modèle OSI d'ISO. Nous l'appellerons simplement le modèle OSI.

2.2.3. ENTITE.

On appelle **entité** E(N) un élément actif (programme ou ensemble matériel) qui réalise dans un système ouvert toutes les fonctions de la couche N ou une partie d'entre elles.

2.2.4. PROTOCOLE.

Une entité E(N) d'un système peut communiquer logiquement avec une entité homologue d'un autre système à l'aide d'un **protocole** P(N). Celui-ci définit les règles et formats caractérisant les communications entre entités homologues.

2.2.5. SERVICES ET PRIMITIVES DE SERVICES.

Une couche de niveau N offre un ensemble de **services** à la couche N+1. Chacun d'entre eux est représenté par un **élément de service**. Chaque élément de service est accessible par l'utilisation explicite de **primitives de services**. Une entité de niveau N+1 utilisant les services d'une entité de niveau N est appelée **utilisateur**, l'entité inférieure étant appelée **fournisseur**.

2.2.6. PDU ET SDU.

Au niveau logique, 2 entités homologues de niveau N dialoguent selon un protocole P(N) en s'échangeant des informations réunies au sein de **Protocol Data Units (PDU).** Concrètement, une entité E(N) se base sur le service offert par une entité E(N-1) pour transmettre ses (N)PDU. Elle les expédie concrètement à son entité E(N-1) au sein de **Service Data Units (SDU).**

Ainsi, l'information est transférée concrètement entre couches adjacentes (communication verticale) d'un système à l'autre, alors que les entités E(N) communiquent logiquement entre elles (horizontalement) en ignorant les détails de ces transferts.

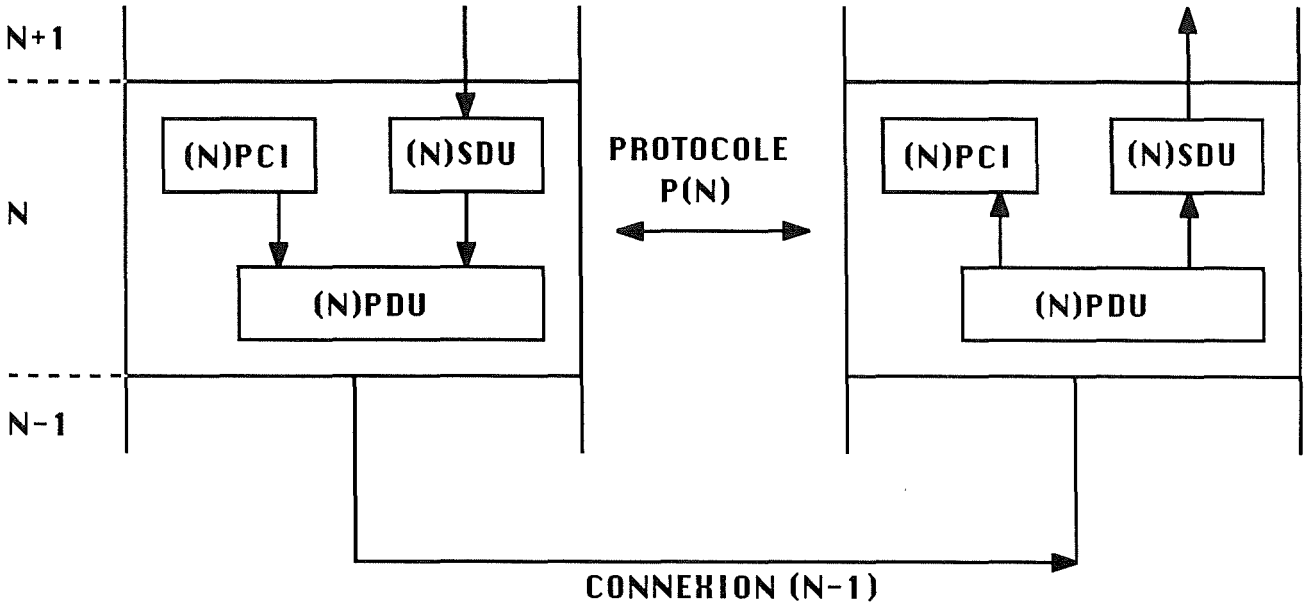

figure 2.2. Liens entre SDU et PDU.

La figure 2.2. montre comment est organisé l'échange de données entre entités. L'entité émettrice E(N) reçoit de la couche N+1 une unité de données de service (N)SDU. Elle y ajoute des informations de commande de protocole (N)PCI pour coordonner l'échange avec l'entité homologue, l'ensemble constituant l'unité de données de protocole (N)PDU échangée entre les entités E(N) par le protocole P(N). L'entité E(N) réceptrice extrait de la (N)PDU les informations de commande de protocole, les interprête, en effectue les actions correspondantes et transmet à la couche N+1 le (N)SDU.

2.3. MECANISME D'UNE DEMANDE DE SERVICE.

2.3.1. TRANSFERT NORMAL AVEC ACCUSE DE RECEPTION.

Le service avec accusé de réception appelé aussi service confirmé à 4 temps est le plus complet. La demande de service donne lieu à un signal de requête émis par l'utilisateur local, un signal d'indication émis par le fournisseur éloigné, un signal de réponse généré par l'utilisateur éloigné et un signal de confirmation produit par le fournisseur local (figure 2.3).

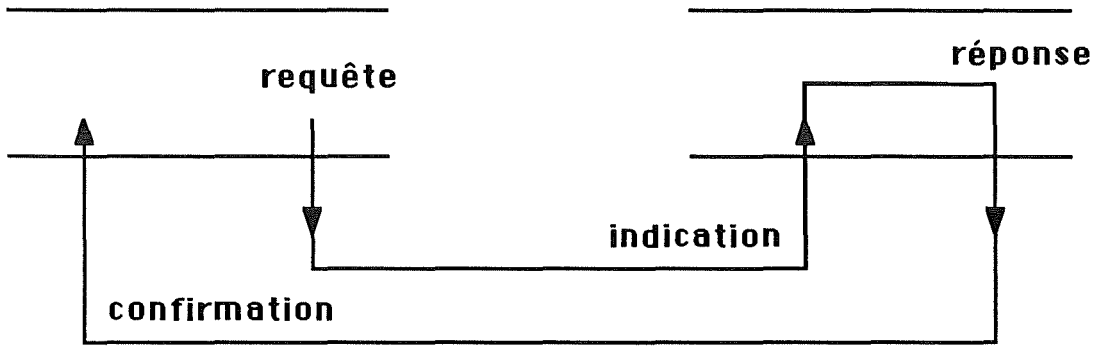

figure 2.3. service à 4 temps.

2.3.2. TRANSFERT NON CONFIRME.

Le service sans accusé de réception est aussi appelé service non confirmé à 2 temps. Il ne met en jeu qu'une requête et une indication (figure 2.4.).

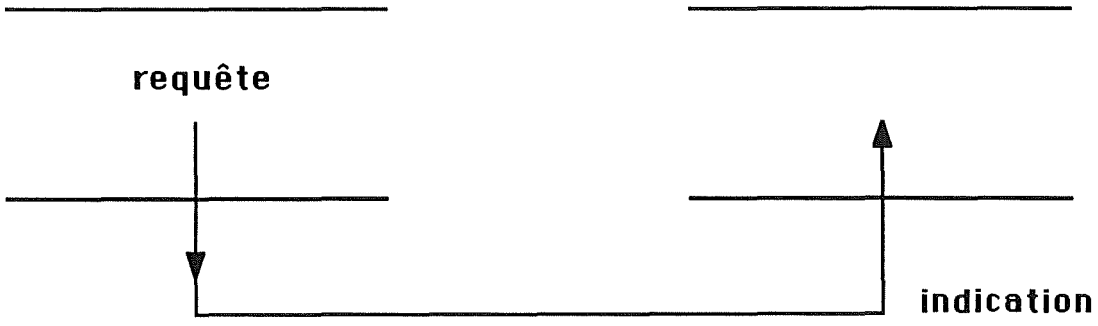

figure 2.4. service à 2 temps.

2.3.3. LE SERVICE A 3 TEMPS.

Le service à 3 temps est assez rare (un exemple de son utilisation est le RT-TRANSFER du RTSE). La différence avec le service à 4 temps est l'absence de réponse (figure 2.5.).

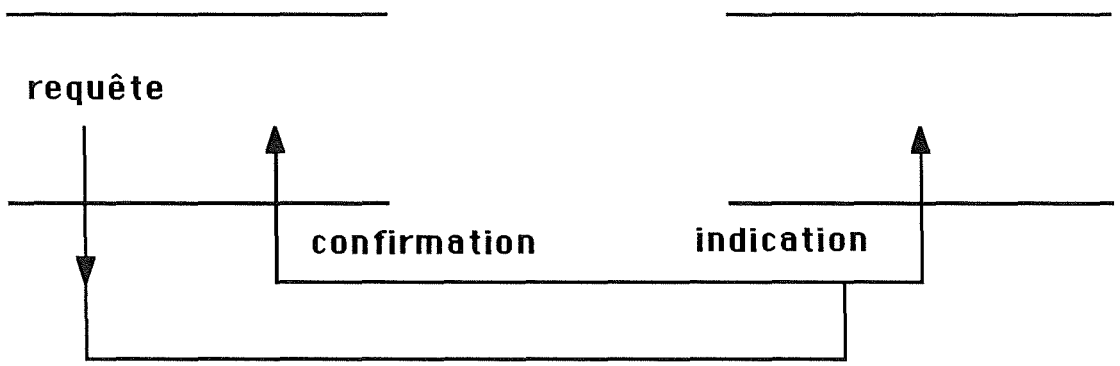

figure 2.5. le service à 3 temps.

2.3.4. SERVICE GENERE PAR LE FOURNISSEUR.

Il est possible que le fournisseur de service prenne l'initiative d'appeler l'utilisateur pour lui communiquer, par exemple, un incident grave qu'il a détecté et pouvant intéresser cet utilisateur (figure 2.6.).

figure 2.6. service généré par le fournisseur

Les lecteurs intéressés pourront consulter la norme (4) ou d'autres ouvrages (1), (2), (3) et surtout (5).

2.4. QUELQUES PARTICULARITES DE LA COUCHE 7.

2.4.1. GENERALITES.

La couche application a quelques particularités. Etant au sommmet de la pile OSI, elle ne fournit aucun service à une couche supérieure. De plus, c'est la couche 7 qui est en relation directe avec toutes les applications pouvant avoir besoin de fonctions de télécommunication. Ces particularités ont donné lieu à la définition de concepts propres à la couche7.

2.4.2. PROCESSUS APPLICATION.

Un **processus application** (abrégé par AP) représente une série de ressources au sein d'un système ouvert qui peut être utilisée dans une tâche informatique donnée. Il peut être invoqué par un programme ou par un utilisateur local ou encore par un autre AP. Il se compose de 2 éléments : une entité application et un agent application.

2.4.3. ENTITE APPLICATION.

Les aspects d'un AP relatifs à la télécommunication (et donc au modèle OSI) sont représentés par une ou plusieurs **entités applications** (abrégées par AE). Une AE correspond donc à un ensemble de capacités de communication d'un AP et peut être vue comme l'intersection du processus application et de la couche 7 du modèle de référence.

2.4.4. AGENT APPLICATION.

L'**agent application** réalise l'interface entre l'AE (OSI) et l'utilisateur (non OSI).

2.4.5. ELEMENTS DE SERVICE D'APPLICATION.

Une AE est vue comme un ensemble d'un ou plusieurs **éléments de service d'application** (abrégés ASE). Un ASE est un ensemble de fonctions de communication homogène qui fournit concrètement des capacités de communication OSI. Les capacités de communication entre 2 ASE du même type sont définies par un protocole, c'est à dire par la spécification d'une série

d'APDU et la procédure gouvernant leur utilisation. La figure 2.7. reprend la plupart de ces concepts.

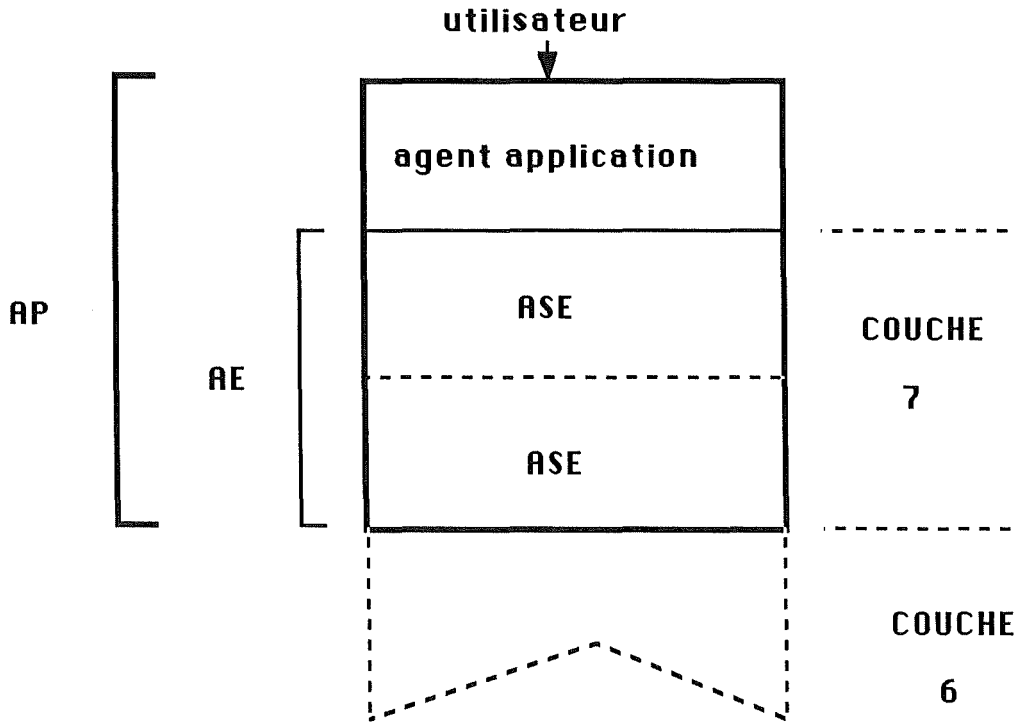

figure 2.7. éléments de la couche 7.

2.4.6. AP-INVOCATION.

Une **AP-invocation** représente l'activité d'un AP donné. A tout moment, un AP peut être représenté par zéro, une ou plusieurs AP-invocations.

2.4.7. AE-INVOCATION.

Une **AE-invocation** représente l'utilisation spécifique des capacités d'une AE, c'est à dire les activités spécifiques de communication d'une AP invocation.

2.4.8. ASSOCIATION D'APPLICATION.

Une **association** est un lien entre 2 AE-invocations pour leur permettre de communiquer et coordonner leurs opérations.

2.4.9. CONTEXTE D'APPLICATION.

Le **contexte d'application** réunit un ensemble de règles de fonctionnement sur la façon dont les AE-invocations doivent collaborer dans le cadre d'une association. Il comprend obligatoirement la spécification des ASE utilisés.

Des informations plus précises peuvent être obtenues dans (5).

2.4.10. EXEMPLES.

-Un logiciel utilisant ISODE + une machine = 1 AP.

-L'exécution de ce logiciel = 1 AP-invocation.

-L'implémentation d'ISODE de FTAM = 1 ASE.

3. ISODE.

3.1. OBJECTIF.

ISODE est un logiciel implémentant les couches supérieures du modèle ISO dont l'objectif avoué est, en tant qu'outil de recherche, de promouvoir la diffusion du modèle OSI principalement dans les communautés de recherche DARPA/NSF Internet (USA) et RARE (Europe). Il est diffusé et distribué librement mais sans aucune garantie d'aucune sorte. ISODE est décrit dans (6).

3.2. QU'EST CE QUE C'EST ?

3.2.1. GENERALITES.

Nous présentons ici la version 4.0. de juillet 1988. ISODE a été mis au point par la **NORTHROP CORPORATION** au **NORTHROP RESEARCH AND TECHNOLOGY CENTER** de PALO ALTO, en Californie (USA). Il est développé en C. Il fonctionne dans les environnements suivants : Berkeley UNIX, AT&T UNIX et d'autres dérivés d'UNIX comme AIX ou HP-UX.

ISODE implémente les éléments de service ACSE, ROSE, RTSE, les services session et présentation de même qu'une version minimale de FTAM. Il comprend aussi une interface entre les couches supérieures et différents services de transport possibles (TCP, TP0, TP4). Y sont aussi inclus quelques outils d'aide à la réalisation d'applications et le service de directory "QUIPU".

Les primitives de service du type "requête" et "réponse" sont des routines directement appelables. Les primitives de type "confirmation" sont réalisées par un retour de la valeur "OK" en fin d'exécution des routines de requête leur correspondant. Les primitives du type "indication" sont implémentées de plusieurs manières. Dans le cas d'un service initié par le fournisseur, l'indication se traduit par une terminaison anormale de la routine en cours d'exécution (celle-ci retourne la valeur "NOTOK"). Dans la plupart des autres cas, l'indication est réalisée par la fin d'exécution d'une routine d'attente d'informations.

3.2.2. LES COUCHES SUPERIEURES.

Les éléments de service d'application ACSE, RTSE et ROSE sont normalement implémentés. Au niveau présentation, seule l'unité fonctionnelle noyau est réalisée. Il n'y a donc aucune routine de modification de contexte. Les sous-ensembles BAS, BCS et BSS accompagnés de l'unité fonctionnelle de données exprès sont présents au niveau session.

3.2.3. TRANSPORT.

ISODE implémente une interface transport (figure 3.1). Sa fonction est de fournir à la couche 5 un service transport OSI alors qu'elle peut s'appuyer sur des services de transport non OSI choisis lors de l'installation d'ISODE. Le choix par défaut est TCP mais on peut sélectionner TP0 au dessus de X25 ou TP4.

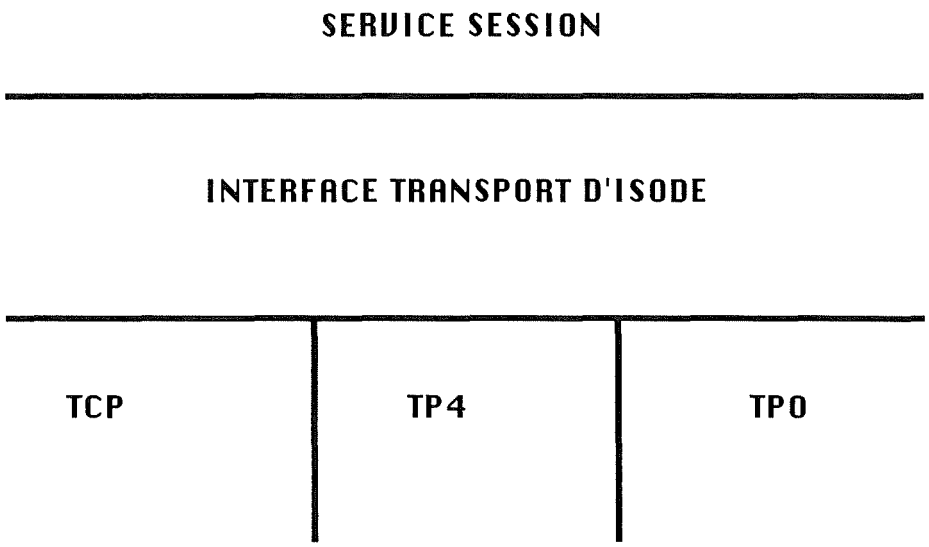

figure 3.1. l'interface transport d'ISODE.

3.2.4. LES OUTILS D'AIDE A LA CONCEPTION D'APPLICATIONS.

ISODE permet de développer des applications distribuées basées sur les opérations à distance. Sa méthodologie s'inspire de celle décrite dans : "Remote operations : concepts, notation and connection-oriented mappings, december 1985, ECMA TR/31". Elle se compose de 6 étapes :

1) Définir un nouveau service : on y précise les informations sur le nom et l'adresse de l'application.

2) Définir un module d'opérations à distance : l'utilisateur définit les opérations, les erreurs et la syntaxe abstraite des structures de données échangées par le service.

3) Définir les structures de données concrètes : définition des structures de données échangées.

4) Construire le programme initiateur.

5) Construire le programme répondeur.

6) Mettre tout ensemble : choix d'une politique consistante face aux noms des fichiers, aux règles du "make" (utilitaire de gestion de programmes), ...

ISODE offre comme aide différents outils :

1) Le programme **ROSY** lit une description du module des opérations à distance et produit les textes C correspondants et exécutables. Les 4 fichiers qu'il fournit sont : -un module de syntaxe abstraite,
-un fichier préprocesseur C contenant les définitions de chaque code d'opération, de chaque code d'erreur et de chaque morceau de routine,
-un fichier définissant les structures de données C utilisées lors de l'exécution,
-un fichier d'aide à la recherche des erreurs d'exécution.

2) A partir d'une description d'un module de syntaxe abstraite, le programme **POSY** produit les structures C correspondantes ainsi qu'un module de syntaxe abstraite amélioré qui sera utilisé par PEPY.

3) Le programme **PEPY**, à partir des données de POSY, produit un programme C qui reconnaît, génère ou imprime les objets décrits dans le module.

Notons quelques limites à ces outils : la notion de macro est inconnue, il existe quelques limites sémantiques pour les notions de COMPONENT OF, DEFAULT, CHOICE.

Des informations plus complètes peuvent être obtenues en consultant (7).

3.2.5. LE SERVICE DE DIRECTORY.

ISODE offre 2 options pour le service de directory. La première se base sur la base de données **isoentities**. Ce service est simple : à partir d'une

information d'entité application, on peut obtenir l'adresse correspondante en consultant isoentities. Dans ce cas, l'hôte maintient en permanence la liste de tous les destinataires possibles dont il connait l'adresse. C'est ce service qui est utilisé par défaut.

La seconde option s'inspire du service proposé par X.500. Il est implémenté par **QUIPU**. Dans ISODE 4.0., QUIPU est plutôt un prototype et ses possibilités sont encore limitées. Ainsi, il ne dispose pas d'interface utilisateur.

3.3. LES AUTRES VERSIONS D'ISODE.

3.3.1. VERSION 3.0.

La version 3.0. ne contenait pas les programmes ROSY et POSY. La syntaxe des bases de données isoentities et isoservices était différente. Le paramètre "qualité de service" n'existait pas, la notion d'information d'entité application non plus. De nombreuses routines et structures ont subi des modifications dans leurs paramètres, notamment pour respecter la norme sur l'ACSE. De nouvelles routines sont apparues comme SWriteRequest (service session).

3.3.2. VERSION 5.0.

ISODE 5.0. est normalement disponible depuis fin mars 1989. Il comprend, outre les éléments présents dans la version 4.0. :
-une passerelle FTAM/FTP,
-des services de directory ISO,
-un terminal virtuel (classe de base),
-une implémentation des versions X400 1984 du ROS et du RTS.
Signalons que le ROSE et le RTSE sont fidèles au "draft" de mars 1988 et que VT est une implémentation "DIS".

Les développements ultérieurs prévus sont :
-un système de gestion de messages,
-une passerelle MHS/SMTP.

4. L'ACSE.

4.1. RAPPEL THEORIQUE.

4.1.1. FONCTION.

L'**élément de service de contrôle d'association**, souvent abrégé par **ACSE** (pour Association Control Service Element) est un ASE obligatoirement utilisée par toute application désirant effectuer de la communication de données.

Son rôle consiste à établir entre 2 AE-invocations (pour le compte de leur AP-invocation respective) une sorte de connexion de niveau 7 appelée **association**. L'établissement de l'association permet aux applications de négocier divers paramètres dont ne s'occupe aucune autre couche du modèle OSI. Ces paramètres sont regroupés au sein du contexte d'application. Les éléments de service sont repris dans la figure 4.1.

SERVICE	TYPE
A-ASSOCIATE	confirmé
A-RELEASE	confirmé
A-ABORT	non confirmé
A-P-ABORT	généré par le fournisseur

figure 4.1. les éléments de service de l'ACSE.

4.1.2. LES ELEMENTS DE SERVICE.

Nous présentons ici brièvement les éléments de service. Leurs paramètres sont repris dans l'annexe 1.

1) A-ASSOCIATE.

Le **A-ASSOCIATE** permet d'établir une association entre deux AE-INVOCATIONS. C'est un service confirmé à 4 temps (figure 4.2.). Le A-ASSOCIATE fait appel à la primitive P-CONNECT de la couche 6.

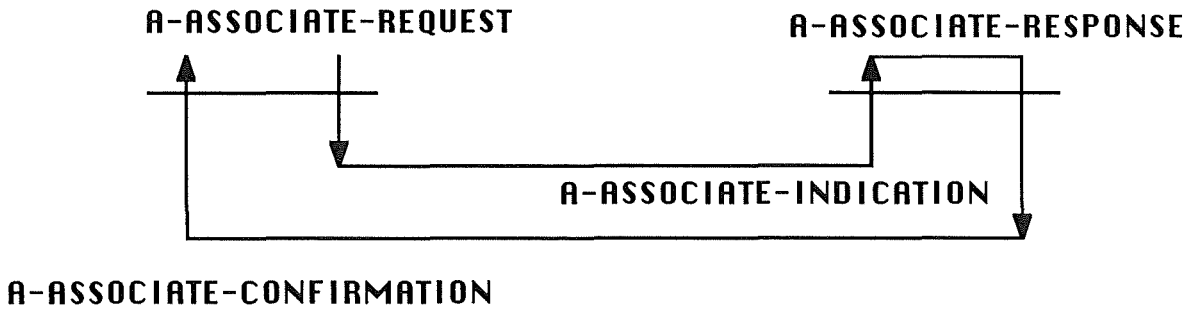

figure 4.2. le A-ASSOCIATE.

2) A-RELEASE.

Le **A-RELEASE** négocie l'abandon d'une association. C'est un service confirmé à 4 temps semblable au précédent. Il existe un lien 1-1 entre le A-RELEASE et le P-RELEASE.

3) A-ABORT.

Le **A-ABORT** a pour effet de rompre immédiatement une association. C'est un service non confirmé à 2 temps (figure 4.3.).

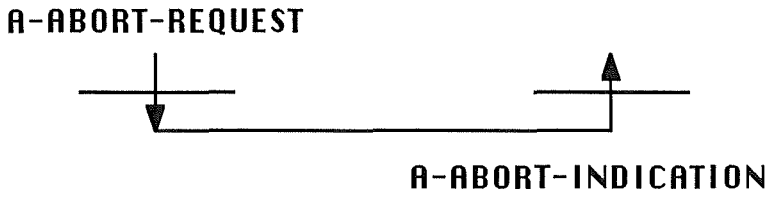

figure 4.3. le A-ABORT.

4) A-P-ABORT.

La primitive **A-P-ABORT** est utilisée par le fournisseur ACSE pour signaler un abandon de l'association dû à un problème rencontré par la couche application. Son effet est de rompre immédiatement l'association en cours (figure 4.4.).

figure 4.4. le A-P-ABORT.

Le service est défini en détails dans (8) et de manière plus succinte dans (7).

4.1.3. LE PROTOCOLE ACSE.

Il existe une différence importante entre le protocole des couches supérieures et celui des couches inférieures. En effet, dans les couches inférieures, un paquet d'ouverture est transmis dans un paquet de données des niveaux inférieurs. Dans les couches supérieures, un tel paquet est transmis dans un paquet d'ouverture des niveaux inférieurs grâce aux données de l'utilisateur. Montrons cela sur un exemple.

Recevant de la couche 4 une demande de connexion, la couche 3 va d'abord demander à la couche 2 de se connecter avec l'hôte éloigné comme le montre la figure 4.5. Ensuite (figure 4.6), la couche 3 emballe le NPDU de demande de connexion transport dans un paquet de données et l'expédie à la couche 2 via un DL-DATA.

