RÉSEAUX INFORMATIQUES

SUPPORT DE COURS
E3i, 2001-2002
Université de Tours

Michel Crucianu

École d'Ingénieurs en Informatique pour l'Industrie
64, avenue Jean Portalis
37200 TOURS

Table des matières

1. Réseaux téléinformatiques et modèle OSI

Intérêt des réseaux téléinformatiques :

1° Le partage des ressources matérielle et logicielles, des données.
2° La fiabilité du système d'information.
3° L'augmentation graduelle des ressources matérielles et logicielles.
4° La communication entre utilisateurs distants et/ou applications distantes.
5° La collaboration entre utilisateurs distants (*groupware*, par exemple Lotus Notes).

Types de réseaux en fonction de l'aire desservie :

1° Réseaux locaux (*Local Area Networks*, LAN) : 10 m ÷ 1 km.
2° Réseaux métropolitains (*Metropolitan Area Networks*, MAN) : 1 km ÷ 100 km.
3° Réseaux très longue distance (*Wide Area Networks*, WAN) : 100 km ÷ 10 000 km.

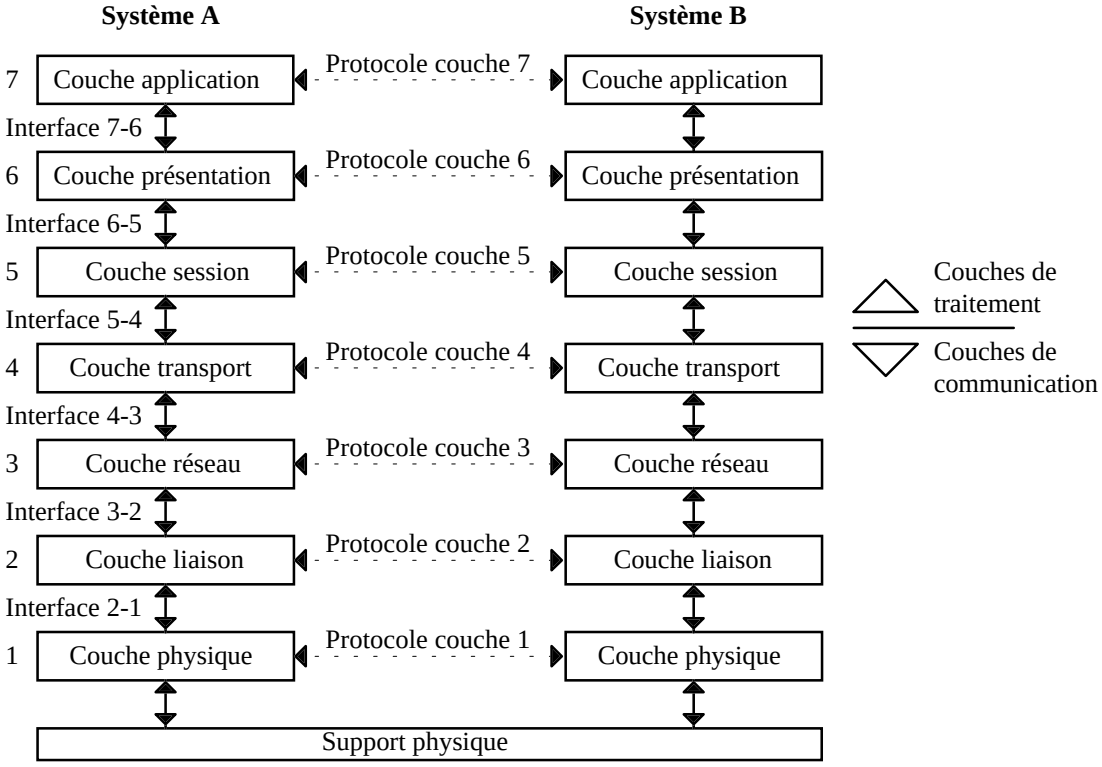
Les principes qui ont mené au modèle OSI (*Open Systems Interconnection*) à 7 couches :

1° Une couche doit être créée lorsqu'un nouveau niveau d'abstraction est nécessaire.
2° Chaque couche exerce une fonction bien définie.
3° Les fonctions de chaque couche doivent être choisies en pensant à la définition de protocoles normalisés internationaux.
4° Le choix des frontières entre couches doit minimiser le flux d'informations aux interfaces.
5° Le nombre de couches doit être assez grand pour que des fonctions très différentes ne cohabitent pas dans une même couche et suffisamment réduit pour que l'architecture soit maîtrisable.