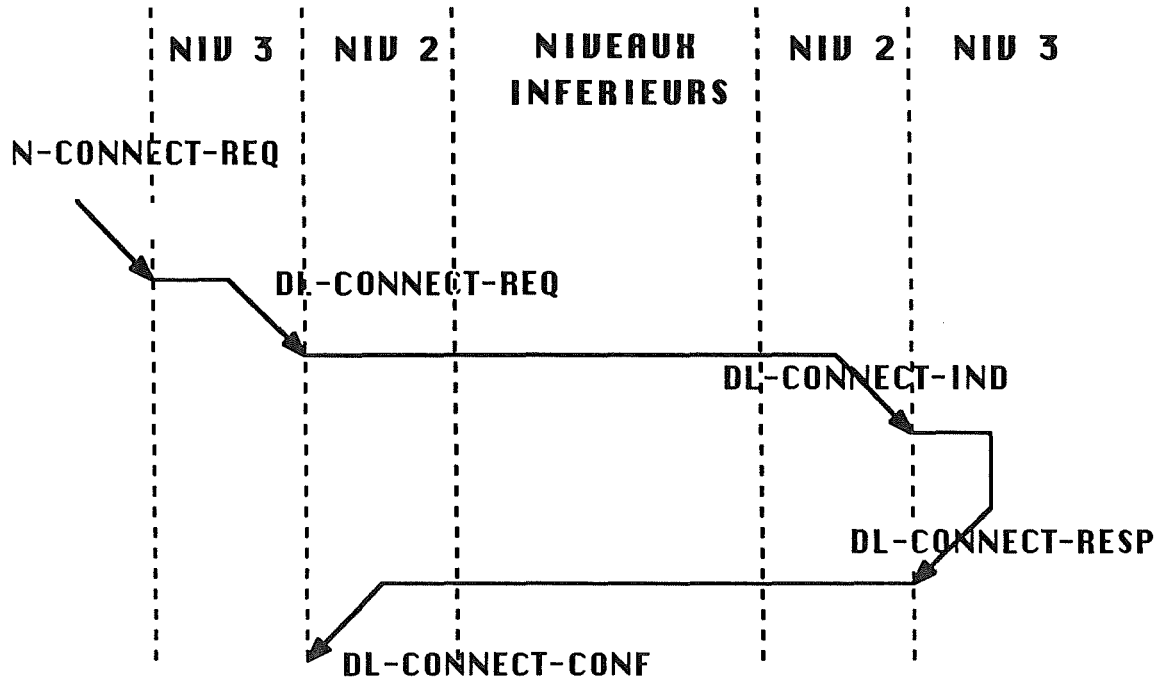

figure 4.5. mécanisme d'ouverture d'une connexion au niveau de la couche 3.

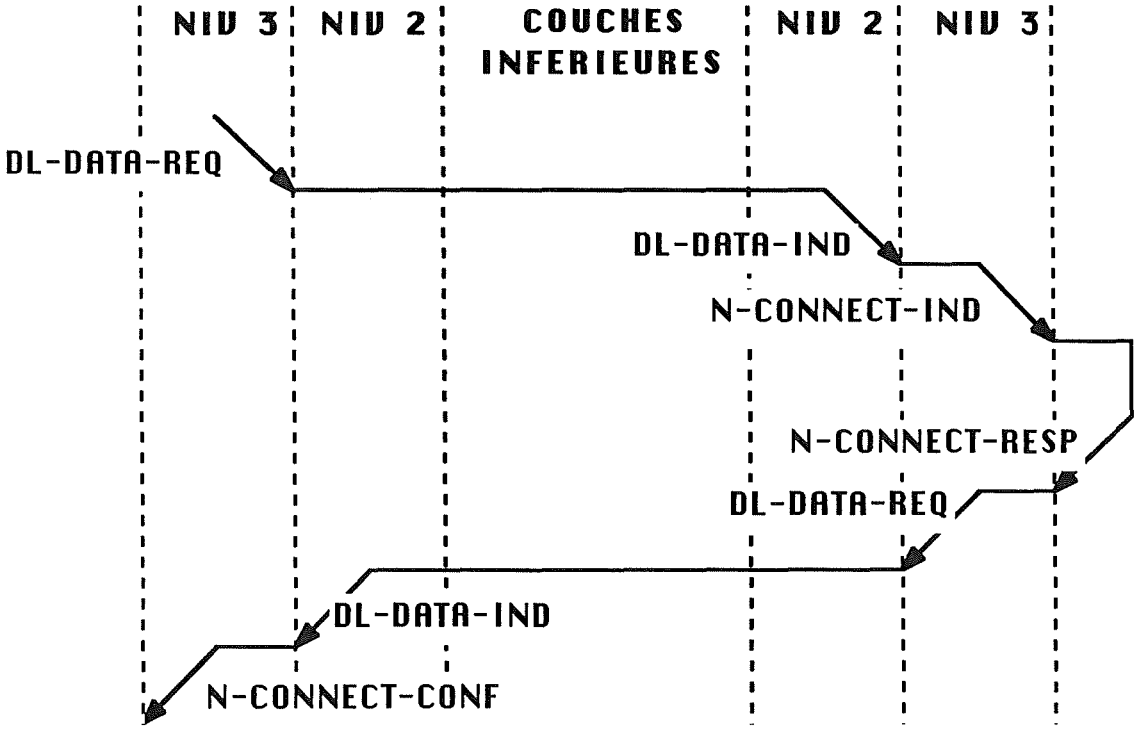

figure 4.6. mécanisme d'établissement d'une connexion dans les couches inférieures : suite.

Dans le cas des couches supérieures, le mécanisme est plus simple et plus rapide. Dans l'exemple de la figure 4.7., le PDU AARQ est véhiculé comme information de l'utilisateur du PDU de P-CONNECT et ainsi de suite.

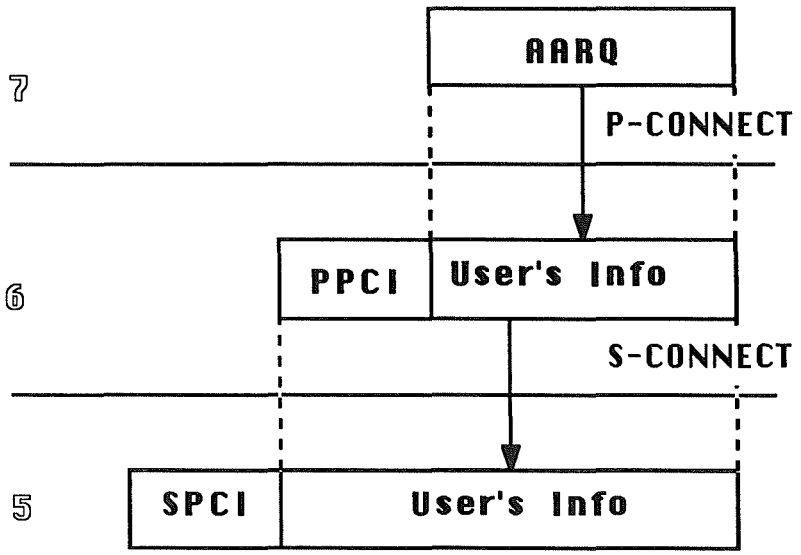

figure 4.7. mécanisme d'établissement d'une connexion dans les couches supérieures : le PDU AARQ est véhiculé comme information de l'utilisateur du P-CONNECT PDU et ainsi de suite.

Les différents APDU de l'ACSE sont repris dans la figure 4.8. La liste des paramètres de ces différents APDU fait l'objet de l'annexe 2. La figure 4.9. montre la correspondance entre l'ACSE et le service présentation. Au centre les APDU de l'ACSE, à droite les éléments de service de la couche 6 qui les prennent en charge.

APDU	FONCTION	PRIMITIVE CREATRICE
AARQ	Demande d'association	A-Associate-Req
AARE	Réponse à l'association	A-Associate-Resp
RLRQ	Demande d'abandon d'association	A-Release-Req
RLRE	Réponse à une demande d'abandon	A-Release-Resp
ABRT	Terminaison anormale d'une association	A-Abort-Req

figure 4.8. les APDU de l'ACSE.

PRIMITIVES ACSE	APDU	PRIMITIVES PRESENTATION
A-ASSOCIATE-REQ/IND	AARQ	P-CONNECT-REQ/IND
A-ASSOCIATE-RESP/CONF	AARE	P-CONNECT-RESP/CONF
A-RELEASE-REQ/IND	RLRQ	P-RELEASE-REQ/IND
A-RELEASE-RESP/CONF	RLRE	P-RELEASE-RESP/CONF
A-ABORT-REQ/IND	ABRT	P-U-ABORT-REQ/IND
A-P-ABORT-IND	-	P-P-ABORT-IND

figure 4.9. correspondance entre l'ACSE et le service présentation.

La définition précise du protocole est présentée dans (9).

4.2. L'ACSE D'ISODE.

4.2.1. GENERALITES.

La librairie **libcasap** contient toutes les routines implémentant l'ACSE. L'ACSE va établir et gérer une association entre deux utilisateurs : l'initiateur (celui qui demande l'association) et le répondeur. Les primitives de service sont implémentées sous forme de routines, chacune retournant 2 valeurs possibles : OK ou NOTOK. L'implémentation de l'ACSE est décrite dans (10).

4.2.2. L'IMPLEMENTATION DES PRIMITIVES DE SERVICE.

Nous allons maintenant reprendre chaque élément de service et voir comment ISODE l'implémente.

1) A-ASSOCIATE .

Les différentes routines impliquées dans le A-ASSOCIATE sont reprises dans la figure 4.10.

PRIMITIVES ISO	ROUTINES ISODE
A-ASSOCIATE-REQ	AcAssocRequest
A-ASSOCIATE-IND	(AcInit)
A-ASSOCIATE-RESP	AcAssocResponse
A-ASSOCIATE-CONF	(AcAssocRequest)

figure 4.10. le A-ASSOCIATE.
Les parenthèses indiquent que la primitive n'est pas implémentée par l'appel à la routine associée mais par la fin d'exécution de celle-ci.

Nous allons maintenant décrire l'enchaînement de ces routines.

-L'initiateur lance l'exécution de AcAssocRequest.

-La demande d'association arrive chez le répondeur. Elle est examinée par un

démon qui va déterminer l'application visée. Il lance alors l'exécution de cette application.

-La première chose qu'elle réalise est de recapturer l'état du dialogue amorcé par le démon. Cela est réalisé par un appel à AcInit.

-La terminaison normale de AcInit correspond à un A-ASSOCIATE-IND. L'application est informée de la demande d'établissement d'association.

-L'application répond à l'association en appelant AcAssocResponse.

-La réponse arrive chez le client. Cela se traduit par la terminaison de la routine AcAssocRequest, synonyme de A-ASSOCIATE-CONFIRMATION.

Voici les arguments de AcInit :

int	vecp	: longueur du vecteur argument,
char	**vec	: vecteur argument,
struct AcSAPstart	*acs	: structure modifiée si l'appel réussit,
struct AcSAPindication	*aci	: structure modifiée si l'appel échoue.

Les figures 4.11. et 4.12. comparent les arguments des autres routines avec ceux définis dans la norme.

ISO	ISODE
mode	-
contexte d'application	context
titre de l'AP appelant identificateur d'invocation de l'AP appelant qualificatif de l'AE appelant identificateur d'invocation de l'AE appelante	callingtitle
titre de l'AP appelé identificateur d'invocation de l'AP appelé qualificatif de l'AE appelé identificateur d'invocation de l'AE appelé	calledtitle
adresse présentation appelante	callingaddr
adresse présentation appelée	calledaddr
liste de contextes de présentation	ctxlist
nom du contexte par défaut	defctxname
qualité de service	qos
exigences de présentation	prequirements
exigences de session	srequirements
numéro du point de synchronisation initial	isn
initialisation des jetons	settings
identificateur de connexion session	ref
données de l'utilisateur	data ndata
-	acc
-	aci

figure 4.11. comparaison des arguments de AAssociateRequest avec la norme.

"Acc" pointe vers une structure AcSAPconnect modifiée si l'appel réussit. "Aci" pointe vers une structure AcSAPindication modifiée en cas d'échec. "Data" contient les données de l'utilisateur et "ndata" en donne la longueur.

ISO	ISODE
contexte d'application	context
titre de l'AP répondant identificateur d'invocation de l'AP répondant qualificatif de l'AE répondante identificateur d'invocation de l'AE répondante	respondtitle
adresse présentation du répondeur	respondaddr
liste de contextes de présentation	ctxlist
exigences de présentation	prequirements
exigences de session	srequirements
numéro du point de synchronisation initial	isn
initialisation des jetons	settings
identificateur de connexion session	ref
données de l'utilisateur	data ndata
résultat	status
contexte de présentation par défaut	defctxresult
qualité de service	-
diagnostic	reason
-	aci
-	sd

figure 4.12. comparaison des arguments de AcAssocResponse avec la norme.

"Aci" contient un pointeur vers une structure AcSAPindication et "sd" est un descripteur utilisé pour identifier l'association (c'est un nombre entier).

2) A-RELEASE.

Les routines du A-RELEASE sont résumées dans la figure 4.13.

PRIMITIVES ISO	ROUTINES ISODE
A-RELEASE-REQ	AcRelRequest
A-RELEASE-IND	(AcFINISHser)
A-RELEASE-RESP	AcRelResponse
A-RELEASE-CONF	(AcRelRequest)

figure 4.13. le A-RELEASE.

Un utilisateur désirant demander l'abandon d'une association effectue un appel à **AcRelRequest.** Dans son exécution, la routine appelle son homologue du niveau 6 : PRelRequest. Les arguments de la routine et leur comparaison avec la norme font l'objet de la figure 4.14.

ISO	ISODE
raison	reason
-	sd
données de l'utilisateur	data ndata
-	acr
-	aci

figure 4.14. comparaison des arguments de AcRelRequest avec la norme.

"Acr" est un pointeur vers une structure AcSAPrelease modifiée si tout marche correctement. Les autres ont déjà été définis plus haut.

Lorsque l'hôte éloigné détectera un P-RELEASE-INDICATION, il fera appel à l'ACSE grâce à la routine **AcFINISHser(sd, pf, aci)**

int sd : descripteur de l'association,
struct PSAPfinish *pf : info. sur l'abandon au niveau présentation,
struct AcSAPindication *aci : autres informations sur la demande d'abandon.

La fin de son exécution est synonyme de A-RELEASE-INDICATON. L'utilisateur éloigné doit maintenant répondre grâce à un appel à **AcRelResponse**. AcRelResponse passe la main au niveau 6 en appelant PRelResponse. La comparaison des paramètres de la routine avec la norme est réalisée dans la figure 4.15.

ISO	ISODE
raison	reason
résultat	status
données de l'utilisateur	data ndata
-	sd
-	aci

figure 4.15. comparaison des arguments de AcRelResponse avec la norme.

Sur l'hôte local, un A-RELEASE-CONFIRMATION apparaîtra si la routine AcRelRequest retourne la valeur OK comme résultat.

3) A-ABORT.

L'implémentation du A-ABORT est présentée à la figure 4.16.

Primitives ISO	Routines ISODE
A-ABORT-REQ	AcUAbortRequest
A-ABORT-IND	(AcABORTser)

figure 4.16. le A-ABORT.

Pour abandonner brutalement une association, il faut faire appel à **AcUAbortRequest.** La figure 4.17. compare la norme à la routine d'ISODE.

ISO	ISODE
raison	-
données de l'utilisateur	data ndata
-	sd
-	aci

figure 4.17. comparaison des arguments de AcUAbortRequest avec la norme.

Du côté de l'hôte éloigné, le A-ABORT-INDICATION est implémenté par un appel à **AcABORTser(sd, pa, aci)**

int sd : descripteur de l'association,
struct PSAPabort *pa : info. sur l'abandon au niveau présentation,
struct AcSAPindication *aci : info. supplémentaires.

4) A-P-ABORT.

La routine **AcABORTser** est également utilisée pour transformer un P-P-ABORT-INDICATION en un A-P-ABORT-INDICATION.

4.2.3. LES STRUCTURES D'INFORMATIONS.

Les différentes routines d'ISODE contiennent parmi leurs arguments un ou deux pointeurs vers autant de structures C que nous appellerons " structures d'informations". Celles-ci contiennent des informations sur les résultats obtenus par les routines et pouvant être utiles à l'utilisateur du service (ici, le service ACSE). Elles sont résumées dans la figure 4.18. Leurs paramètres sont repris dans l'annexe 3.

ROUTINES	STRUCTURES ASSOCIEES
AcAssocRequest	AcSAPconnect (AcSAPindication)
AcAssocResponse	(AcSAPindication)
AcInit	AcSAPstart (AcSAPindication)
AcRelRequest	AcSAPrelease (AcSAPindication)
AcRelResponse	(AcSAPindication)
AcUAbortRequest	(AcSAPindication)
AcFINISHser	PSAPfinish AcSAPindication
AcABORTser	PSAPabort AcSAPindication

figure 4.18. les routines et leurs structures associées, les parenthèses indiquent les structures modifiées en cas d'erreur de la routine.

Passons les rapidement en revue :

1) ACSAPCONNECT.

Cette structure est modifiée si l'appel à AcAssocRequest réussit : elle contient alors la réponse à l'association de l'AE éloignée.

2) ACSAPSTART.

Cette structure contient les propositions et informations diverses relatives à une demande d'association.

3) ACSAPRELEASE.

AcSAPrelease est modifiée si l'appel à AcSAPrelease réussit. Elle comprend la réponse à la demande d'abandon de l'association.

4) ACSAPINDICATION.

AcSAPindication a trois fonctions :

-Elle contient les paramètres d'un problème éventuellement survenu lors d'un appel à AcAssocRequest, AcAssocResponse, AcRelRequest ou AcRelResponse.

-Elle comprend les informations concernant un abandon négocié d'association dans le cas d'un appel à AcFINISHser.

-Elle contient enfin les informations utiles sur un abandon brutal dans l'hypothèse d'un appel à AcUAbortRequest ou à AcABORTser.

Les autres structures seront vues dans la partie consacrée au service présentation.

4.2.4. L'ADRESSAGE.

Voici comment fonctionne au niveau de l'adressage une application utilisant ISODE. A partir d'un descripteur d'une information d'entité application (constituée de la paire hôte et service), le client va consulter la base de données **isoentities** (voir chapitre 11) contenant des informations sur les AE connues de lui-même grâce à la routine **str2aei.** Si ce couple existe, la routine retourne l'identifiant de l'information de l'entité application. C'est à lui que correspondent les arguments "callingtitle" et "calledtitle" de AcAssocRequest. A partir d'un tel identifiant, la routine **aei2addr** retourne l'adresse présentation appropriée. Celle-ci se compose d'un sélecteur de présentation et d'une adresse session. Le même mécanisme est valable aux niveaux session et transport, chaque adresse de niveau N contenant une adresse de niveau N-1. Voici pour l'exemple la construction d'une adresse FTAM :

```
if(aei=str2aei(RemoteHost,"filestore"))=NULLAEI
  error("%s-filestore : unknown application_entity",RemoteHost);
if(pa=aei2addr(aei))==NULLPA
  error("address translation failed");
```

5. LE RTSE.

5.1. RAPPEL THEORIQUE.

5.1.1. FONCTION.

Pour certaines applications, il est important d'avoir un service de transfert des APDU parfaitement fiable. C'est à cette fin qu'a été créé l'élément de service de transfert fiable ou **RTSE** (pour Reliable Transfer Service Element). Le RTSE utilise l'ACSE pour établir une association (figure 5.1.). Il nécessite aussi les services Session BAS à travers la couche présentation. Il implique enfin un dialogue en "two-way-alternate".

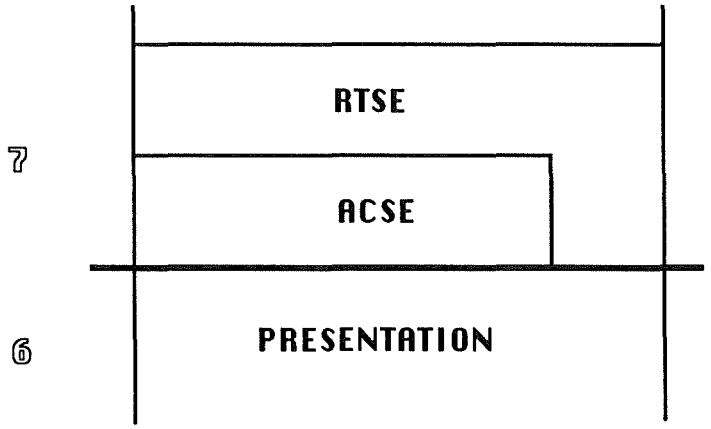

figure 5.1. situation du RTSE par rapport à l'ACSE.

Les éléments de service du RTSE sont repris dans la figure 5.2.

SERVICE	TYPE	FONCTION
RT-OPEN	confirmé	ouverture d'une association
RT-CLOSE	confirmé	fermeture négociée d'une association
RT-TRANSFER	confirmé	expédition d'informations
RT-TURN-PLEASE	non confirmé	demande du tour
RT-TURN-GIVE	non confirmé	cession du tour
RT-P-ABORT	créé par le fournisseur	coupure par le RTSE de l'association
RT-U-ABORT	non confirmé	demande de coupure brutale d'une association

figure 5.2. éléments de service du RTSE.

5.1.2. ELEMENTS DE SERVICE.

Nous passons en revue les éléments de service. Leurs paramètres font l'objet de l'annexe 4.

1) RT-OPEN.

Le **RT-OPEN** permet d'établir une association. C'est un service à 4 temps qui impliquera de la part du RTSE un appel à l'ACSE.

2) RT-CLOSE.

Le **RT-CLOSE** permet de négocier l'abandon d'une association. C'est un service à 4 temps faisant aussi appel à l'ACSE.

3) RT-TRANSFER.

Le service de **RT-TRANSFER** donne la possibilité à un utilisateur du RTSE, propriétaire du tour, d'envoyer ses APDU. C'est un service confirmé un peu spécial : il n'a que trois temps comme le montre la figure 5.3.

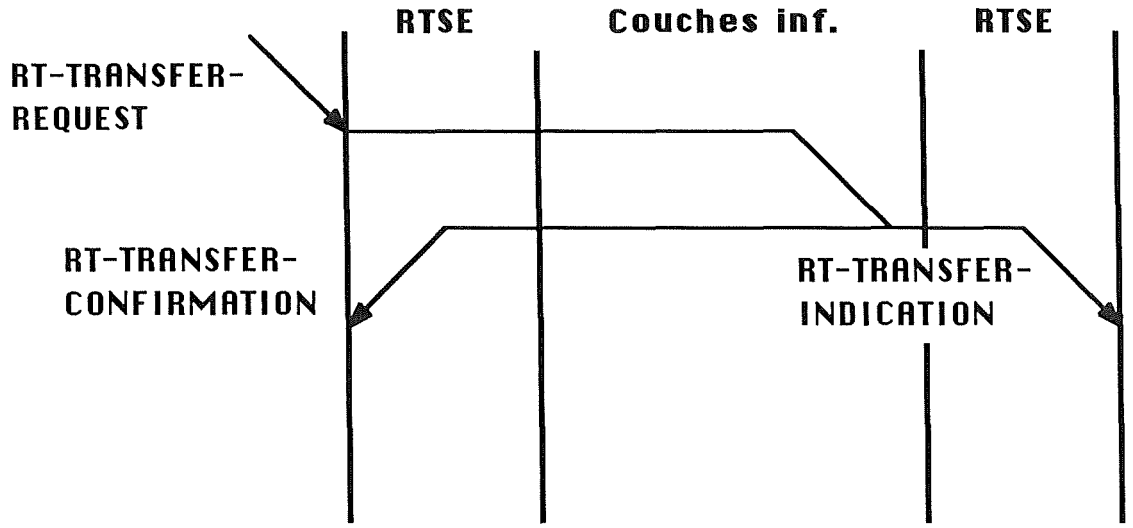

figure 5.3. le RT-TRANSFER.

4) RT-TURN-PLEASE.

Le service de **RT-TURN-PLEASE** a pour but de demander le tour, c'est à dire l'autorisation d'émettre des données. C'est un service non confirmé à 2 temps.

5) RT-TURN-GIVE.

Le **RT-TURN-GIVE** cède le tour de parole au correspondant éloigné. C'est un service à 2 temps, non confirmé et sans paramètre.

6) RT-P-ABORT.

Le **RT-P-ABORT** donne au fournisseur RTSE la possibilité de signaler à son utilisateur qu'en raison d'un grave problème l'association ne peut être maintenue. Ce service n'a aucun paramètre.

7) RT-U-ABORT.

Le **RT-U-ABORT** fournit à un utilisateur du RTSE la faculté de couper brutalement une association.

Le service est décrit dans (11) et (7).

5.1.3. LE PROTOCOLE.

La figure 5.4 résume tous les APDU utilisés par le RTSE.

APDU	FONCTION
RTORQ	demande de création d'une association
RTOAC	acceptation d'une association
RTORJ	refus d'une association
RTTR	transfert de données
RTAB	abandon d'une association
RTTP	demande du tour

figure 5.4. les APDU du RTSE.

Les paramètres de ces APDU sont repris dans l'annexe 5. Nous décrivons ici brièvement les éléments de procédure.

1) ETABLISSEMENT DE L'ASSOCIATION.

L'initiateur va émettre un **RTORQ APDU.** Un tel APDU sera transmis à l'ACSE par un A-ASSOCIATE-REQUEST. Le répondeur va accepter ou non cette demande. S'il l'accepte, il émet un **RTOAC APD.** S'il refuse, il expédie plutôt un **RTORJ APDU.** Ces 2 APDU seront transmis par un A-ASSOCIATE-RESPONSE.

2) ABANDON NEGOCIE D'ASSOCIATION.

Au niveau RTSE, aucun APDU n'est utilisé pour abandonner une association. La procédure est résumée à la figure 5.5.

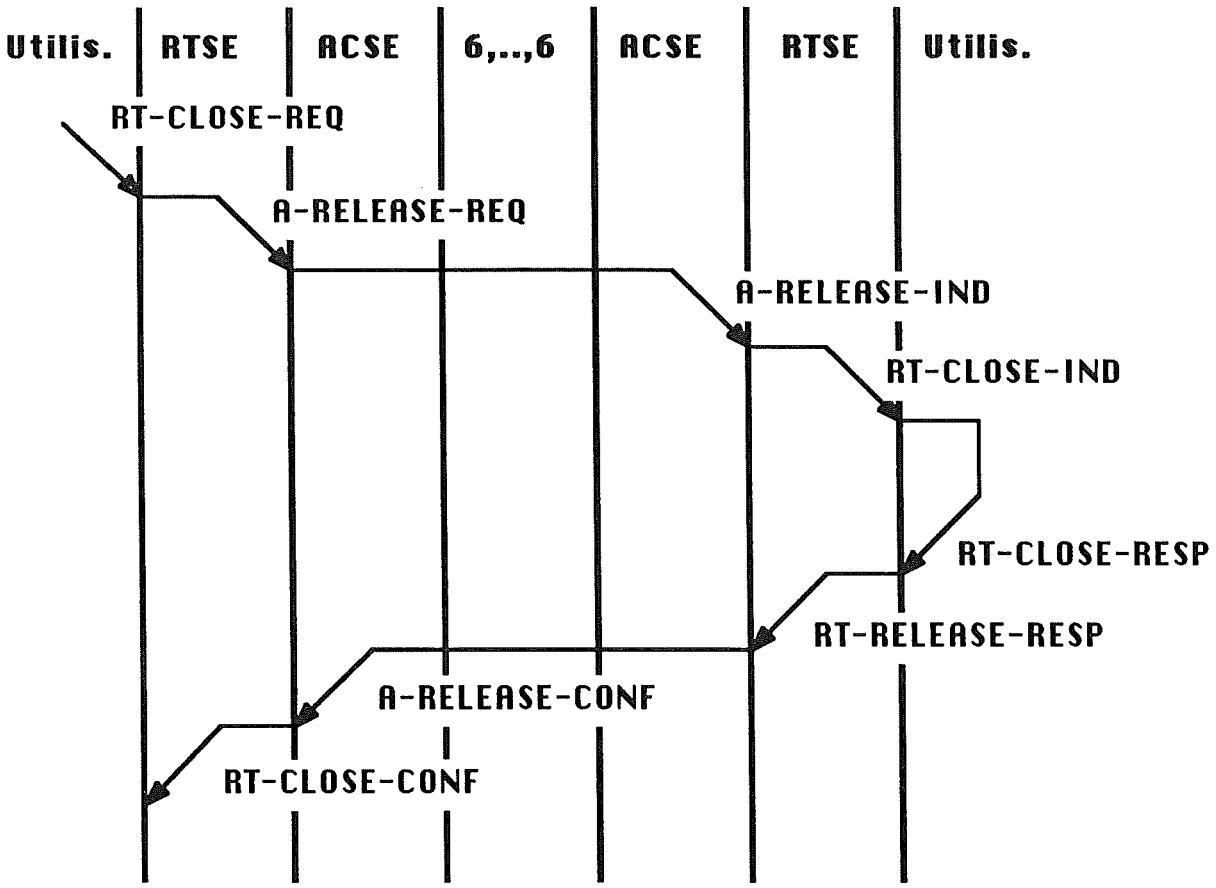

figure 5.5. abandon d'association.

3) TRANSFERT DE DONNEES.

Le transfert de données nécessite la séquence d'actions suivante :

a) Appel de la part de l'utilisateur à la primitive RT-TRANSFER-REQUEST. Cet appel va permettre de transférer un **RTTR APDU.**

b) Ouverture par le RTSE d'une activité (P-ACTIVITY-START-REQUEST).

c) Exécution d'une série d'appels à P-DATA-REQUEST (pour transférer l'APDU) et à P-MINOR-SYNCHRONIZE (pour poser les points de synchronisation mineurs) avec d'éventuelles attentes d'une confirmation pour chaque point posé.

d) Cloture de l'activité (P-ACTIVITY-END-REQUEST).

e) Attente de la confirmation de cette fin d'activité de la part du correspondant (attente d'un P-ACTIVITY-END-CONFIRMATION).

f) Signalisation à l'utilisateur de la fin du travail (RT-TRANSFER-CONFIRMATION).