Pourquoi un modèle en couches ?

1° Facilité de développement et de modification : une couche (un protocole) peut être modifiée de façon indépendante tant que l'interface avec les deux couches adjacentes reste inchangée.
2° Interopérabilité : une même couche de niveau *n+1* peut utiliser les services de couches de niveau *n* très différentes à condition que l'interface *n/n+1* soit la même.

Architecture OSI :



Termes employés : *interface* entre couches adjacentes, *protocole* entre processus *pairs* (de même niveau).

Activités possibles dans chaque couche *n* :

1° Identifier l'interlocuteur : au même niveau, aux niveaux adjacents.
2° Établir/relâcher la connexion.
3° Définir le mode simplex/semi-duplex/duplex. Définir le nombre de canaux par connexion.

4° Effectuer éventuellement :

La segmentation/le regroupement des messages : la taille des messages reçus de la couche *n+1* est supérieure à la taille utilisable par la couche *n*.

La concaténation/la séparation des messages : la taille des messages reçus de la couche *n+1* est inférieure à la taille utilisable par la couche *n*.

Le multiplexage/l'éclatement des messages : la capacité d'une liaison de niveau *n* est un multiple du débit des liaisons de niveau *n+1*.

5° Détecter et corriger des erreurs.

6° Assurer le respect de l'ordre des messages.

7° Assurer l'asservissement émetteur-récepteur.

8° Effectuer le routage des messages.

Spécifique de chaque couche OSI :

1° Physique : transmission des bits sur un support physique déterminé.

2° Liaison de données : liaison fiable point à point.

3° Réseau : acheminement des messages.

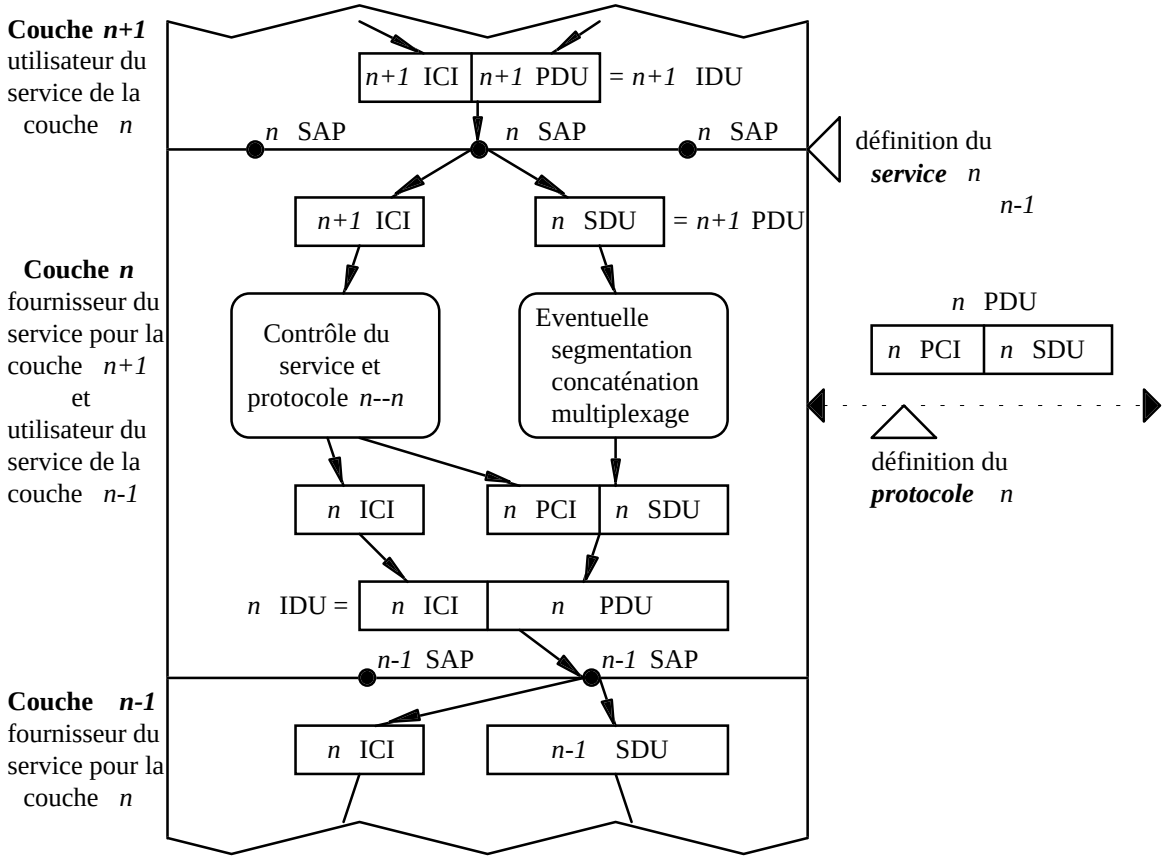
4° Transport : transport fiable de bout en bout.