4) DEMANDE DU TOUR.

La demande du tour se traduit par l'envoi d'un **PTTP APDU** qui sera transmis grâce à la primitive P-TOKEN-PLEASE-REQUEST.

5) CESSION DU TOUR.

Aucun APDU n'est utilisé pour céder le tour. Simplement, la cession se traduira par un appel à P-CONTROL-GIVE-REQUEST de la part du RTSE local. Le RTSE éloigné, recevant un P-CONTROL-GIVE-INDICATION, exécutera un P-CONTROL-GIVE-RESPONSE vers le RTSE local puis un RT-TURN-GIVE-INDICATION vers son propre utilisateur pour l'informer qu'il possède le tour de parole.

6) ABANDON BRUTAL DE L'ASSOCIATION.

Si un problème grave survient (chez le fournisseur ou l'utilisateur) et que l'association doit être rompue, le RTSE émet un **RTAB APDU**. Cet APDU pourra être transmis comme donnée de l'utilisateur par un A-ABORT-REQUEST.

Le protocole est défini dans (12).

5.2. LE RTSE D'ISODE.

5.2.1. GENERALITES.

La librairie **librtsap** implémente le RTSE d'ISODE. Les primitives de service sont implémentées sous forme de routines, chacune retournant la valeur OK ou NOTOK comme résultat de son exécution. L'implémentation est décrite dans (10).

5.2.2. L'IMPLEMENTATION DES PRIMITIVES DE SERVICE.

Nous reprenons ici chaque élément de service et montrons comment ISODE l'implémente.

1) RT-OPEN.

Les routines du RT-OPEN sont résumées dans la figure 5.6. Leur enchaînement est semblable à celui présenté dans l'ACSE pour l'implémentation du A-ASSOCIATE. Les figures 5.7 et 5.8. comparent les routines d'ISODE avec la norme.

PRIMITIVES ISO	ROUTINES D'ISODE
RT-OPEN-REQUEST	RtOpenRequest
RT-OPEN-INDICATION	(RtInit)
RT-OPEN-RESPONSE	RtOpenResponse
RT-OPEN-CONFIRMATION	(RtOpenRequest)

figure 5.6. le RT-OPEN.

ISO	ISODE
mode de dialogue	mode
tour initial	tour
protocole application	-
contexte d'application	context
titre de l'AE appelante	callingtitle
titre de l'AE appelée	calledtitle
adresse présentation appelante	callingaddr
adresse présentation appelée	calledaddr
liste de contextes de présentation	ctxlist
nom du contexte par défaut	defctxname
données de l'utilisateur	data
-	qos
-	rtc
-	rti

figure 5.7. comparaison des arguments de RtOpenRequest avec la norme.

Signalons que "qos" représente la qualité de service, que "rtc" est un pointeur vers une structure RtSAPconnect modifiée en cas de réussite de l'appel et que "rti" est un pointeur vers une structure RtSAPindication utile en cas d'échec. Notons que le protocole application, paramètre absent dans ISODE, est optionnel. Il n'est utilisé que dans le mode X410-1984.

ISO	ISODE
-	sd
contexte d'application	context
titre de l'AE répondante	respondtitle
résultat	status
adresse présentation du répondeur	respondaddr
liste de contextes de présentation	ctxlist
nom du contexte par défaut	defctxresult
données de l'utilisateur	data
-	rti

figure 5.8. comparaison des arguments de RtOpenResponse avec la norme.

Quant aux paramètres de RtInit, les voici :

int	vecp	: longueur du vecteur argument,
char	**vec	: vecteur argument,
struct RtSAPstart	*rts	: modifié si l'appel réussit,
struct RtSAPindication	*rti	: modifié si l'appel échoue.

2) RT-CLOSE.

La figure 5.9. reprend toutes les routines impliquées par le RT-CLOSE.

PRIMITIVES ISO	ROUTINES ISODE
RT-CLOSE-REQ	RtCloseRequest
RT-CLOSE-IND	(RtWaitRequest)
RT-CLOSE-RESP	RtCloseResponse
RT-CLOSE-CONF	(RtCloseRequest)

figure 5.9. le RT-CLOSE.

La fonction de la routine **RtWaitRequest(sd, secs, rti)** est d'attendre des informations sur une association. Ses paramètres sont :
int sd : descripteur de l'association,
secs : temps d'attente maximum en secondes,
struct RtSAPindication *rti : info. sur l'événement survenant.

Elle retourne 3 valeurs :
-NOTOK en cas d'échec,
-OK si des données sont arrivées,
-DONE en cas de survenance d'autres chose que de simples données.

Le cas DONE couvre 3 événements : RT-TURN-PLEASE-INDICATION,
RT-TURN-GIVE-INDICATION,
RT-CLOSE-INDICATION.

Voyons maintenant les différentes étapes du RT-CLOSE sur ISODE.

-Le répondeur, n'ayant pas la parole, est en attente d'informations. Cette attente se marque par un appel à RtWaitRequest.

-L'initiateur décide de cloturer l'association. Il appelle RtCloseRequest.

-La demande de cloture arrive chez le répondeur, provoquant la terminaison de RtWaitRequest en retournant la valeur DONE.

-En examinant la structure RtSAPindication, le répondeur voit de quoi il s'agit et répond à la requête en appelant RtCloseResponse.

-La réponse arrive chez l'initiateur, provoquant la fin de l'appel à RtCloseRequest (synonyme de RT-CLOSE-CONFIRMATION.). La routine retourne la valeur OK.

Les figures 5.10. et 5.11. comparent les arguments des routines avec la norme.

ISO	ISODE
raison	reason
données de l'utilisateur	data
-	sd
-	acr
-	rti

figure 5.10. comparaison des arguments de RtCloseRequest avec la norme.

Signalons que "Acr" est un pointeur vers une structure AcSAPrelease utile si l'appel réussit.

ISO	ISODE
raison	reason
données de l'utilisateur	data
-	sd
-	rti

figure 5.11. comparaison des arguments de RtCloseResponse avec la norme.

3) RT-U-ABORT .

Le RT-U-ABORT est implémenté comme le décrit la figure 5.12.

PRIMITIVES ISO	ROUTINES ISODE
RT-U-ABORT-REQUEST	RtUAbortRequest
RT-U-ABORT-INDICATION	(RtWaitRequest)

figure 5.12. le RT-U-ABORT.

Si un utilisateur décide de couper brutalement une association sans négociation préalable (en cas de problème grave), il appelle la routine **RtUAbortRequest.** La figure 5.13. réalise la comparaison entre les arguments de la routine et la norme.

ISO	ISODE
données de l'utilisateur	data
-	sd
-	rti

figure 5.13. comparaison des arguments de RtUAbortRequest avec la norme.

Cette routine appellera son homologue ACSE à savoir AcUAbortRequest. Le RT-U-ABORT-INDICATION se matérialisera sur l'hôte éloigné par la valeur NOTOK retournée par RtWaitRequest.

4) RT-TRANSFER.

Les routines du RT-TRANSFER sont reprises dans la figure 5.14.

PRIMITIVES ISO	ROUTINES ISODE
RT-TRANSFER-REQUEST	RtTransferRequest
RT-TRANSFER-INDICATION	(RtWaitRequest)
RT-TRANSFER-CONFIRMATION	(RtTransferRequest)

figure 5.14. le RT-TRANSFER.

Voulant transférer des données, l'utilisateur appelle **RtTransferRequest.** Cette routine appellera lors de son exécution son homologue de la couche 6 : PDataRequest. La figure 5.15. réalise la comparaison de ses arguments avec ceux de la norme.

ISO	ISODE
APDU	data
temps maximum de transfert	secs
-	sd
-	rti

figure 5.15. comparaison des arguments de RtTransferRequest avec la norme.

L'indication d'arrivée de données chez le correspondant éloigné se manifestera par un retour de la valeur OK de la part de RtWaitRequest. Un retour de OK par RtTransferRequest sera le reflet d'un RT-TRANSFER-CONFIRMATION.

5) RT-TURN-PLEASE.

Le RT-TURN-PLEASE est implémenté par les routines de la figure 5.16.

PRIMITIVES ISO	ROUTINES ISODE
RT-PLEASE-TURN-REQ	RtPTurnRequest
RT-PLEASE-TURN-IND	(RtWaitRequest)

figure 5.16. le RT-TURN-PLEASE.

Pour demander le tour, il suffit d'appeler **RtPTurnRequest.** La figure 5.17. réalise la comparaison de ses arguments avec ceux de la norme.

ISO	ISODE
priorité	priority
-	sd
-	rti

figure 5.17. comparaison des arguments de RtPTurnRequest avec la norme.

L'événement d'indication est implémenté chez le correspondant éloigné par un retour de la valeur DONE de la part de RtWaitRequest (avec les données adéquates au sein du paramètre "rti").

6) RT-TURN-GIVE.

La figure 5.18. résume les routines intervenant dans le RT-TURN-GIVE.

PRIMITIVES ISO	ROUTINES ISODE
RT-TURN-GIVE-REQ	RtGTurnRequest
RT-TURN-GIVE-IND	(RtWaitRequest)

figure 5.18. le RT-TURN-GIVE.

Pour céder le tour de parole, il faut appeler **RtGTurnRequest.** Comme le montre la figure 5.19., la comparaison avec la norme est très simple.

ISO	ISODE
-	sd
-	rti

figure 5.19. comparaison des arguments de RtGTurnRequest avec la norme.

Sur l'hôte éloigné, RtWaitRequest permettra aussi de signaler l'arrivée d'un RT-TURN-GIVE-INDICATION en retournant la valeur DONE.

7) RT-P-ABORT.

Le fournisseur RTSE signale un problème à son utilisateur en retournant NOTOK comme résultat des routines de service.

5.2.3. LES STRUCTURES D'INFORMATIONS.

Nous passons ici en revue les différentes structures de données C associées aux routines du RTSE d'ISODE (figure 5.20.). Leurs paramètres sont données dans l'annexe 6.

1) Struct RtSAPstart

Cette structure, modifiée par RtInit, contient les paramètres d'une demande d'association.

2) Struct RtSAPconnect

Cette structure, modifiée par RtOpenRequest, contient les informations relatives à la réponse à une demande d'association.

3) Struct AcSAPrelease : voir le chapitre sur l'ACSE.

4) Structure RtSAPindication.

Cette structure décrit soit le problème rencontré par une routine qui a retourné la valeur NOTOK comme résultat, soit l'événement enregistré par RtWaitRequest.

ROUTINES	STRUCTURES ASSOCIEES
RtOpenRequest	RtSAPconnect (RtSAPindication)
RtInit	RtSAPstart (RtSAPindication)
RtOpenResponse	(RtSAPindication)
RtCloseRequest	AcSAPrelease (RtSAPindication)
RtCloseResponse	(RtSAPindication)
RtUAbortRequest	(RtSAPindication)
RtTransferRequest	(RtSAPindication)
RtWaitRequest	RtSAPindication
RtPTurnRequest	(RtSAPindication)
RtGTurnRequest	(RtSAPindication)

figure 5.20. les structures d'information et les routines associées; les parenthèses sont utilisées quand RtSAPindication n'est modifiée qu'en cas d'échec de la routine.

6. LE ROSE.

6.1. RAPPEL THEORIQUE.

6.1.1. FONCTION.

Dans certaines applications, la communication entre les AE est de nature interactive : une AE appelée **invocateur** ou **demandeur** commande l'exécution d'une opération à une autre AE appelée **réalisateur** ou **répondeur** qui, après avoir tenté de la réaliser, communiquera à la première AE le résultat de son travail (figure 6.1.).

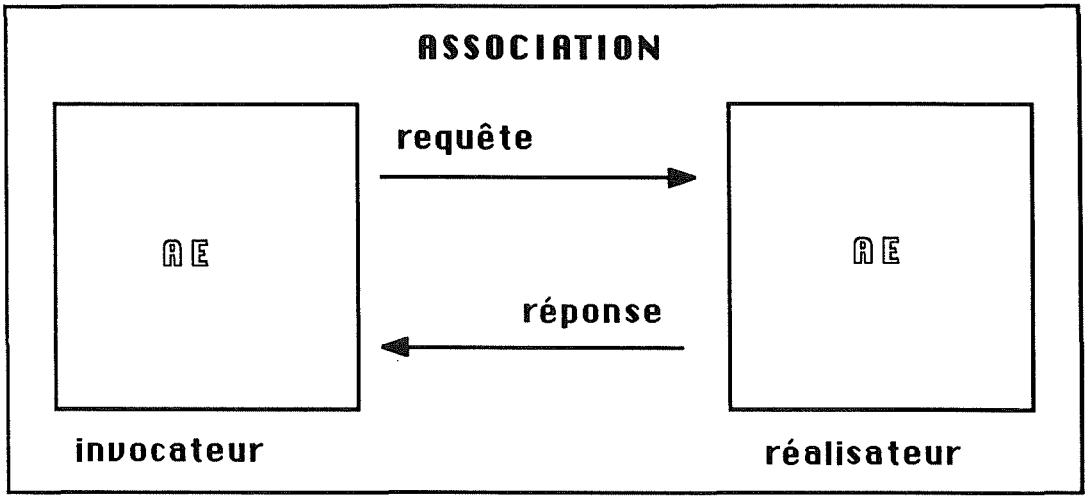

figure 6.1. nature interactive d'une communication.

L'élément de service des opérations lointaines ou **ROSE** (pour Remote Operation Service Element) permet ce comportement. Différentes classes d'opérations sont définies suivant qu'elles soient synchrones ou asynchrones d'une part, et suivant ce que le répondeur signalera au demandeur la fin de son travail d'autre part (figure 6.2.).

CLASSE	MODE	RAPPORT
1	synchrone	résultat ou erreur
2	asynchrone	résultat ou erreur
3	asynchrone	uniquement les erreurs
4	asynchrone	uniquement les résultats
5	asynchrone	rien

figure 6.2. les classes d'opérations.

Certaines opérations associées peuvent être regroupées en opérations liées formées d'une opération dite "mère" et d'une ou plusieurs opérations dites "filles". Les associations sont également classifiées suivant le comportement permis aux AE associées (figure 6.3.).

CLASSE	COMPORTEMENT
1	Seul l'initiateur de l'association invoque des opérations
2	Seul le répondeur à l'association invoque des opérations
3	Les deux invoquent des opérations

figure 6.3. classification des associations.

Les différents éléments de service sont repris à la figure 6.4.

PRIMITIVES	TYPE
RO-INVOKE	non-confirmé
RO-RESULT	non-confirmé
RO-ERROR	non-confirmé
RO-REJECT-U	non-confirmé
RO-REJECT-P	initié par le fournisseur

figure 6.4. éléments de service du ROSE.

Notons qu'aucun élément de service ne concerne l'établissement d'une association. L'utilisation du ROSE suppose un appel préalable à l'ACSE pour créer une association. Signalons enfin que le ROSE transmettra ses APDU soit grâce au RTSE soit en appelant directement les primitives du service présentation, tout cela dépendant du contexte d'application choisi lors de l'établissement de l'association.

6.1.2. ELEMENTS DE SERVICE.

Nous présentons rapidement ici le service du ROSE. Les paramètres des différents éléments de service sont regroupés dans l'annexe 7.

1) RO-INVOKE.

Le service **RO-INVOKE** permet à une AE de demander l'exécution d'une opération à une autre AE. C'est un service non confirmé à 2 temps.

2) RO-RESULT.

Le service **RO-RESULT** donne la possibilité au répondeur d'envoyer le résultat d'une opération réalisée avec succès. C'est un service non confirmé à 2 temps.

3) RO-ERROR.

Le **RO-ERROR** est utilisé par le répondeur pour signaler son échec lors d'une tentative d'effectuer une opération. C'est un service non-confirmé à 2 temps.

4) RO-REJECT-U.

Le service de **RO-REJECT-U** permet au répondeur d'indiquer son refus d'exécuter une opération. C'est un service non confirmé à 2 temps.

5) R0-REJECT-P.

Ici, le fournisseur ROSE signale à son utilisateur qu'il a rencontré un problème au sujet d'une opération.

Le service est défini dans (13) et (7).

6.1.3. LE PROTOCOLE.

Les différents APDU sont repris ici dans la figure 6.5.

APDU	FONCTION	CREATEUR
ROIV	invocation d'une opération	RO-INVOKE-REQ
RORS	retourner un résultat	RO-RESPONSE-REQ
ROER	retourner une erreur	RO-ERROR-REQ
RORJ	rejet d'une opération	RO-REJECT-U-REQ ou RO-REJECT-P-REQ

figure 6.5. les APDU du ROSE.

Ces APDU sont détaillés dans l'annexe 8.

Signalons que si le RTSE est inclus dans le contexte d'application, tous les APDU du ROSE sont transmis par le service RT-TRANSFER (figure 6.6.).

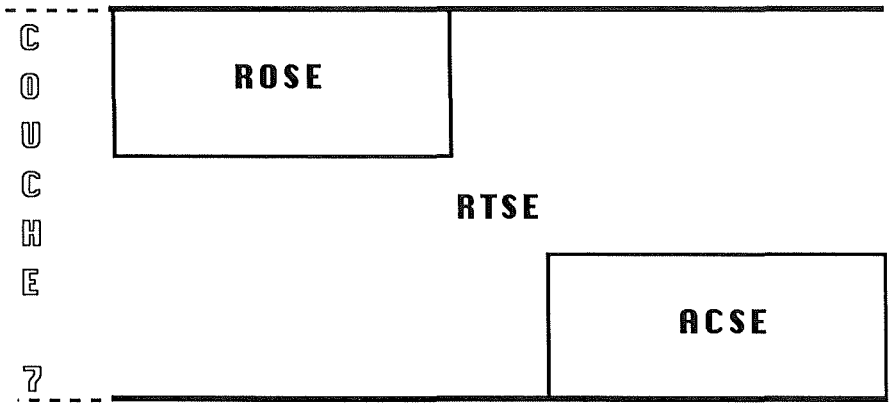

figure 6.6. le ROSE avec le RTSE.

Si le RTSE n'est pas utilisé, le ROSE transmettra ses APDU grâce au service P-DATA de la couche présentation (figure 6.7.).

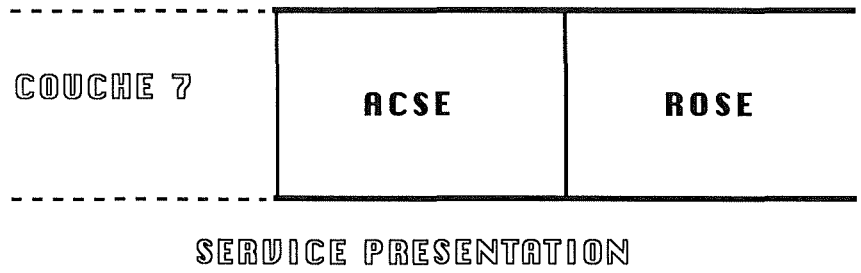

figure 6.7. le ROSE sans le RTSE.

Le protocole fait l'objet de la norme (14).

6.2. LE ROSE D'ISODE.

6.2.1. GENERALITES.

La librairie **librosap** implémente le service offert par le ROSE. Certaines applications demandant plus de liberté ou de fiabilité que d'autres, ISODE offre 3 disciplines de service différentes :
-le service de base correspondant à la vision ECMA du ROSE,
-le service avancé pour l'interprêtation CCITT X400,
-le service complet pour la vision CCITT MOTIS.

Parmi les petites différences les caractérisant, citons :
-dans le service de base, seul l'initiateur de l'association a le droit d'invoquer des opérations chez son correspondant,
-le service complet supporte la notion d'opération liée permettant d'associer une opération à une autre.

De manière générale, le service complet est moins restrictif que le service de base mais il nécessite une implémentation plus complexe de la part du fournisseur et de l'utilisateur. Pour éviter de charger le code objet de ces 3 disciplines, il est possible de donner au chargeur des informations utiles après l'établissement de l'association grâce à la routine **RoSetService(sd, bfunc, roi)**

int sd : descripteur de l'association,
IFP bfunc : service choisi,
struct RoSAPindication *roi : modifié en cas de problème.

La description de l'implémentation est réalisée dans (10).

6.2.2. L'IMPLEMENTATION DES PRIMITIVES DE SERVICE.

1) RO-INVOKE.

Les routines du RO-INVOKE sont résumées à la figure 6.8.

PRIMITIVES ISO	ROUTINES ISODE
RO-INVOKE-REQUEST	RoInvokeRequest
RO-INVOKE-INDICATION	(RoWaitRequest)

figure 6.8. le RO-INVOKE.

Tout comme la routine RtWaitRequest, **RoWaitRequest** sert à attendre des événements. Ses arguments sont :

int sd : descripteur de l'association,
secs : temps maximum d'attente,
struct RoSAPindication *roi : contient les info. relatives à l'événement survenu.

Cette routine retourne 3 valeurs possibles :

-NOTOK en cas d'échec,

-OK si un événement associé à une opération est arrivé, c'est à dire un RO-INVOKE-INDICATION, un RO-RESULT-INDICATION, un RO-ERROR-INDICATION ou un RO-U-REJECT-INDICATION.

-DONE si un autre événement est apparu (exemple : A-RELEASE-INDICATION).

Voyons maintenant la réalisation d'un RO-INVOKE par ISODE en supposant que le RTSE fasse partie du contexte d'application.

-Le serveur se met en attente en appelant RoWaitRequest.

-Le client veut faire exécuter une opération. Il appelle la routine RoInvokeRequest. Celle-ci va transmettre la demande en faisant appel à RtTransferRequest (RTSE). Cette demande envoyée, RoInvokeRequest appelle RoWaitRequest pour attendre le résultat de l'opération demandée.

-Sur le serveur, la demande du client arrive. RoWaitRequest se termine en retournant OK. Le serveur examine la requête décrite dans RoSAPindication. Suivant le résultat, il appellera RoErrorRequest (le résultat de l'opération est une erreur), RoResultRequest (l'opération est normalement réalisée) ou RoURejectRequest (l'opération est refusée).

-Ce résultat est transmis par la routine RtTransferRequest du RTSE.

-Le résultat arrive chez le client, provoquant la fin de RtWaitRequest, puis celle de RoInvokeRequest.

La figure 6.9. réalise la comparaison avec la norme.

ISO	ISODE
valeur d'opération	op
classe d'opération	class
arguments	args
identificateur d'invocation	invoke
identificateur lié	linked
-	sd
-	roi

figure 6.9. comparaison des paramètres de RoInvokeRequest avec la norme

2) RO-RESULT.

La figure 6.10. résume les routines du RO-RESULT.

PRIMITIVES ISO	ROUTINES ISODE
RO-RESULT-REQUEST	RoResultRequest
RO-RESULT-INDICATION	(RoWaitRequest)

figure 6.10. le RO-RESULT.

La figure 6.11. montre la comparaison avec la norme.

ISO	ISODE
valeur	-
resultat	result
identificateur d'invocation	invoke
priorité	priority
-	sd
-	op
-	roi

figure 6.11. comparaison des arguments de RoResultRequest avec la norme.

L'absence du paramètre "pattern" s'explique par le fait que la version du protocole sur laquelle ISODE 4.0. se base ne le définissait pas encore.

Le RO-RESULT-INDICATION se manifestera par un résultat adéquat à un appel à RoWaitIndication.

3) RO-ERROR.

La figure 6.12. rassemble les routines implémentant le RO-ERROR.

PRIMITIVES ISO	ROUTINES ISODE
RO-ERROR-REQUEST	RoErrorRequest
RO-ERROR-INDICATION	(RoWaitRequest)

figure 6.12. le RO-ERROR.

La figure 6.13. compare la routine avec la norme.

ISO	ISODE
valeur de l'erreur	error
paramètres de l'erreur	params
identificateur d'invocation	invoke
priorité	priority
-	sd
-	roi

figure 6.13. comparaison des arguments de RoErrorRequest avec la norme.

4) RO-REJECT-U.

Les routines implémentant le RO-REJECT-U sont dans la figure 6.14.

PRIMITIVES ISO	ROUTINES ISODE
RO-REJECT-U-REQUEST	RoURejectRequest
RO-REJECT-U-INDICATION	(RoWaitRequest)

figure 6.14. le RO-REJECT-U.

La figure 6.15. réalise la comparaison avec la norme.

ISO	ISODE
raison du rejet	reason
identificateur de requête	invoke
priorité	priority
-	sd
-	roi

figure 6.15. comparaison des arguments de RoURejectRequest avec la norme.

5) RO-REJECT-P.

Si, en s'occupant d'une opération, le fournisseur ROSE rencontre un problème, il le signale à son utilisateur en retournant la valeur NOTOK comme résultat de la routine qu'il exécutait.

6.2.3. LES STRUCTURES D'INFORMATIONS.

La seule structure d'infomation apparaissant dans les routines du ROSE est **RoSAPindication**. Elle contient soit la raison de l'échec éventuel d'une routine, soit la description de l'événement rencontré par RoWaitRequest. Elle est décrite en détails dans l'annexe 9.

7. LE SERVICE PRESENTATION.

7.1. RAPPEL THEORIQUE.

7.1.1. FONCTION.

Le service présentation est chargé de tous les problèmes d'ordre syntaxique pouvant apparaître lorsque des AE, utilisant des règles de représentation de données différentes, s'échangent des informations. Son rôle est semblable à celui d'un interprête. Ses fonctions principales :
-négociation de syntaxe de transfert,
-transformation de données de ou vers une syntaxe de transfert.

La couche présentation peut également assurer des fonctions de compression des données ou de chiffrement (18).

Il y a 3 formes de représentation à prendre en compte : la représentation des données à la source, la représentation des données à la destination, la représentation des données lors du transfert.

Exemple : des entiers peuvent être codés à la source sur 16 bits, sur 32 bits à la destination et sur 8 bits lors du transfert.

7.1.2. CONCEPTS IMPORTANTS.

1) SYNTAXE ABSTRAITE.

La **syntaxe abstraite** spécifie les données de la couche application ou des APCI (application-protocol-control-informations) par l'utilisation de règles de notation qui sont indépendantes de la technique d'encodage pour les représenter. Une syntaxe abstraite est identifiée sans ambiguité par un nom. L'ISO a développé et standardisé une syntaxe abstraite pour les systèmes de courrier électronique. Cette syntaxe s'appelle **ASN1** (pour "abstract syntax notation 1").

2) SYNTAXE CONCRETE.

La série de règles décrivant comment les données sont implémentées sur un système local s'appelle la **syntaxe concrète**.

3) SYNTAXE DE TRANSFERT.

La **syntaxe de transfert** se compose d'une série de règles de transformation des informations décrites par la syntaxe abstraite en bytes destinés à être transférés. C'est donc la syntaxe concrète utilisée dans le transfert entre systèmes ouverts. Elle est identifiée par un nom.

4) VALEUR DE DONNEE DE PRESENTATION.

Une **valeur de donnée de présentation** est une unité d'information spécifiée dans la syntaxe abstraite qui sera transférée par le service présentation. Exemple : un bit string, un set ou un boolean constituent chacun une valeur de donnée de présentation pour la syntaxe abstraite ASN1.

5) CONTEXTE DE PRESENTATION.

Un **contexte de présentation** est une association entre une syntaxe abstraite et une syntaxe de transfert. Pour un utilisateur de la couche 6, cela représente un environnement où les valeurs des données de présentation peuvent être transférées sans ambiguité. Plusieurs contextes peuvent être définis pour la même syntaxe abstraite, chacun en représentant alors une utilisation spécifique.

La négociation d'un contexte de présentation se réalise comme suit :

-L'AP initiateur informe l'entité présentation des syntaxes abstraites qu'il veut utiliser.

-L'entité présentation examine chaque syntaxe abstraite et choisit l'ensemble des syntaxes de transfert capables d'y être associées.

-L'entité paire choisit pour chaque syntaxe abstraite une seule syntaxe de transfert parmi celles proposées. Elle expédie son résultat sous forme d'une liste de contextes de présentation.

-L'entité présentation initiatrice voit les contextes qui seront utilisés. Elle informe l'utilisateur qu'il peut envoyer des données.

6) SERIE DE CONTEXTES DEFINIS.

Lors d'une communication, les données peuvent être de types différents nécessitant dès lors plusieurs syntaxes abstraites. Les entités de présentation manipuleront ainsi plusieurs syntaxes abstraites pour une communication particulière.

Une **série de contextes définis** regroupe les contextes de présentation définis de commun accord entre les 3 partis de la communication : le service présentation et deux utilisateurs. L'inclusion d'un contexte dans la liste implique que sa syntaxe abstraite soit acceptable pour les utilisateurs et que les entités de présentation coopérantes se soient mises d'accord sur une syntaxe de transfert pour ce contexte. On abrège souvent cette série de contextes par **DCS** (pour "defined context set").

7) CONTEXTE PAR DEFAUT.

Le **contexte par défaut** est un contexte de présentation toujours connu du fournisseur et des deux utilisateurs sur une connexion présentation donnée. C'est celui qui s'applique toujours au paramètre "données de l'utilisateur" de la primitive P-EXPEDITED-DATA ou qui est utilisé quand le DCS est vide.

8) UNITES FONCTIONNELLES.

Un regroupement logique de services est appelé **unité fonctionnelle**. Celles existants au niveau présentation sont reprises dans la figure 7.1.

UNITES FONCTIONNELLES	FONCTION	COMMENTAIRES
noyau	transfert d'informations	toujours présente
gestion de contexte	modification du DCS	optionnelle
restauration de contexte	restauration d'anciens DCS	optionnelle sélectionnable si la gestion de contexte l'est aussi

figure 7.1. les unités fonctionnelles.

9) EXEMPLE.

Considérons l'analogie suivante : le capitaine d'un vaisseau anglais décide d'envoyer un message de salutation à un collègue égyptien. Il le rédige à la main sur un papier qu'il passe ensuite à son officier-radio.

Celui-ci le transmet au bateau égyptien en utilisant l'alphabet morse. Or, sur ce bateau, le capitaine ne parle ni ne lit l'anglais. Dès lors, son officier-radio va traduire ce message en arabe et le transmettra sous cette forme.

Sur l'émetteur, la syntaxe concrète est une écriture manuelle sur un papier. La syntaxe abstraite associée serait : une séquence de caractères issus de l'alphabet latin. La syntaxe concrète du destinataire correspond à l'écriture arabe. Le code morse constitue la syntaxe de transfert.

7.1.3. LES ELEMENTS DE SERVICE (figure 7.2.).

SERVICE	TYPE	FONCTION
P-CONNECT	confirmé	établir une connexion
P-RELEASE	confirmé	abandon de connexion
P-U-ABORT	non confirmé	abandon brutal de connexion
P-P-ABORT	initié par le fournisseur	abandon par le fournisseur
P-ALTER-CONTEXT	confirmé	changer un liste de contextes
P-DATA	non confirmé	envoyer des données
P-TYPED-DATA	non confirmé	envoyer des données typées
P-EXPEDITED-DATA	non confirmé	envoyer des données exprès
P-CAPABILITY-DATA	non confirmé	envoyer des données de capacité
P-TOKEN-GIVE	non confirmé	cession de jeton

figure 7.2. les éléments de service.

SERVICE	TYPE	FONCTION
P-TOKEN-PLEASE	non confirmé	demande de jetons
P-CONTROL-GIVE	non confirmé	cession de tous les jetons
P-SYNC-MINOR	confirmé	pose de point de synchronisation mineure
P-SYNC-MAJOR	confirmé	pose de point de synchronisation majeure
P-RESYNCHRONIZE	confirmé	resynchronisation
P-U-EXCEPTION-REPORT	non confirmé	rapport d'exception de l'utilisateur
P-P-EXCEPTION-REPORT	initié par le fournisseur	rapport d'exception du fournisseur
P-ACTIVITY-START	non confirmé	début d'activité
P-ACTIVITY-RESUME	non confirmé	reprise d'activité
P-ACTIVITY-END	non confirmé	fin d'activité
P-ACTIVITY-INTERRUPT	confirmé	interruption d'activité
P-ACTIVITY-DISCARD	confirmé	abandon d'activité

figure 7.2bis. les éléments de service (suite).

La plupart des éléments de service concernant des activités spécifiques du service session seront décrits dans le chapitre suivant. Nous n'abordons ici que les éléments spécifiques à la couche présentation. Leurs arguments sont rassemblés dans l'annexe 10.

1) P-CONNECT.

Le service **P-CONNECT** permet d'établir une connexion présentation. C'est un service à 4 temps. Son intérêt est notamment de négocier un ou plusieurs contextes de présentation (ceux du DCS et le contexte par défaut). Ces paramètres sont optionnels. Ainsi, si le DCS est vide, le contexte par défaut sera utilisé. Si celui-ci est absent, le fournisseur présentation fera la supposition qu'il existe déjà un accord préalable sur sa définition. Notons enfin que le répondeur ne peut pas accepter une connexion présentation s'il refuse le contexte par défaut proposé.

2) P-U-ABORT.

Le **P-U-ABORT** a pour effet de forcer l'abandon d'une connexion de présentation de la part d'un utilisateur. C'est un service non confirmé.

3) P-P-ABORT.

Le **P-P-ABORT** informe l'utilisateur d'un abandon de connexion en raison d'un problème interne du fournisseur présentation.

4) P-ALTER-CONTEXT.

Le **P-ALTER-CONTEXT**, seulement disponible si l'unité fonctionnelle de gestion de contexte est sélectionnée, permet d'additionner et de supprimer des contextes de présentation. C'est un service confirmé. Si ce service a pour résultat un DCS vide, c'est le contexte de présentation qui sera utilisé.

5) P-RELEASE.

Le **P-RELEASE** a pour fonction de négocier l'abandon d'une connexion. C'est un service confirmé.

Le service présentation est décrit dans (15) et (5).

7.1.5. LE PROTOCOLE (figure 7.3.).

PPDU	FONCTION	CREATEUR
AC	changer des contextes	P-ALTER-CONTEXT-REQ
ACA	accusé de réception d'un changement de contexte	P-ALTER-CONTEXT-RESP
ARP	abandon anormal du fournisseur	P-P-ABORT-IND
ARU	abandon anormal de l'utilisateur	P-U-ABORT-REQ
CP	demande de connexion	P-CONNECT-REQ
CPA	acceptation de connexion	P-CONNECT-RESP
CPR	refus de connexion	P-CONNECT-RESP
RS	resynchronisation	P-RESYNCHRO.-REQ
RSA	accusé de réception d'un RS	P-RESYNCHRO.-RESP
TC	données de capacité	P-CAPAB.-DATA-REQ
TCC	accusé de réception d'un TC	P-CAPAB.-DATA-RESP
TD	données normales	P-DATA-REQ
TE	données exprès	P-EXPED.-DATA-REQ
TTD	données typées	P-TYPED-DATA-REQ

figure 7.3. le protocole.

Les paramètres de ces PPDU sont dans l'annexe 11.

Certains services, notament ceux de gestion d'activité, ne mènent à aucun PPDU. Pour ceux-ci, il existe un "mapping" direct entre les couches 6 et 5 comme le montre la figure 7.4.

PRIMITIVES DE PRESENTATION	PRIMITIVES DE SESSION
P-TOKEN-GIVE-REQUEST	S-TOKEN-GIVE-REQUEST
P-SYNC-MINOR-REQUEST	S-SYNC-MINOR-REQUEST
P-U-EXCEPTION-REPORT-REQ	S-U-EXCEPTION-REPORT
...	...

figure 7.4. mapping entre les couches 6 et 5.

Le protocole est défini dans (16).

7.2. LE SERVICE PRESENTATION D'ISODE.

7.2.1. GENERALITES.

La librairie **libpsap2(3n)** implémente le service présentation. Seule l'unité fonctionnelle noyau est présente. ISODE permet d'utiliser les exigences de session, la négociation de contextes lors de l'établissement de connexion, le transfert de données mais pas les fonctions de modification de contextes disponibles dans les deux autres unités fonctionnelles.

L'interface est normalement synchrone mais on peut en choisir une asynchrone une fois la connexion établie. Toutes les routines retournent soit la valeur OK (réussite) soit la valeur NOTOK (échec). L'implémentation est décrite dans (17).

Il n'y a pas dans ISODE de réelle négociation de contexte. En effet, ISODE ne supporte qu'une syntaxe abstraite (ASN1) et qu'une syntaxe de transfert, celle décrite dans "Information Processing. Open System Interconnection. Specification of basis encoding rules for ASN1. 1987. International Standard 8825". Ainsi, chaque donnée aura toujours trois composantes : un type,
une longueur,
une valeur.

7.2.2. L'IMPLEMENTATION DES PRIMITIVES DE SERVICE.

1) P-CONNECT.

Les routines du P-CONNECT sont dans la figure 7.5. Leur enchaînement est le même que pour le A-ASSOCIATE. Les figures 7.6. et 7.7. comparent la norme et les routines d'ISODE.

PRIMITIVES ISO	ROUTINES ISODE
P-CONNECT-REQ	PConnRequest
P-CONNECT-IND	(PInit)
P-CONNECT-RESP	PConnResponse
P-CONNECT-CONF	(PConnRequest)

figure 7.5. le P-CONNECT.

ISO	ISODE
adresse appelante	calling
adresse appelée	called
liste de déf. de contextes	ctxlist
nom du contexte par défaut	defctxname
qualité de service	qos
exigences de présentation	prequirements
mode	-
exigences de session	srequirements
point de synchro. initial	isn
initialisation des jetons	settings
identif. de connexion session	ref
données de l'utilisateur	data ndata
-	pc
-	pi

figure 7.6. comparaison des arguments de PConnRequest avec la norme.

"Data" et "ndata" contiennent respectivement les données et leur longueur. "Pc" et "pi" pointent vers soit une structure PSAPconnect soit une structure PSAPindication.

Les arguments de PInit sont :

int	vecp	: longueur du vecteur argument,
char	**vec	: vecteur argument,
struct PSPAstart	*ps	: modifié si tout va bien,
struct PSAPindication	*pi	: modifié en cas d'échec.

ISO	ISODE
adresse répondante	responding
liste résultat des contextes	ctxlist
résultat du contexte par défaut	defctxresult
qualité de service	-
exigences de présentation	prequirements
exigences de session	srequirements
point de synchro. initial	isn
initialisation des jetons	settings
identif. de connexion session	ref
données de l'utilisateur	data ndata
résultat	status
-	sd
-	pi

figure 7.7. comparaison des arguments de PConnResponse avec la norme.

"Sd" est le descripteur de la connexion.

2) TRANSFERT DE DONNEES.

Les autres éléments de service (sauf le P-ALTER-CONTEXT) sont aussi implémentés par des routines. Celles-ci sont identiques à leurs collègues du service session qui seront décrites dans le chapitre suivant. Signalons cependant quelques différences :

-le dernier paramètre de chaque routine est un pointeur vers une structure PSAPindication au lieu de SSAPindication,

-les données de l'utilisateur sont spécifiées comme un tableau d'éléments de présentation et non un tableau de caractères.

7.2.3. LES STRUCTURES D'INFORMATIONS.

ROUTINES	STRUCTURES ASSOCIEES
PInit	PSAPstart (PSAPindication)
PConnResponse	(PSAPindication)
PConnRequest	PSAPconnect (PSAPindication)

figure 7.8. les structures d'informations; les parenthèses indiquent les structures modifiées en cas d'échec de leur routine.

Les paramètres de ces structures sont dans l'annexe 12.

1) Struct PSAPstart

Cette structure contient toutes les informations sur une demande de connexion reçue par le service.

2) Struct PSAPconnect

Cette structure comprend les informations relatives à la réponse à une demande de connexion.

3) Struct PSAPindication

Cette dernière structure contient un indicateur et une union de structures comme le résume la figure 7.9.

STRUCTURE	FONCTION
PSAPdata	contenir des données reçues
PSAPtoken	gestion des jetons
PSAPsync	gestion des points de synchronisation
PSAPactivity	gestion des activités
PSAPreport	gestion des rapports d'exception
PSAPfinish	abandon négocié de connexion
PSAPabort	abandon brutal de connexion

figure 7.9. structures contenues au sein de PSAPindication.

8. LE SERVICE SESSION.

8.1. RAPPEL THEORIQUE.

8.1.1. FONCTION.

La couche session permet d'établir une relation ou connexion de session entre deux applications souhaitant coopérer (au travers de deux entités de présentation), d'en organiser le dialogue, et de le synchroniser. Elle peut permettre un échange duplex, semi-duplex ou simplex. Elle gère aussi les modalités de reprise en cas d'incident.

8.1.2. CONCEPTS IMPORTANTS DU SERVICE SESSION.

1) LE MECANISME DU JETON.

Le **jeton** est un attribut qui restreint l'utilisation de certains services à celui qui le possède. On peut donner un exemple sportif. Dans un 4*100 mètres, un athlète doit posséder le bâton-témoin pour pouvoir courir. Quand il a parcouru sa distance, il le passe à un coéquipier et c'est au tour de ce dernier de courir. Ce bâton-témoin est l'analogie sportive d'un jeton. Au niveau session, il existe 4 jetons :
-jeton de données,
-jeton de synchronisation mineure,
-jeton d'activité et de synchronisation majeure,
-jeton de terminaison.

2) LES POINTS DE SYNCHRONISATION.

Un **point de synchronisation** constitue un point de repère identifié par un numéro. Il existe 2 types de points de synchronisation.
-type **majeur** : un tel point permet de marquer une coupure dans l'expédition. Une fois posé, l'expéditeur en attend la confirmation avant de continuer d'envoyer des données. Deux points de synchronisation majeure encadrent ce qu'on appelle une **unité de dialogue** (figure 8.1.).
-type **mineur :** un tel point constitue un repère au sein d'une unité de dialogue. Il n'est pas obligatoirement confirmé immédiatement.

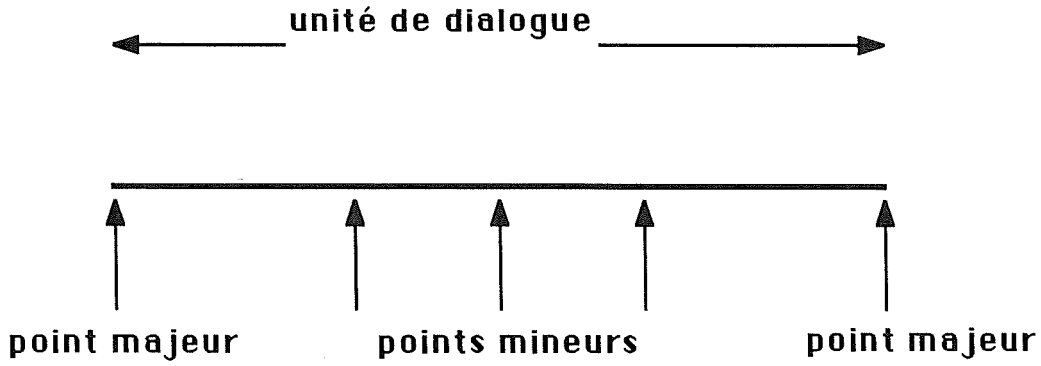

figure 8.1. unité de dialogue et points de synchronisation.

Le positionnement de ces points dépend de l'utilisateur mais leur gestion et leur identification est du ressort du service session. Notons que seules les données normales ou typées peuvent être synchronisées.

3) LA RESYNCHRONISATION.

En cas de problème, le mécanisme de **resynchronisation** permet un retour en arrière à un endroit convenu et bien connu des interlocuteurs. Cet endroit est un point de synchronisation.

4) LE CONCEPT D'ACTIVITE.

Le concept d'**activité** (figure 8.2.) permet le regroupement d'opérations diverses en unités logiques de travail. Une telle entité logique est gérable au niveau session. Ainsi, une activité peut être interrompue, reprise, abandonnée. Elle se compose d'une ou plusieurs unités de dialogue.

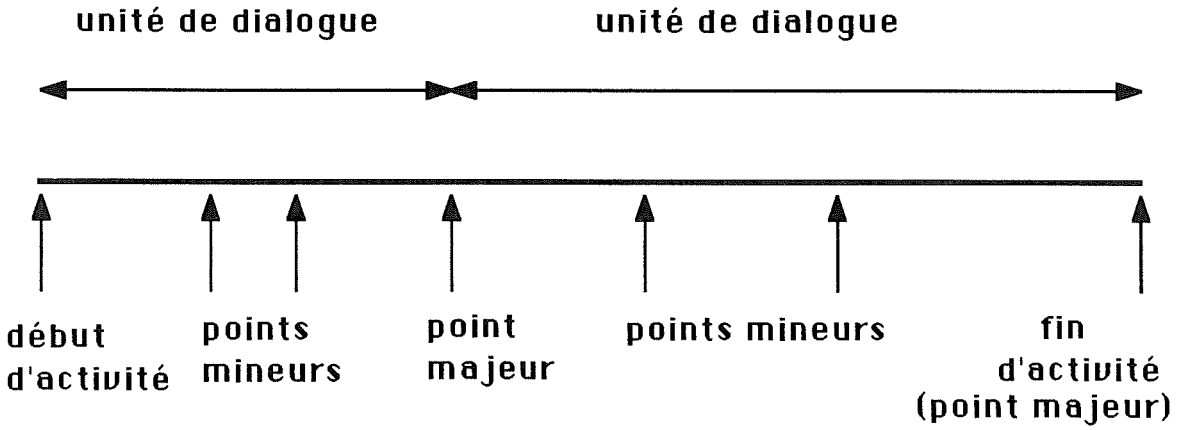

figure 8.2. activité et points de synchronisation.

5) UNITES FONCTIONNELLES.

Les fonctions élémentaires du service session sont regroupées de façon logique en **unités fonctionelles.** Leur utilisation est négociable sauf dans le cas du **noyau.** Le noyau est une unité fonctionnelle regroupant les services obligatoires de la couche session. Il est toujours présent. La figure 8.3. reprend les unités fonctionnelles existantes.

UNITES FONCTIONNELLES	SERVICES
noyau	connexion session transfert de données normales terminaison normale coupure par l'utilisateur ou le fournisseur
terminaison négociée	terminaison normale cession et demande de jetons
semi-duplex	cession et demande de jetons
transmission duplex	aucun service additionnel
transfert de données exprès	transfert de données exprès
transfert de données typées	transfert de données typées
échange d'informations de capacité	échange d'informations de capacité
synchronisation mineure	cession et demande de jetons pose de points mineurs
synchronisation majeure	cession et demande de jetons pose de points majeurs

figure 8.3. les unités fonctionnelles.

UNITES FONCTIONNELLES	SERVICES
resynchronisation	resynchronisation
exception	rapport d'anomalies par le fournisseur ou l'utilisateur
gestion d'activités	lancement, reprise, abandon interruption, terminaison d'activité cession et demande de jetons cession du contrôle

figure 8.3.bis les unités fonctionnelles (suite).

6) SOUS-ENSEMBLE.

SOUS-ENSEMBLE	UNITES FONCTIONNELLES
BCS (Basic Combined Subset)	noyau semi-duplex ou duplex
BAS (Basic Activity Subset)	noyau semi-duplex transfert de données typées échange d'informations de capacité signalisation d'anomalies gestion d'activité
BSS (Basic Synchronized Subset)	noyau terminaison négociée semi-duplex transfert de données typées synchronisation mineure synchronisation majeure resynchronisation

figure 8.4. les sous-ensembles.

On appelle **sous-ensemble** la combinaison du noyau et des unités fonctionnelles utilisées au cours d'une connexion session. La norme ISO-CCITT en distingue 3 où ont été regroupées celles qui allaient bien ensemble (figure 8.4.).

Notons que ces 3 sous-ensembles sont donnés à titre indicatif. En effet, les règles de négociation s'appliquent aux unités fonctionnelles et non aux sous-ensembles. Signalons cependant que si l'unité de données de capacité est inclue dans un sous-ensemble, l'unité de gestion des activités doit aussi s'y trouver. Par ailleurs, la présence de l'unité d'exception implique celle de l'unité de semi-duplex.

7) EXEMPLE.

Donnons un petit exemple pour situer ces concepts. Imaginons le transfert du contenu d'un livre. Un livre est un ensemble de chapitres, chacun constitué d'une suite de pages. Le transfert de ce livre constituera une activité. Pour des raisons de fiabilité, un point de synchronisation mineur sera posé après chaque page et un point de synchronisation majeure après chaque chapitre. Un chapitre constituera une unité de dialogue.

8.1.3. ELEMENTS DE SERVICE (figure 8.5.).

SERVICE	TYPE	FONCTION
S-CONNECT	confirmé	création d'une connexion
S-DATA	non confirmé	envoi de données normales
S-EXPEDITED-DATA	non confirmé	envoi de données exprès
S-TYPED-DATA	non confirmé	envoi de données typées
S-CAPABILITY-DATA	confirmé	envoi de données de capacité
S-TOKEN-GIVE	non confirmé	cession de jetons
S-TOKEN-PLEASE	non confirmé	demande de jetons

figure 8.5. éléments de service.

SERVICE	TYPE	FONCTION
S-CONTROL-GIVE	non confirmé	cession de tous les jetons
S-SYNC-MINOR	confirmé	pose de points mineurs
S-SYNC-MAJOR	confirmé	pose de points majeurs
S-RESYNCHRONIZE	confirmé	resynchronisation
S-P-EXCEPTION-REPORT	fourniseur	rapport d'exception par le fournisseur
S-U-EXCEPTION-REPORT	non confirmé	rapport d'exception par l'utilisateur
S-ACTIVITY-START	non confirmé	début d'activité
S-ACTIVITY-RESUME	non confirmé	reprise d'activité
S-ACTIVITY-INTERRUPT	confirmé	interruption d'activité
S-ACTIVITY-DISCARD	confirmé	suppression d'activité
S-ACTIVITY-END	confirmé	fin d'activité
S-RELEASE	confirmé	fin de connexion
S-U-ABORT	non confirmé	abandon brutal
S-P-ABORT	fournisseur	abandon brutal

figure 8.5.bis. éléments de service (suite).

Leurs arguments sont réunis dans l'annexe 13. Nous allons dire quelques mots sur certains de ces éléments de service.

1) TRANSFERT DE DONNEES.

Le service le plus simple est celui de données normales. Le service de données exprès permet d'envoyer rapidement des données sur une connexion session et cela sans posséder de jeton et sans se soucier des contraintes du contrôle de flux. Le service de données typées est utilisable indépendamment

de la disponibilité ou de la possession du jeton des données. Le service de données de capacité permet d'échanger des données de manière confirmée en dehors de toute activité alors que l'unité fonctionnelle de gestion des activités est sélectionnée.

2) S-CONTROL-GIVE.

Ce service qui permetde céder tous les jetons en une fois n'est disponible que si l'unité fonctionnelle de gestion d'activité a été sélectionnée mais qu'aucune activité n'est en cours.

3) POINTS DE SYNCHRONISATION.

Bien qu'étant un service confirmé, il n'est pas nécessaire de confirmer chaque point de synchronisation mineur. D'ailleurs, le nombre de points mineurs non confirmés n'est pas limité par le fournisseur session. Cependant, la confirmation d'un point confirme tous les points précédents. Par contre, un point de synchronisation majeure doit toujours être confirmé.

Le S-RESYNCHRONIZE permet de rétablir une communication dans un état bien défini et accepté.

4) RAPPORTS D'EXCEPTIONS.

Le S-P-EXCEPTION permet au fournisseur d'avertir les utilisateurs qu'une situation inattendue non couverte par les autres services vient d'arriver.

Le rôle du S-U-EXCEPTION-REPORT est de permettre à l'utilisateur de signaler à son correspondant éloigné une condition exceptionnelle.

Suite à un tel événement, une tentative de "réparation" pourra éventuellement être envisagée sans pour autant détruire la connexion. Une solution possible sera par exemple d'utiliser la resynchronisation ou d'abandonner l'activité posant problème. Tout cela dépendant du problème rencontré et de l'implémentation réalisée.

5) ACTIVITES.

Le S-ACTIVITY-START crée une activité. On ne peut utiliser ce service si une autre activité est déjà en cours. Un de ses effets est que le numéro que prendra le prochain point de synchronisation sera égal à 1.

Le S-ACTIVITY-RESUME réveille une activité interrompue.

Le S-ACTIVITY-INTERRUPT stoppe provisoirement l'exécution d'une activité.

Le S-ACTIVITY-DISCARD supprime une activité en cours.

Le S-ACTIVITY-END termine l'exécution normale d'une activité. Un de ses effets est la pose implicite d'un point de synchronisation majeure.

6) ABANDON BRUTAL.

Le service S-U-ABORT donne la possibilité à un utilisateur d'abandonner immédiatement une connexion session. La fonction du S-P-ABORT est d'abandonner immédiatement une connexion à cause d'un problème interne au fournisseur. Ces deux services peuvent provoquer une perte d'informations.

Le service session est présenté dans (19) et (5).

8.1.4. LE PROTOCOLE .

Les SPDU sont réunis dans la figure 8.6., leur paramètres sont énumérés dans l'annexe 14.

SPDU	FONCTION	CREATEUR
CONNECT	établir une connexion	S-CONNECT-REQ
ACCEPT	accepter une connexion	S-CONNECT-RESP
REFUSE	refuser une connexion	S-CONNECT-RESP
DATA TRANSFER	transfert normal	S-DATA-REQ
EXPEDITED DATA	transfert exprès	S-EXPDED.-DATA-REQ
TYPED DATA	transfert de données typées	S-TYPED-DATA-REQ
CAPABILITY DATA	transfert de données de capacité	S-CAPAB.-DATA-REQ
CAPABILITY DATA ACK	accusé de réception	S-CAPAB.-DATA-RESP
GIVE TOKENS	cession de jetons	S-TOKEN-GIVE-REQ
PLEASE TOKENS	demande de jetons	S-TOKEN-PLEASE-REQ

figure 8.6. le protocole.

SPDU	FONCTION	CREATEUR
GIVE TOKENS CONFIRM	cession du contrôle	S-CONTROL-GIVE-REQ
MINOR SYNC POINT	pose de point mineur	S-SYNC-MINOR-REQ
MINOR SYNC ACK	confirmation d'un point mineur	S-SYNC-MINOR-RESP
MAJOR SYNC POINT	pose de point majeur	S-SYNC-MAJOR-REQ
MAJOR SYNC ACK	confirmation d'un point majeur	S-SYNC-MAJOR-RESP
RESYNCHRONIZE	resynchronisation	S-SYNCHRONIZE-REQ
RESYNCHRONIZE ACK	accusé de réception	S-SYNCHRONIZE-RESP
EXCEPTION REPORT	rapport d'exception du fournisseur	S-P-EXCEP.-REP.-IND
EXCEPTION DATA	rapport d'exception	S-U-EXCEP.-REP.-IND
ACTIVITY START	début d'activité	S-ACTIVITY-START-REQ
ACTIVITY RESUME	reprise d'activité	S-ACTIVITY-RES.-REQ
ACTIVITY INTERRUPT	interruption d'activité	S-ACTIVITY-INTER.-REQ
ACTIVITY INTERR. ACK	accusé de réception	S-ACTIVITY-INTER.-RESP
ACTIVITY DISCARD	suppression d'activité	S-ACTIVITY-DISC.-REQ
ACTIVITY DISC. ACK	accusé de réception	S-ACTIVITY-DISC.-RESP
ACTIVITY END	fin d'activité	S-ACTIVITY-END-REQ
ACTIVITY END ACK	accusé de réception	S-ACTIVITY-END-RESP
FINISH	demande d'abandon	S-RELEASE-REQ
DISCONNECT	acceptation d'abandon	S-RELEASE-RESP

figure 8.6.bis. le protocole (suite).

SPDU	FONCTION	CREATEUR
NOT FINISHED	refus d'abandon	S-RELEASE-RESP
ABORT	abandon brutal	S-U-ABORT-REQ S-P-ABORT-IND

figure 8.6.tris. le protocole (suite et fin).

Vu leur nombre important, nous renonçons à passer en revue les paramètres de chacun des SPDU. Nous invitons le lecteur intéressé à consulter la norme X225. Cependant, certains SPDU, moins classiques n'ont pas été repris à la figure 8.6. Nous nous proposons ici d'en dire quelques mots.

1) OVERFLOW ACCEPT.

Il est possible qu'un CONNECT SPDU soit trop petit pour contenir toutes les données de l'utilisateur. Un de ses paramètres permet de signaler une telle situation. Recevant un CONNECT SPDU incomplet, un récepteur demandera la suite des données de l'utilisateur en envoyant un **OVERFLOW ACCEPT** SPDU.

2) CONNECT DATA OVERFLOW.

Un **CONNECT DATA OVERFLOW** SPDU contient les données de l'utilisateur qui n'ont pas trouvé place dans un CONNECT SPDU. Un tel SPDU est envoyé après réception d'un OVERFLOW ACCEPT SPDU. Le dernier CONNECT DATA OVERFLOW SPDU contient un paramètre l'identifiant comme tel. Après l'avoir réceptionné, l'entité session réceptrice pourra et seulement alors avertir son utilisateur en produisant un S-CONNECT-INDICATION.

3) ABORT ACCEPT.

Recevant un ABORT SPDU, un récepteur peur demander l'abandon ou la conservation de la connexion transport en envoyant un **ABORT ACCEPT** SPDU.

4) GIVE TOKENS ACK.

Un **GIVE TOKENS ACK** SPDU est envoyé comme accusé de réception d'un GIVE TOKENS CONFIRM SPDU mais seulement si aucune activité n'est en progrès.

5) PREPARE.