5° Session : gestion du dialogue et synchronisation.

6° Présentation : syntaxe de transfert, compression, cryptage[1].

7° Application : services application génériques (terminal virtuel, transfert de fichiers, messagerie).

Architecture de principe d'un niveau (terminologie OSI) :



SAP : point d'accès au service (*Service Access Point*)

IDU : unité de données d'interface (*Interface Data Unit*)

ICI : informations de contrôle de l'interface (*Interface Control Information*)

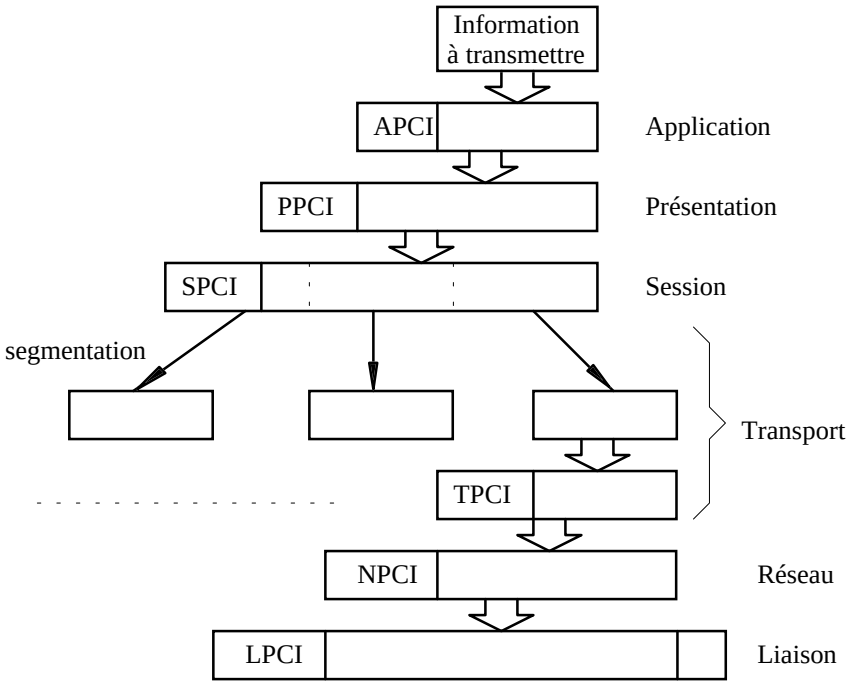
PDU : données du protocole (*Protocol Data Unit*)

PCI : informations de contrôle du protocole (*Protocol Control Information*)

SDU : données du service (*Service Data Unit*)

[1] Dans les protocoles actuels, la compression et le cryptage peuvent intervenir aussi à des niveaux plus bas dans la hiérarchie.