Le **PREPARE** SPDU notifie l'arrivée imminente de certains SPDU. Il est utilisable si le flot exprès de transport est disponible. Voici un exemple d'utilisation : un utilisateur se voit obliger d'effectuer une demande de resynchronisation. Il va d'abord émettre un PREPARE SPDU sur le service de transport exprès puis un RESYNCHRONIZE SPDU sur le service transport de données normales. Recevant le PREPARE SPDU, son homologue détruira tout SPDU reçu par la suite (sauf un ABORT SPDU) jusqu'à réception du RESYNCHRONIZE SPDU.

Le protocole fait l'objet de (20).

8.2. LE SERVICE SESSION D'ISODE.

8.2.1. GENERALITES.

La librairie **libssap(3n)** implémente le service session, en l'occurrence les sous-ensembles BAS, BCS, BSS complétés de l'unité fonctionnelle de transfert de données exprès. L'interface est normalement synchrone mais peut devenir asynchrone. Les éléments de service sont implémentés sous forme de routines retournant soit la valeur OK soit la valeur NOTOK. L'implémentation est présentée dans (17).

8.2.2. L'IMPLEMENTATION DES PRIMITIVES DE SERVICE.

1) S-CONNECT.

Le S-CONNECT d'ISODE est résumé à la figure 8.7.

PRIMITIVES ISO	ROUTINES ISODE
S-CONNECT-REQ	SConnRequest
S-CONNECT-IND	(SInit)
S-CONNECT-RESP	SConnResponse
S-CONNECT-CONF	(SConnRequest)

figure 8.7. le S-CONNECT.

Le mécanisme d'établissement de la connexion session sera explicité au chapitre 10 dans l'exemple 1. La plupart des routines intervenant sont semblables à leurs homologues des couches supérieures. Ainsi, les arguments de **SInit** sont équivalents à ceux de PInit. Pour les autres routines, les figures 8.8. et 8.9 contiennent leurs arguments comparés à ceux de la norme.

ISO	ISODE
identifiant de connexion session	ref
adresse appelante	calling
adresse appelée	called
qualité de service	qos
exigences de session	requirements
numéro du point de synchro. initial	ins
initialisation des jetons	settings
données de l'utilisateur	data cc
-	sc
-	si

figure 8.8. comparaison des arguments de SConnRequest avec la norme.

"Data" contient les données et "cc" en donne la longueur. "Sc" pointe vers une structure SSAPconnect utile si tout va bien et "si" pointe vers une structure SSAPindication utilisée en cas d'échec.

ISO	ISODE
-	sd
identifiant de connexion	ref
adresse répondante	called
résultat	result
qualité de service	-
exigences de session	requirements
numéro du point de synchronisation initial	isn
initialisation des jetons	settings
données de l'utilisateur	data cc
-	si

figure 8.9. comparaison des arguments de SConnResponse avec la norme.

2) TRANSFERT DE DONNEES.

l'implémentation de tous les transferts de données d'ISODE est résumée par la figure 8.10.

PRIMITIVES ISO	ROUTINES ISODE
S-DATA-REQ	SDataRequest SWriteRequest
S-EXPEDITED-DATA-REQ	SExpdRequest
S-TYPED-DATA-REQ	STypedRequest SWriteRequest
S-CAPABILITY-DATA-REQ	SCapdRequest
S-CAPABILITY-DATA-RESP	SCapdResponse
S-CAPABILITY-DATA-CONF	(SReadRequest)
(toutes les indications)	(SReadRequest)

figure 8.10. le transfert des données.

Toutes les routines de la figure 8.10. (sauf SWriteRequest et SReadRequest) ont les mêmes arguments, à savoir :

int sd : descripteur de session,
char *data : données a transférer,
int cc : longueur de ces données,
struct SSAPindication *si : modifié si l'appel échoue.

SWriteRequest est similaire en nature à SDataRequest et STypedRequest mais utilise des paramètres différents :

int sd : descripteur de session,
typed : indication de l'utilisation ou non des données typées,
struct udvec *uv : données décrites sous forme d'un tableau d'éléments,
int cc : nombre d'éléments du tableau,
struct SSAPindication *si : modifié en cas d'échec.

Pour l'attente de données, un récepteur fait appel à **SReadRequest(sd, sx, secs, si)**

int sd : descripteur de session,
struct SSAPdata sx : les données à recevoir, de même que leur type (normal, typé, ...),
int secs : temps d'attente maximum,
struct SSAPindication *si : modifié si aucune donnée n'est arrivée.

SReadRequest retourne 3 valeurs différentes possibles :
-OK si des données sont arrivées dans le paramètre "sx",
-NOTOK en cas d'échec et le paramètre "si" décrit le problème rencontré,
-DONE si autre chose que des données est arrivé, à savoir un événement en rapport avec :
-la gestion des jetons,
-la gestion de la synchronisation,
-la gestion des activités,
-les rapports d'exceptions,
-un abandon de connexion.
Dans tous les cas, "si" décrit l'événement survenu.

3) GESTION DES ACTIVITES .

La figure 8.11. résume toutes les routines impliquées dans la gestion des activités.

PRIMITIVES ISO	ROUTINES ISODE
S-ACTIVITY-START-REQ	SActStartRequest
S-ACTIVITY-RESUME-REQ	SActResumeRequest
S-ACTIVITY-INTERRUPT-REQ	SActIntrRequest
S-ACTIVITY-INTERRUPT-RESP	SActIntrResponse
S-ACTIVITY-DISCARD-REQ	SActDiscRequest
S-ACTIVITY-DISCARD-RESP	SActDiscResponse
S-ACTIVITY-END-REQ	SActEndRequest
S-ACTIVITY-END-RESP	SActEndResponse
(toutes les indications)	(SReadRequest)
(toutes les confirmations)	(SReadRequest)

figure 8.11. la gestion des activités.

Notons que, contrairement à la norme, les routines de suppression et d'interruption d'activité n'ont pas de paramètre "données de l'utilisateur".

Voyons pour l'exemple comment se déroule le S-ACTIVITY-END en supposant que tout fonctionne bien.

-Désirant terminer une activité, un client appelle la routine SActEndRequest synonyme de S-ACTIVITY-END-REQUEST.

-Cette routine va confier le transfert de la requête à une autre routine : SMajSyncRequestAux dont une des fonctions est de transmettre une demande de pose de point de synchronisation majeure (car toute fin d'activité implique la pose d'un tel point). La demande de fin d'activité sera transmise dans le paramètre des données de l'utilisateur de SManSyncRequestAux.

-Ces données émises, SMajSyncRequestAux se termine, suivie de SActEndRequest. Maintenant, le client doit attendre la réponse du serveur. Il

appelle dès lors SReadRequest.

-Le serveur était en attente (il avait appelé SReadRequest). Recevant la requête du client, SReadRequest se termine en retournant la valeur DONE.

-Le serveur examine les informations retournées dans la structure SSAPindication. Il peut ensuite répondre à la demande de fin d'activité grâce à la routine SActEndResponse.

-Cette routine appelle SMajSyncResponseAux pour transmettre la réponse (dans les données de l'utilisateur) d'une part et pour confirmer le point de synchronisation majeure d'autre part.

-Ces informations émises, SMajSyncResponseAux et SActEndResponse se terminent.

-Chez le client, SReadRequest se termine en retournant DONE, la réponse à la fin d'activité étant contenue dans la structure SSAPindication.

4) RAPPORT D'EXCEPTION.

Les routines d'ISODE sont dans la figure 8.12.

PRIMITIVES ISO	ROUTINES ISODE
S-U-EXCEPTION-REPORT-REQ	SUReportRequest
S-U-EXCEPTION-REPORT-IND	(SReadRequest)
S-P-EXCEPTION-REPORT-IND	(SReadRequest)

figure 8.12. les rapports d'exceptions.

5) GESTION DES JETONS.

La figure 8.13. décrit la gestion des jetons d'ISODE.

PRIMITIVES ISO	ROUTINES ISODE
S-TOKEN-GIVE-REQ	SGTokenRequest
S-TOKEN-PLEASE-REQ	SPTokenRequest
S-CONTROL-GIVE-REQ	SGControlRequest
(toutes les indications)	(SReadRequest)

figure 8.13. la gestion des jetons.

6) GESTION DE LA SYNCHRONISATION.

Les routines de la gestion de la synchronisation sont regroupées dans la figure 8.14. Leurs arguments sont les mêmes que la norme avec en plus un descripteur de session et une structure SSAPindication modifiée en cas de problème.

PRIMITIVES ISO	ROUTINES ISODE
S-SYNC-MINOR-REQ	SMinSyncRequest
S-SYNC-MINOR-RESP	SMinSyncResponse
S-SYNC-MAJOR-REQ	SMajSyncRequest
S-SYNC-MAJOR-RESP	SMajSyncResponse
S-RESYNCHRONIZE-REQ	SReSyncRequest
S-RESYNCHRONIZE-RESP	SReSyncResponse
(toutes les indications)	(SReadRequest)
(toutes les confirmations)	(SReadRequest)

figure 8.14. la gestion de la synchronisation.

Voyons comment se déroule dans ISODE la pose de points de synchronisation majeure. Nous supposons que tout marche normalement.

-Le client pose un point de synchronisation majeure en appelant SMajSyncRequest. Cette routine va elle-même appeler SMajSyncRequestAux, responsable en fait de la plupart du travail.

-SMajSyncRequest ayant fini, le client appelle SReadRequest pour attendre la réponse.

-Le serveur a appelé SReadRequest. Cette routine se termine avec la valeur DONE. Un rapide examen de la structure SSAPindication permet au serveur de voir de quoi il s'agit (il détecte un S-SYNC-MAJOR-INDICATION). Il répond grâce à SMajSyncResponse (cette routine confiera la plupart de son travail à SMajSyncResponseAux).

-La réponse arrive finalement chez le client, provoquant le fin de SReadRequest avec la valeur DONE.

7) ABANDON BRUTAL.

L'implémentation de l'abandon brutal est résumée dans la figure 8.15.

PRIMITIVES ISO	ROUTINES ISODE
S-U-ABORT-REQ	SUAbortRequest
S-U-ABORT-IND	(SReadRequest)
S-P-ABORT-IND	(retour de la valeur NOTOK)

figure 8.15. l'abandon brutal de connexion.

8) ABANDON NEGOCIE.

L'implémentation de l'abandon négocié est résumé dans la figure 8.16.

PRIMITIVES ISO	ROUTINES ISODE
S-RELEASE-REQ	SRelRequest
S-RELEASE-RESP	SRelResponse
S-RELEASE-IND	(SReadRequest)
S-RELEASE-CONF	(SRelRequest)

figure 8.16. l'abandon négocié.

8.2.3. LES STRUCTURES D'INFORMATIONS.

La structure C la plus répandue au niveau 5 est **SSAPindication**. Elle se compose d'un indicateur et d'une union de sous-structures. Ces sous-structures sont : SSAPdata, SSAPtoken, SSAPsync, SSAPactivity, SSAPreport, SSAPfinish, SSAPabort. Son rôle est soit d'expliquer ce qui a pu provoquer un problème lors de l'exécution d'une routine, soit de décrire un événement reçu par SReadRequest autre que des données. Ces sous-structures sont détaillées dans l'annexe 15.

9. LE SERVICE TRANSPORT.

9.1. RAPPEL THEORIQUE.

9.1.1. FONCTION.

Le rôle essentiel de la couche de transport est d'assurer le transport de bout en bout de l'information, en débarrassant les niveaux supérieurs de tous les détails des opérations réalisées pour assurer un transport fiable, transparent et efficace des données. Les fonctions réalisées à ce niveau concernent le multiplexage, la segmentation et le groupement des unités de données, ainsi que la régulation du flux de bout en bout. Le niveau transport comporte également des mécanismes de traduction entre les adresses de transport et celles de réseau, ainsi que des moyens qui permettent d'établir ou de fermer des connexions à travers le réseau. Il peut enfin se charger d'un contrôle d'erreurs. Ce service est décrit dans (5). La norme du service fait l'objet de (21) et celle du protocole de (22).

9.1.2. ELEMENTS DE SERVICE.

La figure 9.1. donne la liste des éléments de service, leurs paramètres sont dans l'annexe 16.

PRIMITIVES	TYPE	FONCTION
T-CONNECT	confirmé	établir une connexion
T-DATA	non confirmé	transfert de données
T-EXPEDITED-DATA	non confirmé	transfert de données exprès
T-DISCONNECT	non confirmé fournisseur	abandon de connexion

figure 9.1. éléments de service.

Signalons au sujet du T-CONNECT que si le correspondant éloigné accepte la connexion, il utilisera la primitive T-CONNECT-RESPONSE. S'il la refuse, il utilisera plutôt le T-DISCONNECT. D'autre part , le service T-DISCONNECT n'est pas négocié, l'abandon ne peut donc pas être refusé. Il peut être déclenché soit par le fournisseur transport, soit pas l'utilisateur.

9.1.3. LE PROTOCOLE.

La norme définit 5 classes de protocole.

1) La classe 0, la plus simple, fournit un service transport élémentaire.

2) La classe 1 permet de récupérer certaines erreurs de la couche réseau, principalement les déconnexions et le "reset".

3) La classe 2 fournit des capacités de multiplexage.

4) La classe 3 procure les capacités des classes 1 et 2.

5) La classe 4 permet les fonctionnalités de la classe 3 plus la capacité de détecter et de recupérer des erreurs du type : perte de TPDU, duplication ou corruption de TPDU.

Les différents TPDU sont regroupés dans la figure 9.2. Ils sont détaillés dans l'annexe 17. Les modalités complètes d'utilisation des TPDU sont décrites dans (22).

TPDU	FONCTION
CR	demande de connexion
CC	confirmation de connexion
DR	demande d'abandon
DC	confirmation d'abandon
DT	données normales
ED	données exprès
AK	accusé de réception de données
EA	accusé de réception de données exprès
RJ	rejet
ER	erreur

figure 9.2. les TPDU.

9.2. LE SERVICE TRANSPORT D'ISODE.

9.2.1. GENERALITES.

Le service implémenté est présenté dans (17). ISODE implémente une interface entre les couches supérieures (OSI) et les différents services de transport possibles qui dépendent de l'environnement réel du système où ISODE est installé. Les différentes possibilités supportées par ISODE consistent en un service TP0 pouvant fonctionner au dessus de TCP et X25 ou un service TP4 pour le SunLink OSI.

9.2.2 IMPLEMENTATION DES PRIMITIVES DE SERVICES.

1) T-CONNECT.

L'implémentation du T-CONNECT est décrite par la figure 9.3. TConnResponse n'est utilisée qu'en cas d'acceptation de la connexion. Dans l'hypothèse d'un refus, c'est une routine TDiscRequest qu'appellera le répondeur, conformément à la norme.

PRIMITIVES ISO	ROUTINES ISODE
T-CONNECT-REQUEST	TConnRequest
T-CONNECT-INDICATION	(TInit)
T-CONNECT-RESPONSE	TConnResponse
T-CONNECT-CONFIRMATION	(TConnRequest)

figure 9.3. le T-CONNECT.

Les figures 9.4 et 9.5 comparent les arguments des routines avec la norme.

ISO	ISODE
adresse appelée	called
adresse appelante	calling
présence du service exprès	expedited
qualité de service	qos
données utilisateur	data cc
-	tc
-	td

figure 9.4. comparaison des arguments de TConnRequest avec la norme.

"Tc" est un pointeur vers une structure TSAPconnect contenant la réponse affirmative à une connexion, "td" pointe vers une structure TSAPdisconnect utile si un problème est apparu ou si la connexion a été refusée, "data" contient les données de l'utilisateur et "cc" en donne la longueur.

ISO	ISODE
-	sd
adresse répondante	responding
présence du service exprès	expedited
qualité de service	qos
données de l'utilisateur	data cc
-	td

figure 9.5. comparaison des arguments de TConnResponse avec la norme.

"Sd" est un entier utilisé comme descripteur de la connexion, "td" pointe vers une structure TSAPdisconnect consultée en cas de problème.

2) TRANSFERT DE DONNEES.

La figure 9.6. décrit l'implémentation des transferts de données.

PRIMITIVES ISO	ROUTINES ISODE
T-DATA-REQUEST	TDataRequest/TWriteRequest
T-EXPEDITED-DATA-REQUEST	TExpdRequest
(toute indication)	(TReadRequest)

figure 9.6 le transfert de données.

TDataTequest et TWriteRequest sont semblables en nature mais utilisent une représentation différentes des données. Ainsi, les paramètres de TDataRequest sont :

int sd : descripteur de transport,
char *data : données,
int cc : longueur des données,
struct TSAPdisconnect *td : paramètre utile en cas d'échec de la routine.

Ceux de TWriteRequest sont :

int sd : descripteur de transport,
struct udvec *uv : données sous forme d'un tableau d'éléments,
struct TSAPdisconnect *td : informations relatives à un éventuel problème.

La routine TReadRequest sert à attendre soit un T-DATA-INDICATION soit un T-EXPEDITED-DATA-INDICATION. Ses arguments sont :

int sd : descripteur de transport,
struct TSAPdata *tx : données reçues,
int secs : temps d'attente maximum,
struct TSAPdisconnect *td : informations sur un éventuel problème.

3) T-DISCONNECT.

L'abandon ou le refus de connexion sont réalisés par un appel à TDiscRequest (figure 9.7).

ISO	ISODE
-	sd
données de l'utilisateur	data cc
-	td

figure 9.7. comparaison des arguments de TDiscRequest avec la norme.

"Sd" est le descripteur de transport et "td" une structure d'information utile en cas d'échec.

9.2.3. SERVICE D'ECOUTE

ISODE fournit à un processus les moyens d'écouter après certaines connexions. Ces facultés d'écoute peuvent être utilisées par n'importe quelle couche supérieure du système. Un programme commence à écouter après une connexion en appelant **TNetListen(ta, td)**
struct TSAPaddr *ta : adresse transport sur laquelle on écoute,
struct TSAPdisconnect *td : modifié en cas de problème.

Si l'appel réussit, le programme écoute après des connexions pouvant arriver sur l'adresse indiquée.

Pour vérifier quand une nouvelle connexion est en attente ou quand d'anciennes connexions ont de l'activité, on appelle **TNetAccept(vecp, vec, nfds, rfds, wfds, efds, secs, td)**

```
int                     *vecp,
char                    **vec  : vecteur d'initialisation pour la nouvelle
                                 connexion,
int                     nfds,
fd_set                  *rfds  : descripteur de connexion à utiliser avec xselect,
                        *wfds : idem,
                        *efds  : idem,
int                     secs   : temps d'attente maximum pour une activité,
struct TSAPdisconnect *td      : modifié en cas de problème.
```

Si l'appel à TNetAccept réussit, la valeur de "vecp" devra être vérifée. Si elle est plus grande que 0, alors une nouvelle connexion est arrivée. TInit puis TConnResponse seront normalement appelés.

Pour arrêter d'écouter, on appelle **TNetClose** dont les arguments sont les mêmes que TNetListen.

9.2.4. LE DEMON TSAPD.

Le démon **tsapd** est un serveur de transport dont la fonction est d'attendre et de traiter des demandes de connexion. Dans ce but, il réalise les actions suivantes :

1) Mise en écoute grâce à un appel à TNetListen.
2) Attente d'une demande de connexion par une série d'appels à TNetAccept.
3) Si une connexion arrive (TNetAccept retourne la valeur OK et son paramètre "vecp" est plus grand que 0), après analyse de ses arguments, avertissement de l'homologue session **ssapd.**
4) Appel par ssapd à TInit (pour générer un T-CONNECT-INDICATION).
5) Appel à TConnResponse.
6) Appel à TReadRequest pour attendre la suite. Ainsi, si après cette attente, un CONNECT-SPDU survient, ssapd en avertira l'entité présentation supérieure.

9.2.5. LES STRUCTURES D'INFORMATIONS.

ROUTINES	STRUCTURES ASSOCIEES
TInit	TSAPstart
TConnRequest	TSAPconnect
TReadRequest	TSAPdata

figure 9.8. les structures d'informations.

Les principales structures d'informations de la couche 4 sont reprises dans la figure 9.8. Outre celles-ci, chaque routine possède un pointeur vers une structure TSAPdisconnect utilisée si l'appel échoue. Leurs arguments font l'objet de l'annexe 17.

TSAPstart contient tous les paramètres d'un T-CONNECT-IND, TSAPdisconnect signale un abandon du fournisseur ou de l'utilisateur et TSAPconnect contient la réponse à une demande de connexion. TSAPdata contient les données reçues par TReadRequest plus un paramètre spécifiant si elles sont arrivées via le service de transfert exprès.

10. EXEMPLES.

10.1. INTRODUCTION.

Dans ce chapitre, nous présentons l'utilisation des couches décrites dans les chapitres précédents. Nous abordons la vision théorique d'ISO et l'implémentation réalisée dans ISODE.

10.2. EXEMPLE 1.

10.2.1. CAS THEORIQUE.

Cet exemple est inspiré de (5). C'est celui de l'établissement d'une association. Nous supposons qu'une connexion réseau a déjà été établie mais pas une connexion transport. Nous nous plaçons dans le cadre d'une application utilisant FTAM. Les lecteurs intéressés par FTAM sont invités à consulter (23). L'exemple est illustré par les figures 10.1 à 10.3.

Voyons la séquence d'événements :

-(1) : Demande d'établissement d'un régime d'association FTAM.
-(2) : Demande d'établissement d'une association.
-(3) : Demande d'établissement d'une connexion présentation.
-(4) : Demande d'établissement d'une connexion session.
-(5) : Demande de création d'une connexion transport.
-(6) : Expédition du CR TPDU sur une connexion réseau par un N-DATA-REQ.
-(7) : Arrivée des données à l'entité transport du destinataire.
-(8) : L'entité transport informe l'entité session supérieure de la demande de connexion
-(9) : Réponse de l'entité session.
-(10) : Transport de cette réponse.
-(11) : Arrivée de la réponse chez l'entité transport.
-(12) : Confirmation de la demande de connexion transport. L'entité session a la réponse de son correspondant.

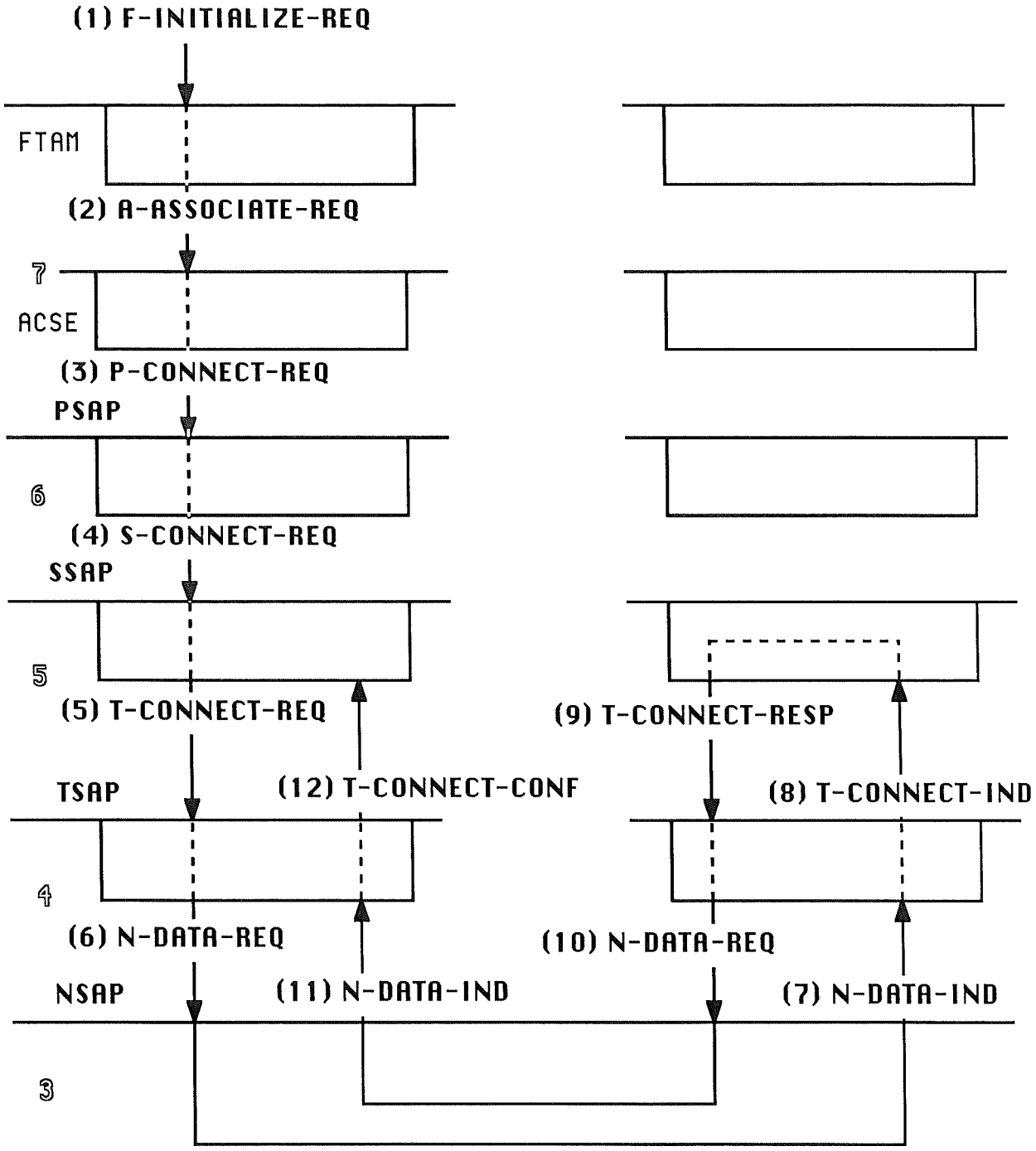

figure 10.1. le F-INITIALIZE.

-(13) : Expédition d'un CN SPDU de demande de connexion session contenant dans sa partie "données de l'utilisateur" le CP PPDU, la AARQ APDU et l'APDU de FTAM.
-(14) : Passage des données à la couche réseau.
-(15) et (16) : Arrivée des données chez le destinataire.
-(17) : Après réception et interprétation du CN SPDU, l'entité session en informe son supérieur.

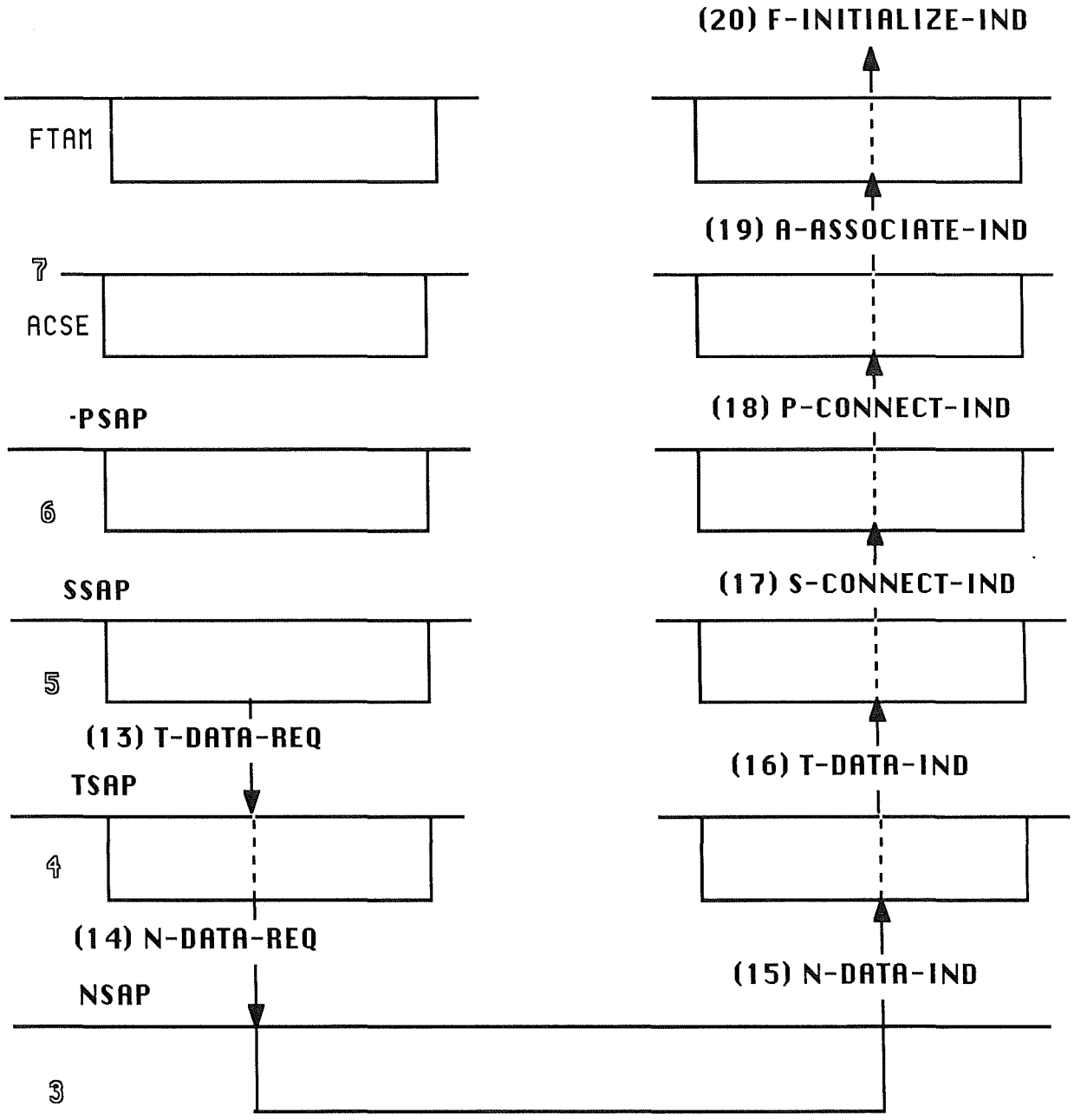

figure 10.2. le F-INITIALIZE.

-(18) : Après réception et interprétation du CP PPDU, l'entité présentation en informe l'ACSE.
-(19) : Après réception et interprétation de l'AARQ APDU, l'ACSE en informe FTAM.
-(20) : Acceptation du régime d'association par FTAM et signalisation à l'application.

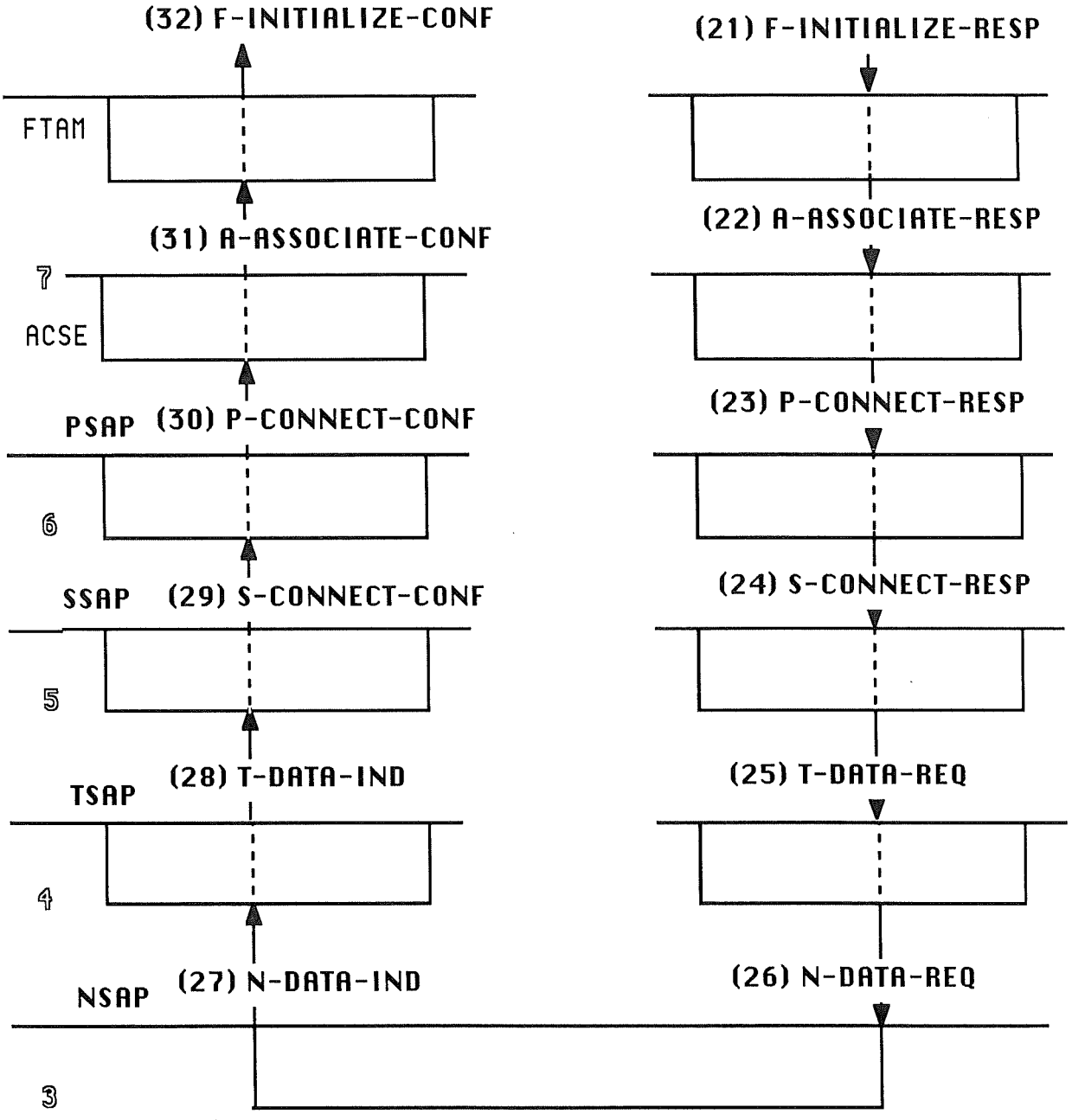

figure 10.3. le F-INITIALIZE.

-(21) : Réponse au régime d'association.
-(22) : Réponse à l'association. La réponse au régime d'association est dans la partie "données de l'utilisateur" du A-ASSOCIATE-RESP.
-(23) : Le AARE APDU est dans la partie "données de l'utilisateur" du P-CONNECT-RESP.
-(24) Passage du CPA PPDU dans la partie "données de l'utilisateur" du S-CONNECT-RESP.
-(25) Transfert de l'AC SPDU dans un T-DATA-REQ.
Les étapes 26 à 32 ne nécessitent pas d'autres explications.

10.2.2. L'IMPLEMENTATION D'ISODE.

Nous supposerons dans notre exemple que tout se passe normalement comme dans le cas théorique. Voici d'abord la séquence des événements.

CLIENT	*	SERVEUR
début FInitializeRequest	*	début Tsapd
début AcAssocRequest	*	début TNetAccept (avec temps attente
début PConnRequest	*	infini)
début SConnRequest	*	
début TConnRequest	*	
	*	fin TNetAccept en retournant OK, son
	*	paramètre "vecp" est plus grand que 0
	*	début Ssapd
	*	début TInit
	*	TInit retourne OK
	*	exament de TSAPstart modifié par
	*	TInit
	*	exécution de TConnResponse
	*	début TReadRequest
fin TConnRequest en	*	
retournant OK	*	
examen TSAPconnect	*	
exécution de TDataRequest	*	
début TReadRequest	*	
	*	fin TReadRequest avec valeur OK
	*	examen TSAPdata
	*	consultation de isoservices
	*	lancement de l'application visée
	*	début FInit
	*	début AcInit
	*	début PInit
	*	début SInit
	*	fin SInit avec valeur OK
	*	fin PInit avec valeur OK
	*	fin AcInit avec valeur OK
	*	fin FInit avec valeur OK
	*	début FInitializeResponse
	*	début AcAssocResponse
	*	début PConnResponse
	*	début SConnResponse
	*	exécution de TDataRequest
	*	fin SConnResponse
	*	fin PConnResponse
	*	fin AcAssocResponse

```
                                        *    fin FInitializeResponse
    fin TReadRequest avec  OK           *
     examen de TSAPdata                 *
     modification de SSAPconnect        *
   fin SConnRequest                     *
   examen de SSAPconnect                *
   SSAPconnect copié dans               *
    PSAPconnect                         *
  fin PConnRequest                      *
   examen PSAPconnect                   *
   modification AcSAPconnect            *
  fin AcAssocRequest                    *
  examen AcSAPconnect                   *
  modification FTAMconnect              *
fin FInitializeRequest                  *
*****************************************************************
```

Au niveau des couches supérieures du client, la demande d'association se traduit par une série d'appels emboîtés (de FInitializeRequest à TConnRequest). Dès que TConnRequest se termine (événement de T-CONNECT-CONFIRMATION), l'entité session expédie la demande de connexion des couches supérieures par un appel à TDataRequest. Ensuite, elle attend patiemment la réponse en appelant TReadRequest. Quand TReadRequest se termine (T-DATA-INDICATION, la routine retourne la valeur OK), l'entité session peut examiner la réponse à la demande de connexion session (examen de TSAPdata). Si tout va bien, un S-CONNECT-CONFIRMATION sera produit vers l'entité présentation grâce à la fin d'exécution normale de SConnRequest (retour de la valeur OK). Les paramètres de la confirmation sont contenus dans la structure SSAPconnect. Ensuite, de la même manière, tous les autres appels emboîtés vont se terminer, générant autant d'événements de confirmation.

Chez le serveur, le démon Tsapd est à l'écoute de toute demande d'établissement de connexion (appel à TNetAccept). Dès qu'une demande est détectée, Tsapd commande le lancement de Ssapd (niveau session). Ssapd va appeler la routine TInit. La fin de son exécution est synonyme de T-CONNECT-INDICATION. Ensuite, Ssapd examine la demande du client (contenue dans SSAPstart) et y répond grâce à TConnResponse. Puis il se met en attente (appel à TReadRequest). Lorsque TReadRequest se termine (arrivée d'un T-DATA-INDICATION), Ssapd voit qu'il s'agit d'une demande d'association. Il va alors consulter la base de données Isoservices pour déterminer l'application locale visée. Il va ensuite lancer l'exécution de cette dernière.

La première chose que fera l'application sera d'appeler FInit (générant par là une série d'appels emboîtés jusqu'à SInit). La fin normale de SInit est équivalente à un S-CONNECT-INDICATION (et ainsi de suite jusqu'à FInit). Il ne reste plus alors à cette application qu'à y répondre grâce à

FInitializeResponse et à la série d'appels emboîtés que cela occasionne.

10.3. EXEMPLE 2.

10.3.1. CAS THEORIQUE.

L'exemple est aussi issu de (5). Il met en oeuvre la réalisation de l'élément de service F-DATA de FTAM (23). Il est illustré à la figure 10.4.

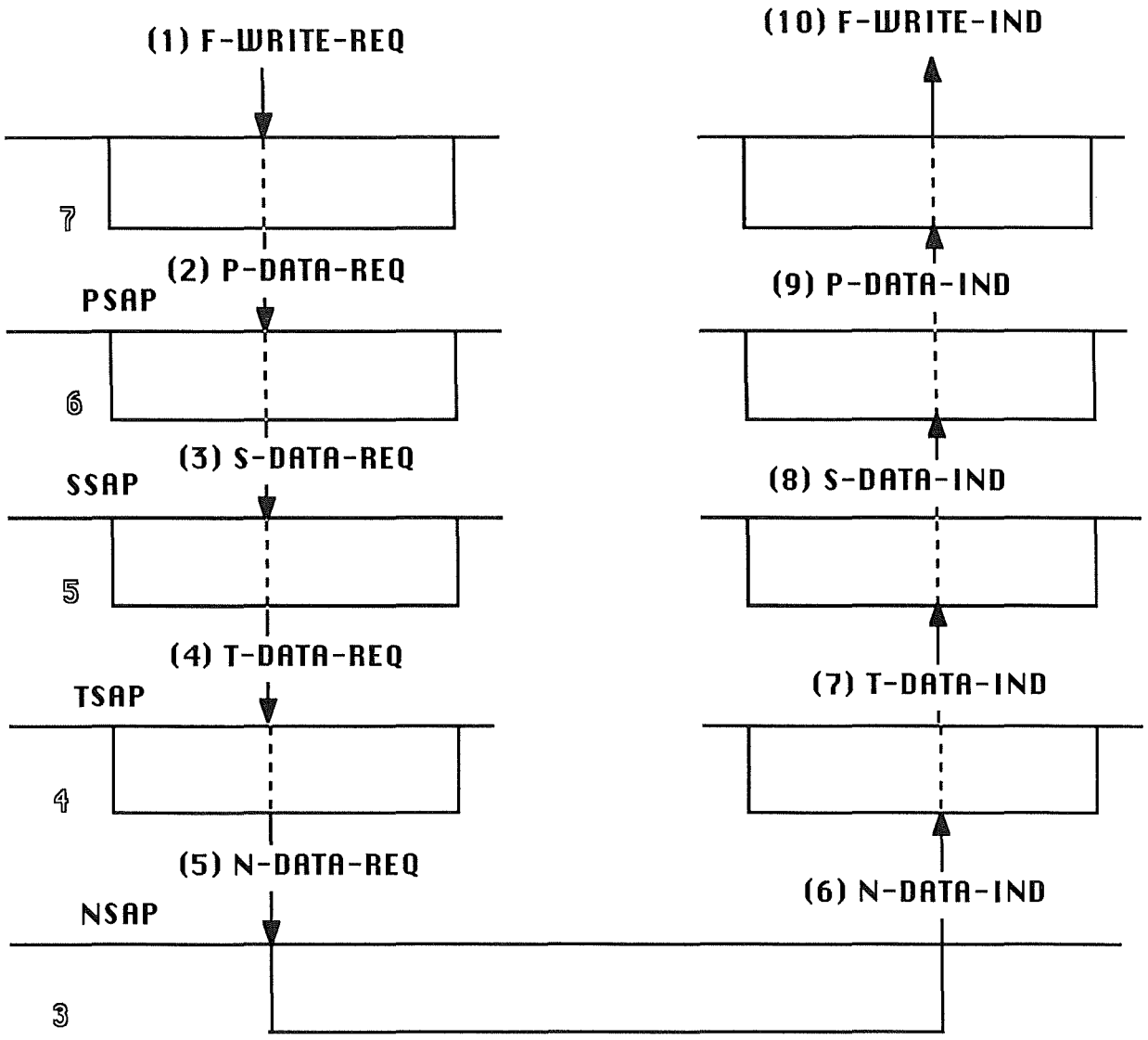

figure 10.4. le F-WRITE.

Voici la séquence d'événements.

-(1) : Production de la requête par l'émetteur.
-(2) : Passage des données de l'entité FTAM vers l'entité présentation.
-(3) : Après la transformation syntaxique, transmission des données à l'entité session.
-(4) à (8) : Livraison du message par les fournisseurs de session, de transport et de réseau.
-(9) : Après réalisation des transformations syntaxiques nécessaires, transmission des données vers l'entité FTAM.
-(10) : Arrivée des données chez l'application.

10.3.2. L'IMPLEMENTATION D'ISODE.

Voici d'abord la séquence des événements.

```
      CLIENT                    *        SERVEUR

début FDataRequest              * début FWaitRequest
 début PDataRequest             *  début PReadRequest
  début SDataRequest            *   début SReadRequest
   début TWriteRequest          *    début TReadRequest
   fin TWrite Request           *
  fin SDataRequest              *
 finPDataRequest                *
fin FDataRequest                *
                                *    fin TReadRequest avec valeur OK
                                *    examen TSAPdata
                                *   fin SReadRequest avec valeur OK
                                *   examen SSAPdata
                                *  fin PReadRequest
                                *  examen PSAPdata
                                * fin FWaitRequest
                                * FTAMdata contient les données
**************************************************************
```

Au niveau du client, l'expédition des données se traduit par une série d'appels emboîtés (de FDataRequest à TWriteRequest).

Quant au serveur, il doit s'être mis en état d'attendre ces données. Cela se manifeste par une série d'appels emboîtés à des routines d'attente (de FWaitRequest à TReadRequest). Lorsque les données arrivent, TReadRequest retourne la valeur OK. L'examen de la structure TSAPdata permet de faire la

différence entre un T-DATA-INDICATION et un T-EXPEDITED-DATA-INDICATION. Ensuite, et de la même manière, les autres routines se termineront, générant autant de (N)-DATA-INDICATIONS.

11. LES BASES DE DONNEES D'ISODE.

11.1. INTRODUCTION.

ISODE utilise et gère plusieurs bases de données. Nous allons dans ce chapitre les évoquer brièvement. Elles sont au nombre de 4 et voici leur nom:
- isoentities,
- isoservices,
- isobjects,
- isotailor.

Elles sont décrites en détails dans (10) et (17).

11.2. LA BASE DE DONNEES DES ENTITES ISO.

11.2.1. CONTENU ET FONCTION.

La base de données **isoentities** contient des informations concernant les titres d'entités d'applications ISO connus dans le réseau. Cette base de données fait fonction de "directory service" dans ISODE. En elle-même, c'est un fichier texte ASCII ordinaire. Chaque entrée contient les éléments suivants :
-L'hôte et le service associé au titre d'entité d'application. Ces deux éléments forment l'objet descripteur du titre d'entité d'application.
-L'objet identifiant de l'information sur l'entité d'application.
-Les sélecteurs de présentation, de session et de transport de l'adresse de présentation associée. Ils sont interprêtés comme un nombre s'ils commencent par une petite grille, un "string" s'ils apparaissent entre guillements, un "octet string" autrement.
-Une ou plusieurs adresses de réseau associées à l'adresse de présentation.

11.2.2. EXEMPLE.

Voici l'entrée correspondante à l'université de Nottingham :

```
nott default 1.17.4.3.3.1.0 " " " " " "
        X.25 000021000018 03010100
        X.25 23426020017299
```

11.2.3. ROUTINES D'ACCES.

La librairie **libacsap** contient les routines d'accès à cette base de données. Citons la routine **str2aei** qui, à partir d'un hôte et d'un service, retourne une information d'entité d'application; **aei2addr** qui transforme cette entité d'application en une adresse de présentation.

11.3. LA BASE DE DONNEES DES SERVICES ISO.

11.3.1. CONTENU ET FONCTION.

La base de données **isoservices** contient des informations sur les services ISO connus sur l'hôte. Elle consiste en un fichier texte ASCII ordinaire. Chaque entrée contient :
-Le nom du fournisseur.
-Le nom de l'entité résidant sur le fournisseur.
-Le sélecteur utilisé pour identifier l'entité au fournisseur, interprêté comme les sélecteurs dans isoentities.
-Le programme et le vecteur argument à execve(2) quand le service est sélectionné.

11.3.2. EXEMPLE.

Voici comment apparaît l'entrée correspondant au FTAM d'ISODE 4.0. :

```
"tsap/filestore" #258 δ (ETDIR)iso.ftam
```

11.3.3. ROUTINES D'ACCES.

La librairie **libcompat** contient les routines d'accès à isoservices.

11.4. LA BASE DE DONNEES DES OBJETS ISO.

11.4.1. CONTENU.

La base de données **isobjects** contient un "mapping" simple entre les descripteurs d'objets et les identifiants de ces objets. C'est un fichier texte ASCII contenant des informations sur les objets connus sur l'hôte. Chaque ligne se compose du descripteur de l'objet (un simple "string") et de l'identificateur lui correspondant.

11.4.2. EXEMPLE.

L'entrée correspondant au contrôle d'association se présente comme suit :

"acse pci version 1" 2.2.1.0.1

11.4.3. ROUTINES D'ACCES.

Les routines d'accès sont réunies dans la librairie **libpsap.**

11.5. LE FICHIER ISO "SUR MESURE".

Le fichier **isotailor** contient des informations utiles pour la configuration d'exécution d'ISODE. C'est un fichier ASCII dont chaque ligne se compose du nom d'une option, d'un blanc et de la valeur de l'option. On y trouve des paramètres généraux de configuration pour ISODE, d'autres spécifiques au système d'exploitation ou aux interfaces, etc.

12. TCP/IP.

12.1. POURQUOI EN PARLER ?

L'avenir appartient au modèle OSI. C'est du moins l'espoir de ses concepteurs. Malheureusement, OSI n'est pas seul. Il doit faire face à la concurrence de strandards plus anciens avec parmi eux TCP/IP qui figure en très bonne place.

TCP/IP bénéficie de bonnes performances et est très répandu. On peut douter de la spontanéité de ses adeptes à l'abandonner au profit d'OSI. Enfin, TCP/IP est américain et OSI est européen. Quand on voit l'empressement avec lequel les américains acceptent qu'une technologie étrangère remplace une des leurs, on peut se poser des questions sur le destin universel d'OSI.

Voilà de bonnes raisons de dire quelques mots sur TCP/IP. De plus, cette présentation aura aussi l'avantage de décrire une autre manière d'aborder le problème de la téléinformatique que celle proposé par OSI. TCP/IP est présenté dans (24).

12.2. QU'EST CE QUE C'EST ?

12.2.1. GENERALITES.

En gros, on peut dire que **TCP** est l'équivalent d'une couche 4 OSI orientée connexion (comme TP4). **IP** correspond plutôt à une couche réseau sans connexion.

C'est un peu un abus de langage que de parler de protocole TCP/IP. En fait, TCP et IP sont les porte-drapeaux d'une famille de protocoles appelée souvent la **suite de protocoles INTERNET**. TCP et IP en étant les plus connus, on a par extension donné leur nom à toute la famille INTERNET.

12.2.2. SERVICES OFFERTS.

Les services les plus traditionnels de la famille INTERNET sont les suivants :

-**Transfert de fichiers** grâce au protocole **FTP**. Ce service permet soit de transférer des fichiers, soit d'en consulter à distance. La sécurité est prise en compte : un utilisateur A désirant effectuer un transfert de fichier avec une machine X éloignée doit spécifier un nom d'utilisateur et un mot de passe.

-**Remote login** avec le protocole **TELNET**. Ce service permet de se connecter à n'importe quel ordinateur éloigné dans un réseau. Un utilisateur va ainsi dialoguer avec une machine A alors qu'il est en fait sur une machine B. Le protocole TELNET lui rendra la machine B complètement invisible, tout se passant comme s'il n'avait que A en face de lui (figure 12.1).

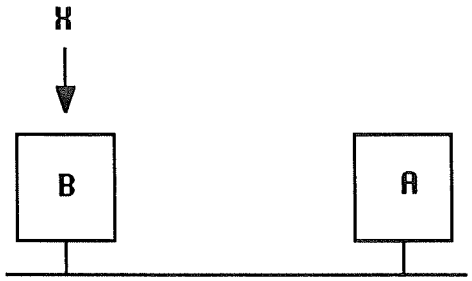

figure 12.1. remote login : X travaille avec A à partir de B.

-**Messagerie électronique** : c'est une sorte de service postal permettant d'envoyant des messages vers d'autres utilisateurs sur d'autres machines.

Ces services correspondent aux besoins rencontrés sur d'anciens réseaux où chaque hôte possédait toutes les ressources dont il avait besoin. On peut dire que chaque ordinateur du réseau était auto-suffisant. Mais de nos jours, les réseaux ont changé. Les appareils connectés sont de plus en plus variés. Ils sont de plus en plus configurés pour réaliser des tâches spécialisées utiles à d'autres. Si les utilisateurs travaillent toujours sur une seule machine, celle-ci fait de plus em plus appel à d'autres pour lui rendre des services spécialisés. Cel nous mène au **modèle client-serveur**.

Un **serveur** est un système qui fournit un service spécialisé au reste du réseau. Un **client** est un autre système qui fait appel à un serveur pour effectuer une tâche spécifique. Voyons les différents services spécialisés capables d'utiliser TCP/IP :

-**Système de fichiers de réseau** : ce type de service permet à une machine de travailler sur des diques qui ne lui appartiennent pas sans que l'utilisateur ne s'en rende compte. Il existe ainsi des ordinateurs sans disque, entièrement dépendant de ceux gérés par des serveurs de fichiers.

-**Impression éloignée** : il est possible d'utiliser les imprimantes d'autres ordinateurs comme si c'étaient les nôtres.

-**Exécution éloignée** : on peut demander qu'un certain programme soit exécuté pour nous sur une autre machine.

-**Serveur de nom** : les gros systèmes doivent gérer d'importantes collections de noms : noms d'utilisateurs + leur mot de passe, noms des hôtes du réseau + leur adresse, ... Pour éviter que toutes ces données soient conservées sur chaque machine, on les regroupe dans des bases de données dont on confie la garde à quelques systèmes spécialisés. Les autres systèmes auront accès à tous ces noms par l'intermédiaire du réseau.

-**Serveur de terminaux** : un serveur de terminaux est un petit ordinateur juste capable de lancer l'exécution correcte de TELNET. Il permet d'éviter d'avoir chaque terminal directement connecté à une seule machine particulière.

-**Système de fenêtres orienté réseau** : ce service permet à un programme de s'exécuter sur l'écran d'un autre ordinateur, ce qui peut s'avérer très utile dans le cas d'applications graphiques.

12.3. DESCRIPTION DES PROTOCOLES TCP ET IP.

12.3.1. GENERALITES.

La pluspart du temps, les applications TCP/IP utilisent 4 couches :
-une couche application comme le courrier électronique,
-un protocole s'assurant que les données transmises par l'application arriveront bien à l'autre bout de la ligne (exemple : TCP),
-IP qui fournit un service de transfert de **datagrammes**,
-un protocole gérant l'usage d'un moyen physique de communication comme **ETHERNET**.

Un point important est que TCP/IP se base sur une technologie sans connexion : les informations sont transmises comme une séquence de datagrammes. Chacun d'entre eux est envoyé individuellement dans le réseau.

12.3.2. LE NIVEAU TCP.

TCP a pour fonction de découper les données à transmettre en datagrammes. Ceux-ci sont passés à IP avec leur destination. A l'autre bout, TCP réunira les datagrammes pour reconstituer le message original. TCP doit assurer un service orienté connexion alors qu'il se base sur un service sans connexion. Il se charge aussi du recouvrement des erreurs et du contrôle de flux.

En pratique, TCP reçoit un flot de données comme schématisé à la figure 12.2.

.....

figure 12.2. message reçu par TCP.

TCP découpe d'abord le tout en morceaux plus petits (figure 12.3.).

....

figure 12.3. les morceaux du message.

Ensuite, TCP ajoute à chaque morceau un en-tête (représenté dans la figure 12.4. par "T").

T..... T..... T..... T..... T..... T..... T..... T.....

figure 12.4. les morceaux avec l'en-tête.

Tous ces morceaux sont alors passés à IP.

La figure 12.5. décrit le contenu de l'en-tête.

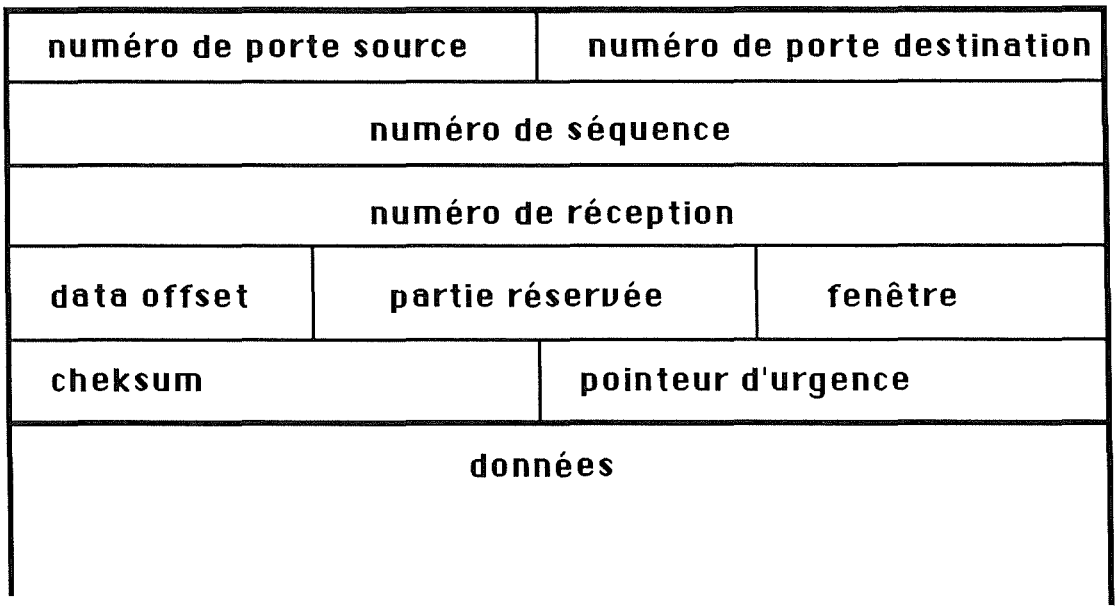

figure 12.5. l'en-tête de TCP.

Voyons quelques éléments importants ;

-Les **numéros de porte** : TCP alloue un numéro de porte à l'émetteur et un autre au destinataire pour pouvoir les distinguer.

-Le **numéro de séquence** : chaque datagramme possède un numéro de séquence pour permettre au destinataire de les remettre dans le bon ordre. En fait, ce que TCP numérote, ce sont les octets. Si les morceaux de données ont une taille de 500 octets, les numéros de séquence seront : 500, 1000, 1500, ... et non : 1, 2, 3.

-Le **numéro de réception** : il indique le numéro du dernier datagramme reçu par le destinataire. Cette partie est remplie quand le destinataire envoie un avis de réception.

-La **fenêtre** : elle est utilisée pour effectuer un contrôle de flux, ce qui autorise l'émetteur à envoyer plusieurs datagrammes à la fois sans devoir attendre pour chacun un avis de réception.

-**Cheksum** : cette partie permet de détecter des erreurs de transmission.