Encapsulation successive des informations à transmettre :



Mode de transmission :

1° Avec connexion : établissement d'une connexion avant le transfert des données. Avantages : permet de s'assurer que le destinataire peut accepter les messages ; l'ordre des messages est respecté ("tuyau"). Désavantage : durée élevée d'établissement de la connexion. Mode intéressant uniquement pour le transfert de volumes importants de données (nombre élevé de messages ordonnés).
2° Sans connexion : les données sont envoyées sans qu'une connexion soit préalablement établie. L'ordre des messages n'est pas nécessairement respecté. Mode utilisable sur des réseaux à voie unique (l'ordre des messages est maintenu grâce à la structure du réseau) ou pour des messages individuels (l'ordre n'a aucune importance).

Qualité de service – paramètre de fiabilité :

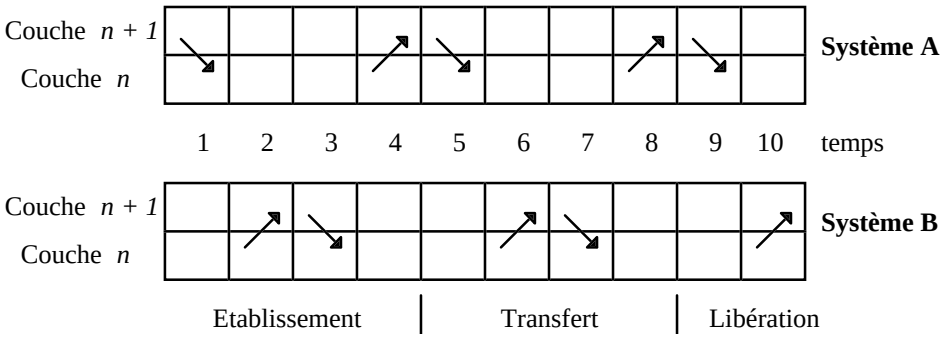
1° Service fiable : aucune perte de données grâce au contrôle des erreurs et à l'acquittement de chaque message (exemple : transfert de fichiers). Entraîne des délais supplémentaires.
2° Service non fiable : les erreurs ne sont pas détectées, il n'y a pas d'acquittement pour les messages (exemple : téléphone).

Le mode connecté et le service fiable ne sont en général pas utilisés dans toutes les couches ; si le support est très fiable, le contrôle des erreurs peut être effectué uniquement pour le transfert de bout en bout (au niveau transport).

L'accès à un service utilise des primitives de service :

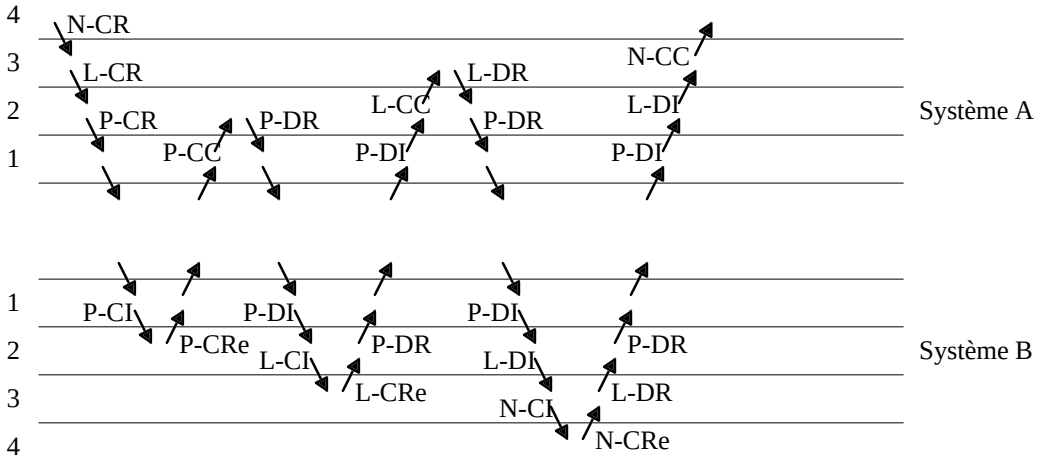
1° Requêtes (*request*) : une entité sollicite un service pour une activité.
2° Confirmation (*confirm*) : une entité est informée de sa demande de service.
3° Réponse (*response*) : une entité répond à un événement.
4° Indication (*indication*) : une entité est informée d'un événement.