-**Pointeur d'urgence** : ce pointeur demande au destinataire d'aller voir directement à un certain octet contenant le plus souvent une commande urgente.

12.3.3. LE NIVEAU IP.

IP va donc recevoir de TCP des datagrammes comme ceux de la figure 12.6. avec l'adresse où les envoyer.

T..... T..... T..... T..... T..... T..... T..... T.....

figure 12.6. datagrammes reçus par IP.

La mission d'IP est de faire parvenir ces datagrammes à bon port. Pour faciliter le routage, IP ajoute son propre en-tête représenté dans la figure 12.7. par "I".

IT..... IT..... IT..... IT..... IT..... IT..... IT.....

figure 12.7. datagrammes après ajout de l'en-tête IP.

Le contenu de l'en-tête est repris à la figure 12.8.

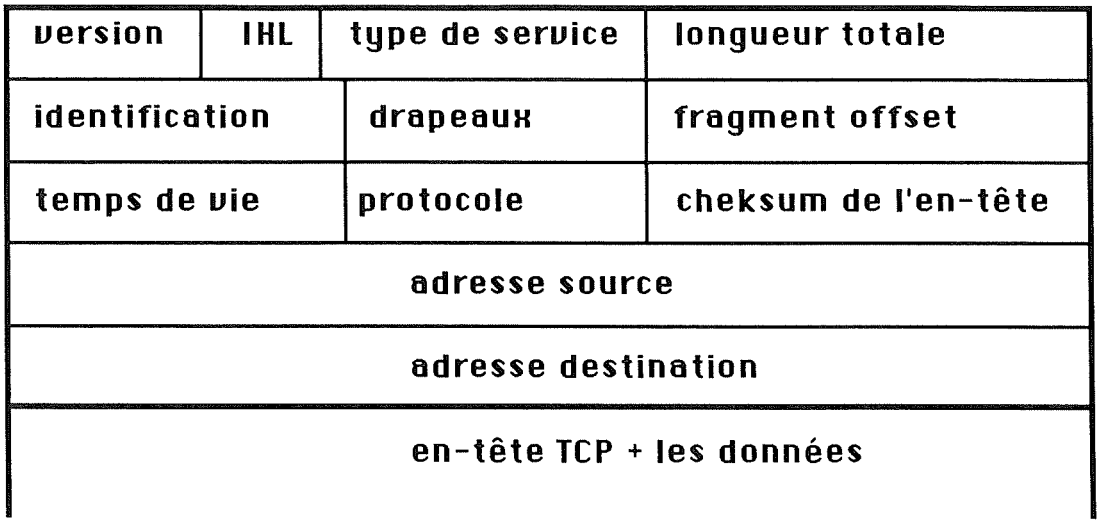

figure 12.8. l'en-tête IP.

Voyons les éléments les plus intéressants :

-Les **adresses source et destination** : des adresses INTERNET de 32 bits qui sont en fait les adresses des machines engagées dans la communication.

-Le **numéro de protocole** : indique à l'IP destinataire à quel protocole envoyer le datagramme. C'est souvent TCP mais pas toujours.

-**Cheksum de l'entête** : permet à l'IP destinataire de voir si l'en-tête IP a été endommagé.

-Le **temps de vie** : c'est un nombre donnant la durée de vie permise au datagramme. Au fur et à mesure que le temps passe, ce nombre est décrémenté. Quand il atteind zéro, le datagramme est détruit.

12.3.4. LE NIVEAU ETHERNET.

De nombreux réseaux utilisent ETHERNET comme moyen physique de transmission, c'est donc de lui que nous parlerons. Tout d'abord, ETHERNET utilise des adresses qui lui sont propres (les adresses ETHERNET) et qui n'ont rien à voir avec les adresses INTERNET utilisées par IP. Chaque controleur ETHERNET contient une adresse ETHERNET en son sein dès sa fabrication. Cette adresse (aussi appelée adresse physique) a 48 bits. Sa fonction est la suivante : chaque fois qu'un paquet ETHERNET arrive en face d'un contrôleur, ce dernier regarde l'adresse physique du destinataire. Si c'est la sienne, il "prend" le paquet. Sinon, il n'examine rien du paquet et le laisse continuer son chemin. Pour établir un lien entre les adresses INTERNET et ETHERNET, chaque machine conserve une table de correspondance.

ETHERNET ajoute à tout datagramme un en-tête "E" et une queue "C" contenant un cheksum d'ETHERNET comme le montre la figure 12.9.

EIT.....C EIT.....C EIT.....C EIT.....C EIT.....C EIT.....C

figure 12.9. les datagrammes d'ETHERNET.

La figure 12.10 résume le contenu d'un datagramme.

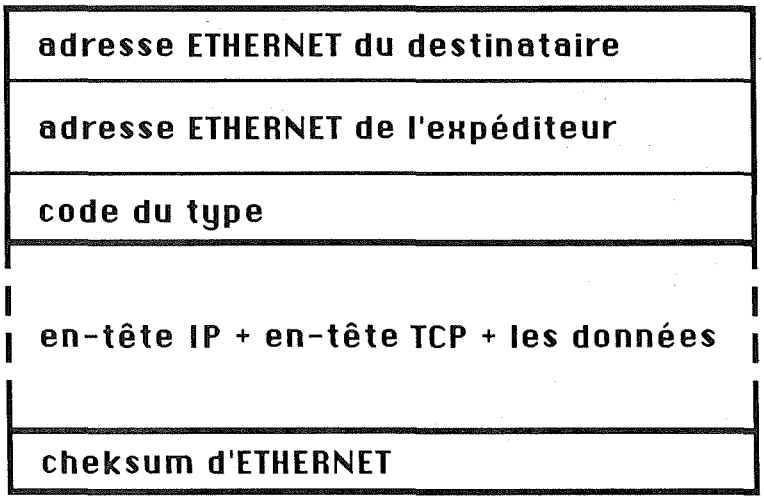

figure 12.10. un datagramme ETHERNET.

Le code de type indique la famille de protocole utilisant le réseau. Le cheksum d'ETHERNET est calculé sur tout le paquet ETHERNET. Si, grâce à lui, le destinataire détecte une erreur de transmission, le paquet est purement et simplement jeté.

12.3.5. LES SOCKETS BIEN CONNUS.

Nous avons vu comment un flux de données provenant de la couche application était traité et transmis par les couches inférieures. Remontons dans la couche application et voyons comment se passe une ouverture de connexion entre 2 machines.

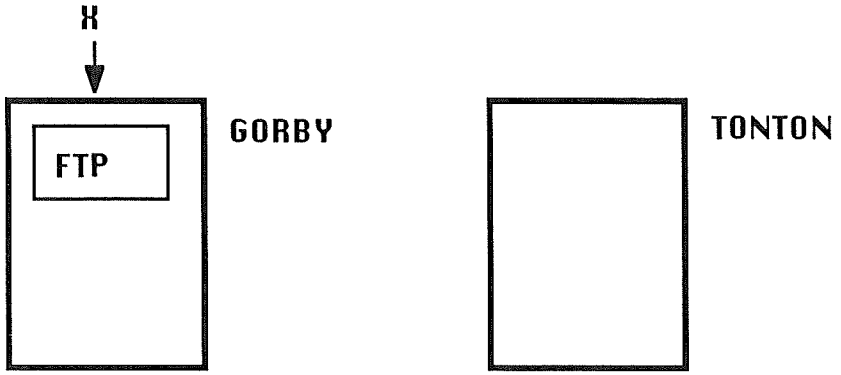

figure 12.11. lancement de FTP sur GORBY.

Supposons qu'un utilisateur X sur la machine GORBY décide d'effectuer un transfert de fichier avec la machine TONTON. Il va lancer l'exécution de FTP sur GORBY (figure 12.11.).

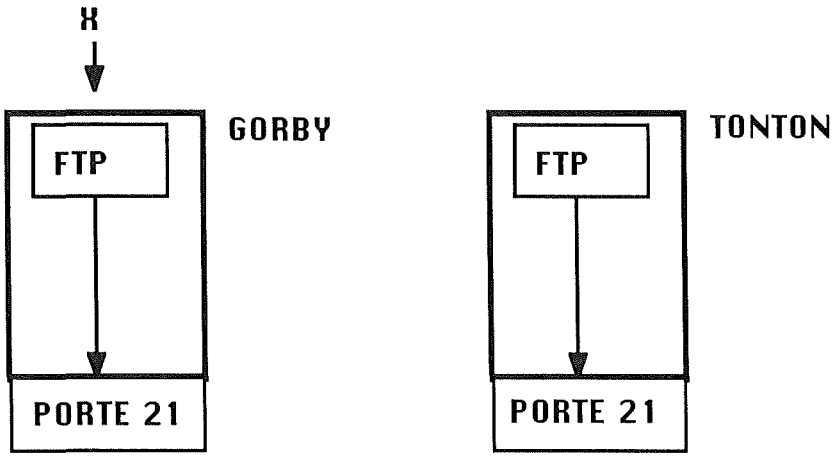

figure 12.12. le socket bien connu de FTP.

Or, TCP utilise des numéros de porte pour différencier les applications en communication. Chacune d'entre elles a, sur son hôte, un numéro de porte qui lui est propre, c'est ce qu'on appelle un **socket bien connu**. Ainsi, le numéro de porte officiel de TFP est 21 (figure 12.12).

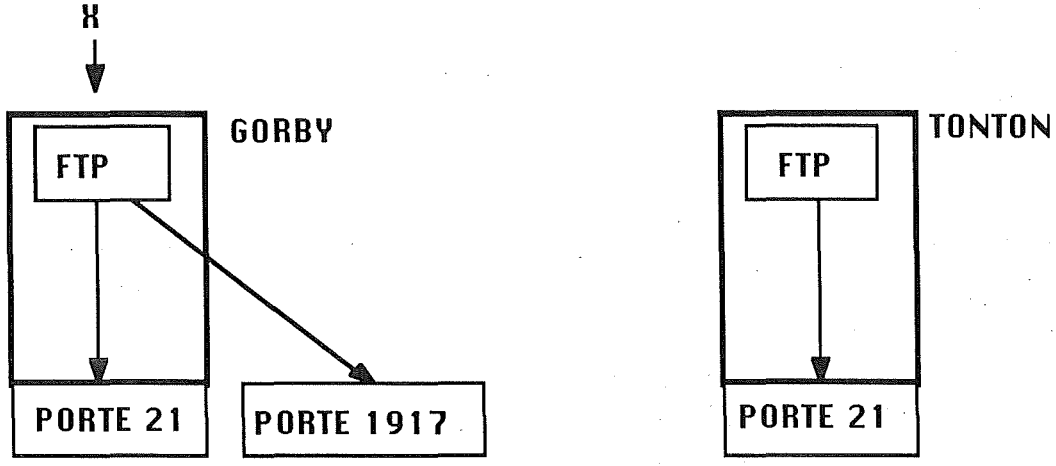

figure 12.13. le dédoublement de porte.

Sur GORBY, FTP va créer une nouvelle porte alléatoire avec un numéro. En effet, si toutes les applications utilisant FTP faisaient appel à la porte 21, on ne pourrait pas différencier 2 connexions entre les 2 mêmes machines (figure 12.13.).

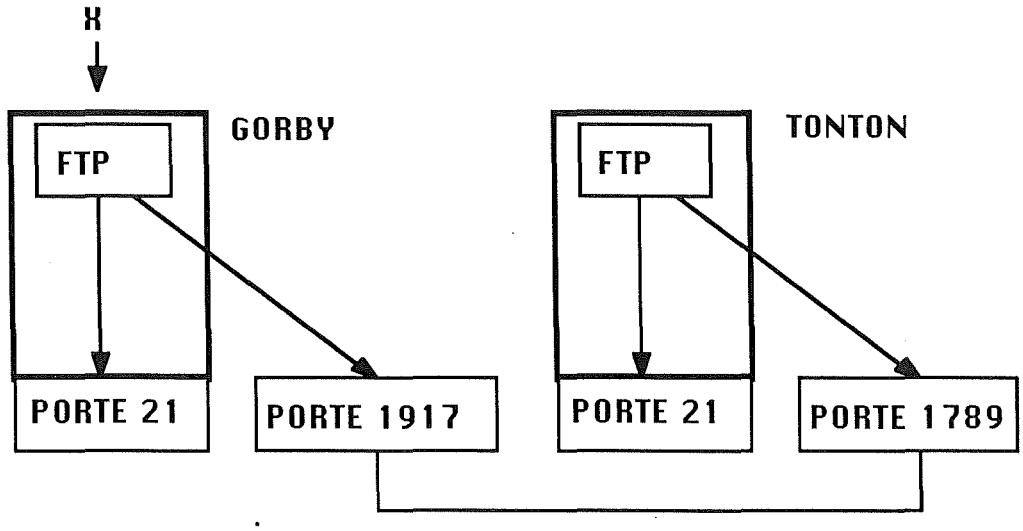

figure 12.14. l'échange sur les portes dédoublées.

Ensuite, le système cherchera l'adresse de TONTON et le numéro de porte pour parler avec FTP (21). La connexion va s'établir. Sur TONTON, un démon est à l'écoute. Recevant une demande de connexion sur la porte 21, il l'accepte et il dédouble cette porte (figure 12.14.).

Puis il envoie un paquet d'acceptation vers GORBY. Dans la discussion qui suivra, les données circuleront entre ces portes dupliquées, réservées uniquement à cet échange (1917 et 1789), tandis que les commandes circuleront entre les sockets bien connus (21). Ainsi, l'ouverture verra partir un paquet avec comme porte origine le numéro 1917 et 21 comme numéro de porte destinataire. Le paquet d'acceptation partira de la porte 1789 vers la porte 1917. A la fin, X expédira sur le socket 21 une demande de fermeture (c'est une commande et non une donnée). Les portes dupliquées seront alors supprimées.

12.3.6. LES PASSERELLES.

Dans le monde TCP/IP, on peut trouver plusieurs réseaux différents interconnectés. Pour que des machines puissent communiquer, il faut passer par un mécanisme de **passerelle** (figure 12.15.).

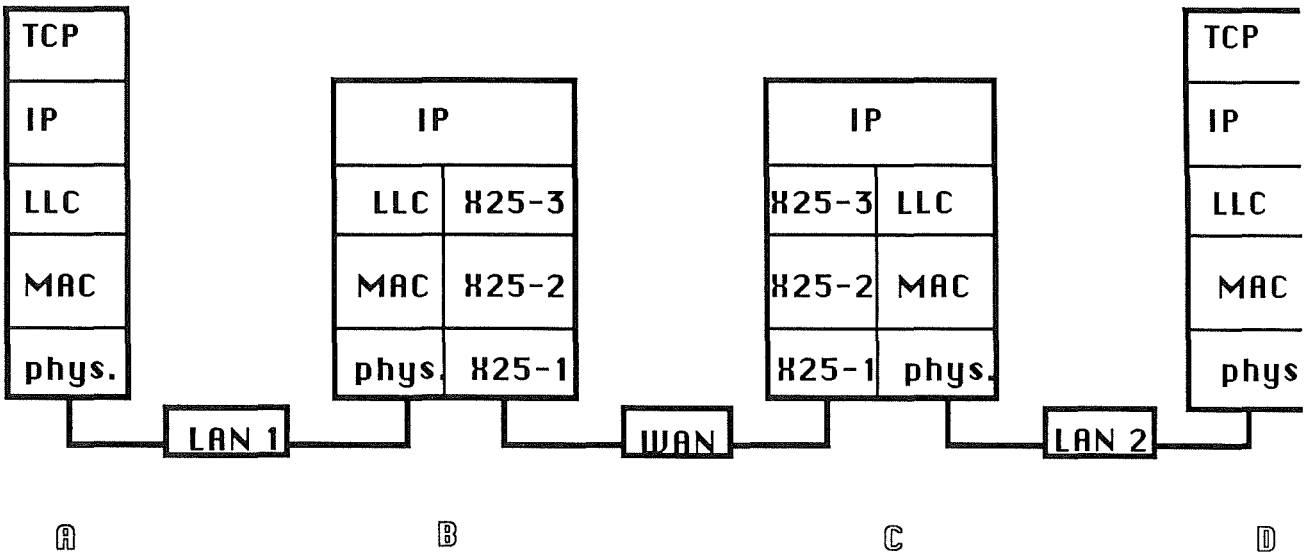

figure 12.15. les passerelles.

La machine A envoie un message vers B situé sur un autre réseau que LAN1 auquel appartient A. Pour résoudre ce problème, A fait appel à un système connecté à plusieurs réseaux appelé "passerelle" (ici : B et C). De passerelle en passerelle, de réseau en réseau, le message de A finira par atteindre D. En général, des machines comme A contiennent des tables leur indiquant les passerelles idéales pour envoyer des messages vers d'autres réseaux.

12.3.7. ARP.

IP et ETHERNET utilisent des adresses différentes. Or, pour envoyer un message, nous avons besoin des 2 adresses. Un problème peut apparaître quand on veut communiquer avec quelqu'un dont on connait l'adresse IP mais pas l'adresse physique. Pas de panique ! Il existe un protocole spécial appelé **ARP** (pour "address resolution protocol) et des tables ARP. Chaque entrée de la table contient une adresse IP et son adresse physique associée. Les machines d'un réseau s'échangent des informations pour maintenir ces tables suivant le protocole ARP. Si une machine A désire connaître l'adresse physique d'une machine dont l'adresse INTERNET est B, elle va émettre une requête ARP concernant cette adresse. La requête sera envoyée à tous. La seule personne autorisée à y répondre, c'est B. B enverra donc une réponse contenant son adresse INTERNET avec son adresse physique et A pourra modifier sa table ARP.

13. EN PRATIQUE.

13.1 INTRODUCTION.

Nous abordons ici quelques considérations quant à l'utilisation pratique d'ISODE. La description précise des procédures de configuration (adaptation des fichiers de configuration à la configuration réelle), de génération (compilation) et d'installation (placement des fichiers là où il faut) est réalisée dans le fichier ISODE-GEN d'ISODE.

13.2. OPTIONS POSSIBLES.

13.2.1. INTRODUCTION.

Lors de la configuration d'ISODE, il est possible de choisir différentes options relatives à l'environnement local. Tout cela est expliqué dans le fichier config/OPTIONS. Nous en présentons ici les principales.

13.2.2. SYSTEME D'EXPLOITATION.

ISODE supporte trois systèmes : l'UNIX de Berkeley,
: l'UNIX de AT&T,
: l'émulation UNIX de Ridge.

Pour l'UNIX d'AT&T, trois sous-options sont possibles, à savoir :
-le système V version 3,
-S5RZ avec les améliorations HP,
-S5RZ avec les amèliorations IBM.

13.2.3. RESEAU ET TRANSPORT.

ISODE supporte n'importe quelle combinaison de TCP/IP avec TP0, X25 avec TP0, TP4, BRIDGE_X25 (une passerelle TCP/X25). De plus, pour chaque choix, différentes options mutuellement exclusives sont offertes. Ainsi, pour X25, il faut choisir entre : le X25 de SUNLINK version 4.0 ou 5.0,
: l'interface CAMTEC,
: l'interface CAMTEC avec des sockets,
: le X25 d'UBC.

13.2.4. AUTRES POSSIBILITES.

Une fois ISODE installé, il est également possible d'installer l'implémentation de FTAM à condition d'être soit dans un système UNIX de Berkeley ou dans un UNIX d'AT&T (système V).

Un prototype du service de directory OSI est également disponible mais uniquement sur des systèmes UNIX de Berkeley.

14. CONCLUSIONS.

L'objectif principal de ce mémoire était d'une part l'étude des couches supérieures du modèle OSI et d'autre part l'examen d'une implémentation de ce modèle, à savoir le logiciel ISODE 4.0.

Ce travail a été décrit dans les pages précédentes. Il nous a permis de compléter notre vision globale du modèle OSI après les cours magistraux consacrés aux couches inférieures. L'étude d'ISODE nous a donné une vision très concrète d'un modèle fort théorique et abstrait. Il serait intéressant de le comparer avec d'autres implémentations existantes et de voir si elles peuvent communiquer entre elles.

Pour les aspects pratiques, la seule fonctionnalité immédiatement utilisable est l'implémentation minimale de FTAM. Il serait intéressant de voir les éventuelles améliorations apportées à ce sujet par la version 5.0. de même que les implémentations annoncées d'un service de directory et d'un terminal virtuel.

Quoiqu'il en soit, l'objectif des concepteurs d'ISODE de permettre une meilleure diffusion du modèle OSI grâce à cette implémentation des couches supérieures nous semble en bonne voie et nous sommes curieux de voir les futures applications d'ISODE appelées à se développer.

BIBLIOGRAPHIE.

-(1) "Distributed Systems : concept and design" par G.F. Coulouris et J. Dallimore chez International Computer Science Série. 1988.

-(2) "Téléinformatique 1" par Nussbaumer aux Presses Polytechniques Romandes. 1987 Lausanne.

-(3) "Introduction à la téléinformatique" par J.M. Munier chez E. Eyrolles. 1986 Paris.

-(4) "ISO 7498 : Information processing systems - Open systems interconnection, basic reference model 1984". International organisation for standardisation. Suisse 1984.

-(5) "OSI explained" par John Henshall et Sandy Shaw. Ellis Harwood. 1988 Chichester.

-(6) Marshall T. Rose : "The ISO development environment : user's manual, volume 1-4". Palo Alto USA. July 1988.

-(7) Ph. Cayphas : "Contribution à la conception d'application pour systèmes ouverts : les éléments de service d'application et leur développement dans ISODE 4.0., réalisation d'une application les utilisant." Mémoire présenté en vue de l'obtention du titre de licencié et maître en informatique. FNDP Namur. 1989.

-(8) CCITT : blue book, vol. VIII, fascicle VIII.4. "Data communication networks, open systems interconnection (OSI), model and notation service definition. Recommendation X.217 : association control service definition for open systems interconnection for CCITT application". Melbourne, november 1988.

-(9) CCITT : blue book, vol. VIII, fascicle VIII.4. "Data communication networks, open systems interconnection (OSI), protocol specifications conformance testing. Recommendation X.227 : association control protocol specification for open systems interconnection for CCITT applications". Melbourne, november 1988.

-(10) Marshall T. Rose." The ISO development environment : user's manual, volume 1 : application services". Palo Alto USA. July 1988.

-(11) CCITT : blue book, vol. VIII, fascicle VIII.4. "Data communication networks, open systems interconnection (OSI), model and notation service definition. Recommendation X.218 : reliable transfer, model and service definition". Melbourne, november 1988.

-(12) CCITT : blue book, vol VIII, fascicle VIII.5. "Data communication networks, open sustems interconnection (OSI), protocol specifications, conformance testing. Recommendation X.228 : reliable transfer, protocol specification". Melbourne, november 1988.

-(13) CCITT : blue book, vol. VIII, fascicle VIII.4. "Data communication networks, open systems interconnection (OSI), model and notation service definition. Recommendation X.219 : remote operations, model, notation and service definition". Melbourne, november 1988.

-(14) CCITT : blue book, vol. VIII, fascicle VIII.5. "Data communication networks, open system interconnection (OSI), protocol specifications, conformance testing. Recommendation X.229 : remote operations, protocol specification". Melbourne, november 1988.

-(15) CCITT : blue book, vol VIII, fascicle VIII.4. "Data communication networks open systems interconnection (OSI). Model and notation, service definition. Recommendation X.216 : definition del servicio de presentacion para la interconexion de sistemas abiertos para aplicaciones del CCITT". November 1988.

-(16) CCITT : blue book, vol VIII, fascicle VIII.5. "Data communication networks. Open systems interconnection (OSI). Protocol specification, conformance testing. Recommendation X.226 : presentation protocol specification for open system interconnection for CCITT applications". November 1988.

-(17) Marshall T.Rose. "The ISO development environment : user's manual, volume 2 : underlying services". Palo Alto USA. July 1988.

-(18) F Caneschi, E. Merelli. "Standardizing the presentation layer. Why and what ? ". 7th international conference on distributed systems. Berlin, september 1987.

-(19) CCITT : blue book, vol VIII, fascicle VIII.4. "Data communication netwoeks open systems interconnection (OSI). Model and notation, service definition. Recommendation X.215. : session service definition for open systems interconnection for CCITT applications.

-(20) CCITT : blue book, vol VIII, fascicle VIII.5. "Data communication networks. Open systems interconnection (OSI). Protocol specification, conformance testing. Recommendation X.225. : session protocol specification for open systems interconnection for CCITT applications.

-(21) CCITT : blue book, volume VIII, fascicle VIII.4. "Data communication networks open systems interconnection (OSI). Model and notation, service definition. Recommendation X.214. : Transport service

definition for open system interconnection for CCITT applications". November 1988.

-(22) CCITT : blue book, volume VIII, fascicle VIII.5. "Data communication networks open systems interconnection (OSI). Protocol specification, conformance testing. Recommendation X.226. : transport protocol specification for open system interconnection for CCITT applications". November 1988.

-(23) "Contribution à l'étude d'un logiciel de transfert, d'accés et de gestion de fichiers". Mémoire présenté par Th. Debatty. Année académique 1988-1989. FNDP Namur.

-(24) Charles Hedrick, Rutgers : The state university of New Jersey. "Introduction to Internet protocols". July 1987. Listing du Computer Science Facilities Group.

ANNEXE 1 : ELEMENTS DE SERVICE DE L'ACSE.

Nous regroupons ici les paramètres des éléments de service de l'ACSE.

1) A-ASSOCIATE.

Voyons d'abord les paramètres visant à déterminer la nature de l'association.

-Le mode : identification de la version du protocole en vigueur (normal ou X410-1984).
-Le qualificatif des entités applications appelées, appelantes et répondantes : identification de ces diverses AE.
-Le titre des AP appelés, appelants et répondants : identifiant de ces différents AP.
-L'identificateur d'invocation des AE appelées, appelantes et répondantes: identification de ces invocations.
-L'identificateur d'invocation des AP appelés, appelants et répondants.
-Le nom du contexte d'application : identifiant du contexte caractérisant l'association.
-Les données de l'utilisateur : paramètre laissé libre pour l'application.
-Le résultat : acceptation ou refus de l'association proposée.
-La source du résultat : service d'origine du résultat.
-Le diagnostic : explication d'un éventuel rejet de l'association.

Signalons qu'appelant désigne celui qui (AE ou AP) désire établir l'association, qu'appelé désigne celui avec qui l'appelant désire communiquer et que répondant désigne celui qui répond effectivement à la demande d'association de l'appelant. Voyons maintenant les paramètres concernant les couches Session et Présentation.

-L'adresse présentation de l'appelant, de l'appelé et du répondant.
-La liste des définitions du contexte de présentation et son résultat.
-Le nom du contexte de présentation par défaut et son résultat.
-Une qualité de service.
-Les exigences de présentation et de session.
-Le numéro du point de synchronisation initial.
-L'installation initiale des jetons.
-L'identificateur de connexion session.

2) A-RELEASE.

-La raison : la raison de la demande de fermeture.
-Les données de l'utilisateur : les données finales définies dans le protocole.
-Le résultat : l'acceptation ou le refus de la fin d'association.

3) A-ABORT.

-La source : service responsable de l'abandon.
-Les données de l'utilisateur : paramètre réservé à l'application et défini par le protocole utilisé.

4) A-P-ABORT.

-La raison du fournisseur.

ANNEXE 2 : LES APDU DE L'ACSE.

Voici les paramètres des APDU de l'ACSE.

1) AARQ.

-La version du protocole ACSE employé.
-Le nom du contexte d'application.
-Le titre de l'AE appelée.
-Le titre de l'AE appelante.
-Des informations sur l'implémentation.
-Les données de l'utilisateur.

Notons que le titre d'AE regroupe 4 informations, à savoir :
-Le titre de l'AP.
-Le qualificatif de l'AE.
-L'identificateur d'invocation de l'AP.
-L'identificateur d'invocation de l'AE.

2) AARE.

-La version du protocole.
-Le résultat.
-Le titre de l'AE répondante.
-Le nom du contexte d'application.
-La source du résultat.
-Des informations sur l'implémentation.
-Les données de l'utilisateur.

3) RLRQ.

-La raison de l'abandon.
-Les données de l'utilisateur.

4) RLRE.

-La raison.
-Les données de l'utilisateur.

5) ABRT.

-La source de l'abandon.
-Les données de l'utilisateur.

ANNEXE 3 : LES STRUCTURES D'INFORMATIONS DE L'ACSE D'ISODE.

1) ACSAPCONNECT.

```
Struct AcSAPconnect (
int                     acc_sd              : descripteur de
                                              l'association,
                        acc_result          : réponse à l'association,
                        acc_diagnostic      : diagnostic de l'association,
OID                     acc_context         : nom du contexte
                                              d'application,
AEIinfo                 acc_respondtitle    : info. sur l'AE répondante,
structure PSAPconnect   acc_connect         : info. du niveau
                                              présentation,
int                     acc_info            : taille des données de
                                              l'utilisateur,
PE                      acc_info (NACDATA) : données de l'utilisateur).
```

2) ACSAPSTART.

```
Struct AcSAPstart (
int                 acs_sd              : descripteur de l'association,
OID                 acs_context         : nom du contexte d'application,
AEIinfo             acs_callingtitle    : info. sur l'AE appelante,
AEIinfo             acs_calledtitle     : info. sur l'AE appelée,
struct PSAPstart    acs_start           : info. du niveau présentation,
int                 acs_ninfo           : taille des données de
                                          l'utilisateur,
PE                  acs_info (NACDATA)  : données de l'utilisateur).
```

3) ACSAPRELEASE.

```
Struct AcSAPrelease(
int   acr_affirmative       : réponse à la demande d'abandon,
      acr_reason            : raison de la réponse,
      acr_ninfo             : taille des données utilisateur,
char acr_info(NACDATA) : données de l'utilisateur).
```

4) ACSAPINDICATION.

La structure AcSAPindication est en fait une union discriminée, c'est-à-dire qu'elle se compose d'une étiquette et d'une union de structures. L'étiquette indique quelle structure de l'union est utile.

Les structures de cette union sont AcSAPabort et AcSAPfinish dont les définitions suivent.

Struct AcSAPabort(
int aca_source : source de l'abandon,
acar_reason : raison de l'abandon,
aca_ninfo : taille des données de l'utilisateur,
PE aca_info(NACDATA) : données de l'utilisateur).

Struct AcSAPfinish(
int acf_reason : raison de l'abandon négocié,
acf_ninfo : taille des données de l'utilisateur,
char acf_info(NACDATA) : données de l'utilisateur).

AcSAPindication a trois fonctions :

-Elle contient les paramètres d'un problème éventuellement survenu lors d'un appel à AcAssocRequest, AcAssocResponse, AcRelRequest ou AcRelResponse.

-Elle comprend les informations concernant un abandon négocié d'association (au sein de AcSAPfinish) dans le cas d'un appel à AcFINISHser.

-Elle contient enfin les informations utiles sur un abandon brutal (dans AcSAPabort) dans l'hypothèse d'un appel à AcUAbortRequest ou à AcABORTser.

ANNEXE 4 : ELEMENTS DE SERVICE DU RTSE.

Voici la liste des paramètres des éléments de service du RTSE.

1) RT-OPEN.

Outre les paramètres transmis à l'ACSE (déjà vus dans l'annexe 1), examinons ceux spécifiques au RTSE :
-le mode de dialogue : soit un dialogue soit un monologue,
-le tour initial : choix de celui qui parlera le premier,
-le protocole d'application : le protocole d'application en vigueur sur l'association,
-les données de l'utilisateur : paramètre laissé libre pour l'utilisateur.

2) RT-CLOSE.

Les paramètres du RT-CLOSE sont les mêmes que ceux du A-RELEASE à l'exception du résultat, absent chez le RTSE.

3) RT-TRANSFER.

-APDU : unité de données de protocole d'application à envoyer,
-temps de transfert : temps de transfert maximum autorisé,
-résultat : échec ou réussite du transfert.

4) RT-TURN-PLEASE.

-priorité : urgence de la demande.

5) RT-TURN-GIVE.

Le RT-TURN-GIVE n'a aucun paramètre.

6) RT-P-ABORT.

Ce service n'a pas de paramètre.

7) RT-U-ABORT.

-données de l'utilisateur : contenu laissé libre pour l'application utilisatrice.

ANNEXE 5 : LES APDU DU RTSE.

Voici les paramètres des APDU du RTSE.

1) RTORQ.

-taille du point de contrôle : quantité maximale de données (en KiloBytes) pouvant être transmise entre 2 points de synchronisation mineurs,
-taille de la fenêtre : nombre maximum de confirmations de poses de points de synchronisation mineurs à attendre avant de continuer à émettre.
-mode de dialogue,
-données de l'utilisateur,
-identifiant de connexion session,
-protocole application.

2) RTOAC.

-taille du point de contrôle,
-taille de la fenêtre,
-données de l'utilisateur,
-identificateur de connexion session.

3) RTORJ.

-raison du refus,
-données de l'utilisateur.

4) RTTR.

-données à expédier.

5) PTTP.

-indice de priorité.

6) RTAB.

-raison de l'abandon,
-paramètre posant problème,
-données de l'utilisateur.

ANNEXE 6 : LES STRUCTURES D'INFORMATIONS DU RTSE D'ISODE.

1) TRSAPSTART.

```
Struct RtSAPstart (
int                  rts_sd       : descripteur de l'association,
                     rts_mode     : mode de dialogue,
                     rts_turn     : propriétaire du tour initial,
PE                   rts_data     : données de l'utilisateur,
struct AcSAPstart rts_start       : informations du niveau ACSE)
```

2) RTSAPCONNECT.

```
Struct RtSAPconnect (
int                    rtc_sd       : descripteur de l'association,
                       rtc_result   : réponse à l'association,
PE                     rtc_data     : données de l'utilisateur,
stuct AcSAPconnect rtc_connect : informations de l'ACSE)
```

3) RTSAPINDICATION.

Cette structure décrit soit le problème rencontré par une routine qui a retourné la valeur NOTOK comme résultat, soit l'événement enregistré par RtWaitRequest. Tout comme AcSAPindication, elle se compose de plusieurs sous-structures dont nous donnons ici la description.

a) Struct RtSAPabort (

```
int   rta_peer                : source du problème,
      rta_reason              : raison du problème,
PE    rta_data                : éventuelles données de l'utilisateur,
char rta_data(RTA_SIZE) : texte diagnostique du fournisseur)
```

Cette sous-structure est concernée par les coupures brutales d'associations (du fournisseur ou de l'utilisateur).

b) Struct RtSAPtransfer (

```
PE rtt_data : données à transférer)
```

Les données transférées et reçues par RtWaitRequest sont contenues dans cette sous-structure.

c) Struct RtSAPturn (

```
int rtu_please   : différencie un RT-TURN-PLEASE-INDICATION d'un
                   RT-TURN-GIVE-INDICATION,
int rtu_priority : priorité de la demande du tour)
```

RtSAPturn contient les informations concernant la gestion du tour de parole.

d) Struct RtSAPclose (
int rtc_dummy)

Cette sous-structure comprend les données relatives à un événement du type X410-CLOSE-INDICATION.

e) Struct AcSAPfinish : voir l'ACSE.

AcSAPfinish concerne les demandes négociées d'abandon d'association.

ANNEXE 7 : ELEMENTS DE SERVICE DU ROSE.

Les arguments des éléments de service du ROSE sont rassemblés ici.

1) RO-INVOKE.

-valeur de l'opération : un identifiant de l'opération connu des 2 AE,
-classe de l'opération,
-arguments de l'opération,
-identifiant de la requête : cet identifiant permet de relier la requête avec les éventuelles réponses ultérieures auxquelles elle donnera lieu,
-identificateur lié : si l'opération est de type "lié",
-priorité : urgence des APDU à transférer et correspondant à la requête.

2) RO-RESULT.

-valeur de l'opération,
-résultat de l'opération,
-identificateur de la requête d'invocation : identifie la requête dont on communique le résultat,
-priorité.

3) RO-ERROR.

-valeur de l'erreur : identifie l'erreur pour les 2 AE,
-paramètre de l'erreur : contient des informations supplémentaires sur l'erreur,
-identifiant de la requête d'invocation,
-priorité.

4) R0-REJECT-U.

-raison du rejet,
-identifiant de la requête d'invocation,
-priorité.

5) RO-REJECT-P.

-identifiant de la requête d'invocation,
-APDU éventuellement non transféré en raison du problème,
-raison du rejet.

Signalons que ces deux derniers paramètres sont mutuellement exclusifs.

ANNEXE 8 : LES APDU DU ROSE.

Voici les paramètres des APDU du ROSE.

1) ROIV.

-identificateur d'invocation,
-identificateur lié : en cas d'opération liée,
-valeur de l'opération,
-arguments.

2) RORS.

-identificateur d'invocation,
-résultat,
-valeur de l'opération.

3) ROER.

-identificateur d'invocation,
-valeur de l'erreur,
-paramètre de l'erreur.

4) RORJ.

-identificateur d'appel,
-problème rencontré.

ANNEXE 9 : LES STRUCTURES D'INFORMATIONS DU ROSE D'ISODE.

La seule structure d'infomation apparaissant dans les routines du ROSE est **RoSAPindication**. Elle contient soit la raison de l'échec éventuel d'une routine, soit la description de l'événement rencontré par RoWaitRequest. Comme ses homologues du RTSE et de l'ACSE, elle se compose d'un indicateur et d'une union de sous-structures que nous allons examiner en donnant la liste de leurs paramètres.

1) ROSAPINVOKE.

Struct RoSAPinvoke(
int rox_id : identifiant d'invocation de la requête,
rox_linked : identifiant d'invocation de la requête liée,
rox_nolinked : signale si l'opération est liée ou non,
rox_op : code d'opération,
PE rox_args : arguments de l'opération).

Cette structure, modifiée par RoWaitRequest, décrit une opération invoquée.

2) ROSAPRESULT.

Struct RoSAPresult(
int ror_id : identifiant d'invocation de la réponse, identique à celui de la requête générant cette réponse,
ror_op : code d'opération,
PE ror_result : résultats de l'opération).

RoSAPresult, modifiée par RoWaitRequest, décrit un résultat d'une opération.

3) ROSAPERROR.

Struct RoSAPerror(
int roe_id : identifiant d'invocation de la réponse, identique à celui de la requête amenant cette réponse,
roe_error : code d'erreur,
PE roe_param : paramètres de l'erreur).

Cette structure décrit une erreur rencontrée par le correspondant en tentant de réaliser une opération. Elle est modifiée par RoWaitRequest.

4) ROSAPUREJECT.

Struct RoSAPUreject(
int rou_id : identifiant d'invocation de la requête générant le rejet,
rou_noid : utilisé si aucune requête particulière n'a généré le rejet,

rou_reason : raison du rejet).

RoSAPUreject décrit un rejet par l'utilisateur et est modifiée par RoWaitRequest.

5) ROSAPPREJECT.

```
Struct RoSAPpreject(
int    pop_reason          : raison du rejet,
       rop_id              : identifiant de l'APDU qui ne pourrait pas
                             être transmis à cause du rejet,
PE     rop_apdu            : APDU non transmis,
int    rop_cc,
char rop_data(ROP.SIZE) : texte diagnostique du fournisseur).
```

Cette structure décrit un rejet du fournisseur ROSE. Elle peut être modifiée par toutes les routines du ROSE.

6) ROSAPEND.

```
Struct RoSAPend(
int roe_dummy).
```

RoSAPend signale un événement de type X400-CLOSE-INDICATION. Elle est modifiée par RoWaitRequest.

7) ACSAPINDICATION. (voir ACSE).

Cette structure peut décrire un A-RELEASE-INDICATION ou un RT-CLOSE-INDICATION. Elle est aussi modifiée par RoWaitRequest.

ANNEXE 10 : ELEMENTS DE SERVICE SPECIFIQUES DE LA COUCHE PRESENTATION.

Voici la liste des paramètres des éléments de service du service présentation.

1) P-CONNECT.

-adresse présentation appelante,
-adresse présentation appelée,
-adresse présentation répondante,
-liste des définitions des contextes de présentation,
-résultat de la liste précédente : acceptation ou refus de chaque contexte proposé,
-qualité de service,
-exigences de présentation : unités fonctionnelles optionnelles choisies,
-exigences de session,
-numéro du point de synchronisation initial,
-initialisation des jetons,
-données de l'utilisateur,
-résultat : acceptation ou refus de la connexion.

2) P-U-ABORT.

-données de l'utilisateur.

3) P-P-ABORT.

-raison du fournisseur : raison de l'abandon.

4) P-ALTER-CONTEXT.

-liste de contextes à additionner,
-résultat de cette liste,
-liste de contextes à supprimer,
-résultat de cette liste,
-données de l'utilisateur.

5) P-RELEASE.

-résultat de la demande d'abandon,
-données de l'utilisateur.

ANNEXE 11 : LES PPDU DU PROTOCOLE PRESENTATION.

Les PPDU sont détaillés cette annexe.

1) CP.

-sélecteur de mode,
-version du protocole,
-sélecteur de présentation appelant,
-adresse session appelante,
-sélecteur de présentation appelé,
-adresse session appelée,
-liste de définitions de contextes de présentation,
-nom du contexte par défaut,
-qualité de service,
-exigences de présentation,
-exigences de session,
-exigences de session révisées,
-numéro du point de synchronisation initial,
-initialisation des jetons,
-identifiant de connexion session,
-données de l'utilisateur.

2) CPA.

-sélecteur de mode,
-version du protocole,
-sélecteur de présentation répondant,
-adresse session répondante,
-liste de résultats aux contextes proposés,
-qualité de service,
-exigences de présentation,
-exigences de session,
-exigences de session révisées,
-numéro du point de synchronisation initial,
-initialisation des jetons,
-identificateur de connexion session,
-données de l'utilisateur.

3) CPR.

-version du protocole,
-sélecteur de présentation répondant,
-adresse session répondante,
-liste de résultats des contextes proposés,

-résultat du contexte par défaut,
-qualité de service,
-exigences de session,
-identifiant de connexion session,
-raison du fournisseur,
-données de l'utilisateur.

4) ARU.

-liste d'identifiants de contextes,
-données de l'utilisateur.

5) ARP.

-raison du fournisseur,
-identifiant d'événement.

6) AC.

-liste de contextes à additionner,
-liste de contextes à supprimer,
-données de l'utilisateur.

7) ACA.

-liste de résultats des contextes à additionner,
-liste de résultats des contextes à supprimer,
-données de l'utilisateur.

8) Les PPDU suivants, à savoir : TD, TTD, TE, TC, TCC, n'ont qu'un seul et même paramètre : les données de l'utilisateur.

9) RS.

-type de resynchronisation,
-numéro du point de synchronisation,
-jetons,
-liste d'identifiants de contextes,
-données de l'utilisateur.

10) RSA.

-numéro du point de synchronisation,
-jetons,
-listes d'identifiants de contextes,
-données de l'utilisateur.

ANNEXE 12 : STRUCTURES D'INFORMATIONS DE LA COUCHE PRESENTATION D'ISODE.

1) PSAPSTART.

```
Struct PSAPstart(
int                 ps_sd              : descripteur de la connexion,
structPSAPaddr      ps_calling         : adresse de l'initiateur,
                    ps_called          : adresse de l'appelé,
struct PSAPctxlist  ps_ctxlist         : liste de contextes proposés,
OID                 ps_defctx          : contexte par défaut proposé,
int                 ps_defctxresult    : réponse du fournisseur à
                                         ps_defctx,
                    ps_prequirements   : exigences de présentation,
                    ps_srequirements   : exigences de session,
                    ps_settings        : initialisation des jetons,
long                ps_isn             : point de synchronisation initial,
struct SSAPref      ps_connect         : identifiant de connexion,
int                 ps_ssdusize        : la plus grande taille atomique
                                         permise à chaque SSDU qui sera
                                         transmis sur la connexion,
int                 ps_sversion        : numéro du service session,
struct              QOStype ps_qos     : qualité de service,
PE                  ps_info(NDATA)     : tableau d'éléments de l'utilisateur,
int                 ps_ninfo           : nombre d'éléments du tableau
                                         précédent).
```

2) PSAPCONNECT.

```
Struct PSAPconnect(
int                 pc_sd              : descripteur de la connexion,
struct PSAPaddr     pc_responding      : adresse du répondant,
struct PSAPctxlist  pc_ctxlist         : contextes de présentation,
int                 pc_defctxresult    : réponse au contexte par défaut,
                    pc_prequirements   : exigences de présentation,
                    pc_srequirements   : exigences de session,
                    pc_settings        : initialisation des jetons,
                    pc_please          : jetons que le répondeur veut
                                         posséder,
long                pc_isn             : point de synchronisation initial,
struct SSAPref      pc_connect         : identifiant de connexion utilisé par
                                         le répondeur,
int                 pc_ssdusize        : taille maximale des SSDU,
                    pc_version         : numéro du service session,
```

stuct QOStype	pc_qos	: qualité du service,
int	pc_result	: réponse à la connexion,
char	pc_data(PC_SIZE)	: données de l'utilisateur,
int	pc_cc	: nombre d'éléments dans pc_data).

ANNEXE 13 : ELEMENTS DE SERVICE DE LA COUCHE SESSION.

Voici les arguments des éléments de service de la couche session.

1) S-CONNECT.

-identifiant de connexion session,
-adresse session appelante,
-adresse session appelée,
-adresse session répondante;
-résultat,
-qualité de service,
-exigences de session,
-numéro du point de synchronisation initial,
-initialisation des jetons,
-données de l'utilisateur.

2) TRANSFERT DE DONNEES.

Les 4 primitives de transfert de données (S-DATA, S-EXPEDITED-DATA, S-TYPED-DATA, S-CAPABILITY-DATA) n'ont qu'un argument :
-données de l'utilisateur.

3) GESTION DES JETONS.

Pour le S-TOKEN-PLEASE et le S-TOKEN-GIVE :
-le ou les jetons cédés/demandés,
-des données de l'utilisateur.

Quant au S-CONTROL-GIVE :
- données de l'utilisateur.

4) POINTS DE SYNCHRONISATION.

Pour le S-MAJOR-SYNC et le S-MINOR-SYNC :
-type : soit explicite, soit optionnel,
-un numéro,
-des données de l'utilisateur.

Pour le S-RESYNCHRONIZE :
-type de resynchronisation,
-numéro du point de synchronisation,
-assignation des jetons,
-données de l'utilisateur.

5) RAPPORTS D'EXCEPTIONS.

Pour le S-P-EXCEPTION :
-raison : soit une erreur de protocole, soit une erreur non-spécifiée.

Pour le S-U-EXCEPTION-REPORT
-raison (exemple : erreur de séquence, erreur son spécifique, ..),
-données de l'utilisateur.

6) ACTIVITES.

Le S-ACTIVITY-START :
-identifiant d'activité,
-données de l'utilisateur.

Le S-ACTIVITY-RESUME:
-nouvel identifiant d'activité,
-ancien identifiant d'activité,
-numéro de point de synchronisation
-ancien identifiant de connexion session,
-données de l'utilisateur.

Le S-ACTIVITY-INTERRUPT :
-raison,
-données de l'utilisateur.

Le S-ACTIVITY-DISCARD a les mêmes arguments que le S-ACTIVITY-INTERRUPT.

Le S-ACTIVITY-END :
-numéro du point du synchronisation,
-données de l'utilisateur.

7) S-RELEASE

-résultat,
-données de l'utilisateur.

8) ABANDON BRUTAL.

Le S-U-ABORT :
-des données de l'utilisateur.

Le S-P-ABORT :
-raison de l'abandon.

ANNEXE 14 : LES SPDU DU PROTOCOLE SESSION.

Voici la description des SPDU.

1) CONNECT.

-paramètres d'identification de la connexion,
-taille maximale des TSDU,
-numéro de version du protocole,
-indicateur des capacité de concaténation de l'initiateur,
-paramètres pour l'initialisation des jetons,
-unités fonctionnelles proposées,
-identifiant de l'appelant et de l'appelé,
-données de l'utilisateur,
-data overflow si le SPDU est trop petit pour tout contenir.

2) OVERFLOW ACCEPT.

-taille maximale des TSDU,
-numéro de version du protocole.

3) CONNECT DATA OVERFLOW.

-paramètre indiquant si ce SPDU est ou non le dernier de la série,
-données de l'utilisateur.

4) ACCEPT.

-identificateur de la connexion,
-capacités de concaténation du répondeur,
-taille maximale des TSDU,
-version du protocole,
-initialisation des jetons,
-exigences sur les unités fonctionnelles,
-adresse de l'appelant,
-données de l'utilisateur.

5) REFUSE.

-identificateur de la connexion,
-indicateur de maintien ou non de la connexion transport,
-unités fonctionnelles supportées,
-numéro de version du protocole,
-raison du rejet.

6) FINISH.

-indicateur de maintien ou non de la connexion transport,
-données de l'utilisateur.

7) DISCONNECT.

-données de l'utilisateur.

8) NOT FINISHED.

-données de l'utilisateur.

9) ABORT.

-indicateur de maintien ou non de la connexion transport,
-informations sur l'implémentation,
-données de l'utilisateur.

10) ABORT ACCEPT : pas de paramètre.

11) DATA TRANSFER.

-indicateur de fin du SSDU,
-données.

12) EXPEDITED.

-données de l'utilisateur.

13) TYPED DATA.

-indicateur de fin du SSDU,
-données de l'utilisateur.

14) CAPABILITY DATA.

-données de l'utilisateur.

15) CAPABILITY DATA ACK.

-données de l'utilisateur.

16) GIVE TOKENS.

-jetons cédés,
-données de l'utilisateur.

17) PLEASE TOKENS.

-jetons demandés,
-données de l'utilisateur.

18) GIVE TOKENS CONFIRM.

-données de l'utilisateur.

19) GIVE TOKENS ACK : pas de paramètre.

20) MINOR SYNC POINT.

-indicateur de demande de confirmation explicite,
-numéro du point,
-données de l'utilisateur.

21) MINOR SYNC ACK.

-numéro du point confirmé,
-données de l'utilisateur.

22) MAJOR SYNC POINT.

-paramètre indiquant si ce point termine une activité,
-numéro du point,
-données de l'utilisateur.

23) MAJOR SYNC ACK.

-numéro du point confirmé,
-données de l'utilisateur.

24) RESYNCHRONIZE.

-propositions sur la position des jetons et sur leur numéro,
-option de resynchronisation,
-données de l'utilisateur.

25) RESYNCHRONIZE ACK.

-réinitialisation des jetons,
-données de l'utilisateur.

26) PREPARE.

-type de SPDU à attendre.

27) EXCEPTION REPORT.

-informations sur le SPDU reçu avec une erreur de protocole.

28) EXCEPTION DATA.

-raison d'émission,
-données de l'utilisateur.

29) ACTIVITY START.

-identificateur d'activité,
-données de l'utilisateur.

30) ACTIVITY RESUME.

-informations relatives à l'activité reprise,
-nouvel identifiant,
-données de l'utilisateur.

31) ACTIVITY INTERRUPT.

-raison de l'interruption,
-données de l'utilisateur.

32) ACTIVITY INTERRUPT ACK.

-données de l'utilisateur.

33) ACTIVITY DISCARD.

-raison,
-données de l'utilisateur.

34) ACTIVITY DISCARD ACK.

-données de l'utilisateur.

35) ACTIVITY END.

-numéro du point de synchronisation majeur,
-données de l'utilisateur.

36) ACTIVITY END ACK.

-numéro du point de synchronisation confirmé,
-données de l'utilisateur.

ANNEXE 15 : STRUCTURES D'INFORMATIONS DE LA COUCHE SESSION D'ISODE.

1) SSAPDATA.

```
struct SSAPdata(
int          sx_type  : manière dont les données sont reçues (S-DATA-
                        INDICATION, S-TYPED-DATA-INDICATION),
int          sx_cc    : nombre total d'octets,
struct qbuf sx_qbuf : données).
```

2) SSAPTOKEN.

```
struct SSAPtoken(
int      st_type    : type d'indication survenue (ex : S-TOKEN-GIVE-
                      INDICATION),
u_char st_tokens : jetons demandés ou cédés,
u_char st_owned : tous les jetons actuellement possédés par l'utilisateur,
char   st_data    : données associées,
int      st_cc      : longueur de ces données).
```

3) SSAPSYNC.

```
struct SSAPsync(
int   sn_type     : type d'événement survenu,
int   sn_options : dans le cas de la synchronisation mineure, indique si
                   une confirmation explicite est ou non demandée),
int   sn_ssn      : numéro du point de synchronisation associé,
int   sn_settings : remise des jetons utile pour la resynchronisation,
char  sn_data     : données associées,
int   sn_cc       : longueur des données).
```

4) SSAPACTIVITY.

```
struct SSAPindication(
int                 sv_type    : type d'événement survenu,
struct SSAPactid sv_id      : identifiant d'activité,
struct SSAPactid sv_oid     : ancien identifiant d'activité (pour le
                              S-ACTIVITY-RESUME),
long                sv_ssn     : numéro du point de synchronisation associé,
int                 sv_reason : raison de l'abandon ou de l'interruption de
                                l'activité,
char                sv_data    : données associées,
int                 sv_cc      : longueur des données).
```

5) SSAPREPORT.

```
struct SSAPreport(
int     sp_peer   : différenciation entre rapport de l'utilisateur et du
                    fournisseur,
int    sp_reason : raison du rapport,
char sp_data     : données de l'utilisateur,
int    sp_cc     : longueur des données).
```

6)SSAPFINISH.

```
struct SSAPfinish(
char sf_data : données finales,
int    sf_cc   : longueur des données).
```

7) SSAPABORT.

```
struct SSAPabort(
int    sa_peer   : différenciation entre abandon du fournisseur et de
                   l'utilisateur,
int    sa_reason : raison de l'abandon par le fournisseur,
char sa_data     : données
int    sa_cc     : longueur des données).
```

ANNEXE 16 : ELEMENTS DE SERVICE DE LA COUCHE TRANSPORT.

Sont repris ici les arguments des éléments de service de la couche transport.

1) T-CONNECT.
-adresse appelée,
-adresse appelante,
-paramètre indiquant si le service de données exprès est utilisé ou non,
-qualité de service,
-données initiales.

2) TRANSFERT (normal et exprès).
-données envoyées.

3) ABANDON DE CONNEXION.
-raison de l'abandon,
-données de l'utilisateur.

ANNEXE 17 : TPDU DU PROTOCOLE DE TRANSPORT.

Voici résumée la composition des TPDU.

1) CR

-crédit : utile pour le contrôle de flux,
-identifiant de la connexion chez l'initiateur et le répondeur (dans ce dernier cas, la valeur est égale à 0),
-classe de transport proposée et des options associées,
-adresse complète de l'appelant et de l'appelé,
-taille maximale proposée des TPDU,
-numéro de version du protocole,
-paramètre de protection : dépend de l'utilisateur,
-checksum : pour le contrôle d'erreurs dans la classe 4,
-options supplémentaires,
-classe de protocoles alternatifs,
-temps de réponse maximum,
-débit,
-taux d'erreurs résiduelles,
-priorité,
-délai de transit,
-temps de réassignement permis pour récupérer une erreur,
-données de l'utilisateur.

2) CC

Le contenu du CC TPDU est le même que celui du CR TPDU (à l'exception du paramètre concernant les classes alternatives).

3) DR

-identificateur de connexion chez l'initiateur et le répondeur,
-raison de l'abandon,
-informations additionnelles,
-checksum,
-données de l'utilisateur.

4) DC

-identificateur de connexion,
-checksum.

5) DT

-identificateur de connexion,
-paramètre pour signifier si le TPDU est ou non la fin d'un TSDU,
-numéro du TPDU,
-checksum,
-données.

6) ED

-identificateur de connexion,
-numéro du TPDU,
-checksum,
-données.

7) AK

-valeur du crédit,
-identificateur de connexion,
-numéro de séquence du dernier TPDU reçu,
-checksum,
-numéro de sous-séquence pour s'assurer que ces AK TPDU sont générés dans le bon ordre.

8) EA

-identificateur de connexion,
-numéro de l'ED TPDU confirmé,
-checksum.

9) RJ

-valeur du crédit,
-identificateur de connexion,
-numéro du prochain TPDU à partir duquel la transmission devrait être reprise.

10) ER

-identificateur de connexion,
-cause du rejet,
-TPDU invalide,
-checksum.

ANNEXE 18 : STRUCTURES D'INFORMATIONS DE LA COUCHE DE TRANSPORT D'ISODE.

1) TSAPSTART.

```
struct TSAPstart(
int                 ts_sd          : descripteur de transport pour identifier
                                     la connexion,
struct TSAPaddr ts_calling         : adresse de l'initiateur,
struct TSAPaddr ts_called          : adresse du destinataire,
int                 ts_expedited   : indication si le transport exprès est
                                     utilisé,
int                 ts_tsdusize    : plus grande taille atomique admise des
                                     TSDU,
char                ts_data        : données de l'utilisateur,
int                 ts_cc          : taille des données).
```

2) TSAPDISCONNECT.

```
struct TSAPdisconnect(
int    td_reason : raison de l'abandon,
char td_data     : données finales,
int    td_cc     : taille des données).
```

3) TSAPCONNECT.

```
struct TSAPconnect(
int                 tc_sd          : descripteur de connexion,
struct TSAPaddr tc_responding      : adresse répondante,
int                 tc_expedited   : indication si le transfert exprès est
                                     utilisé,
struct QOStype   tc_qos            : qualité de service,
char                tc_data        : données de l'utilisateur,
int                 tc_cc          : longueur des données).
```

4) TSAPDATA.

```
struct TSAPdata(
int             tx_expedited : si les données sont arrivées ou non par le
                               transport exprès,
                tx_cc        : nombre d'octets lus,
struct qbuf  tx_qbuf         : données reçues).
```