Transfert de données *n+1* ↔ *n+1* en mode connecté :



1°: CONNECT.request
2°: CONNECT.indication
3°: CONNECT.response
4°: CONNECT.confirm
5°: DATA.request
6°: DATA.indication
7°: DATA.request
8°: DATA.indication
9°: DISCONNECT.request
10°: DISCONNECT.indication

Etablissement d'une connexion entre deux couches transport (modèle OSI) :



CR : CONNECT.request
CI : CONNECT.indication
CRe : CONNECT.response
CC : CONNECT.confirm
DR : DATA.request
DI : DATA.indication

Service confirmé : demande, indication, réponse et confirmation. Service non confirmé (ne pas confondre fiabilité et confirmation) : uniquement demande et indication. En général, un service DATA n'a pas besoin de confirmation, alors qu'un service CONNECT doit être toujours confirmé (les deux entités entre lesquelles s'établit la connexion doivent se mettre d'accord sur les paramètres de la connexion).

Transmission d'informations — deux exemples :

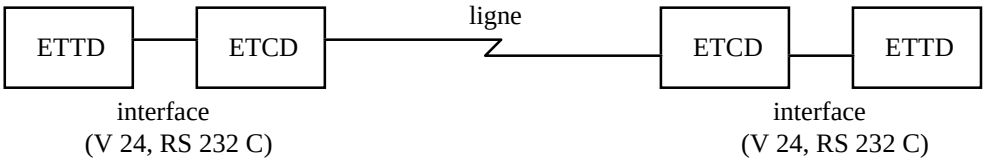
1° Envoi de courrier électronique par un ordinateur relié au RTC à un ordinateur relié à un réseau local.

2° Multiplexage voix/données sur RNIS classique.

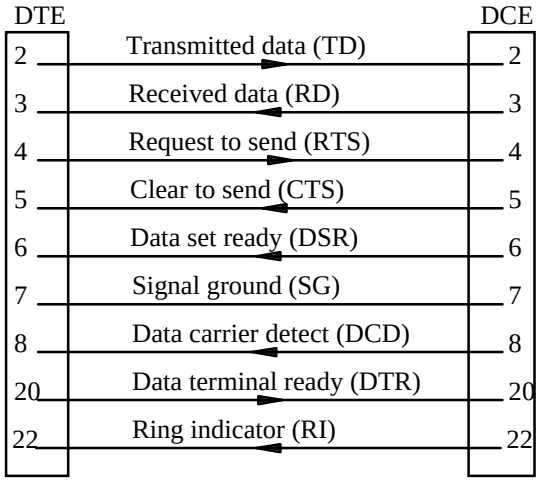
2. La couche physique

2.1. Interfaces V 24 / V 28 (CCITT) et RS 232 C (pour info…)

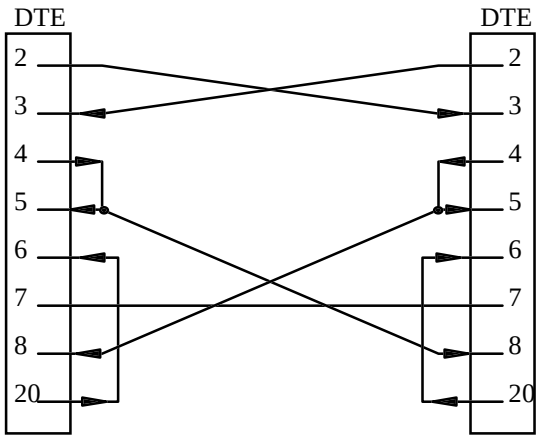
ETTD (Equipement Terminal de Transmission de Données) ou DTE (*Data Terminal Equipment*)
ETCD (Equipement Terminal de Circuit de Données) ou DCE (*Data Circuit Equipment*)



Interfaçage minimal ETTD ↔ ETCD (V 24, RS 232 C), les numéros correspondent au connecteur DB 25 :



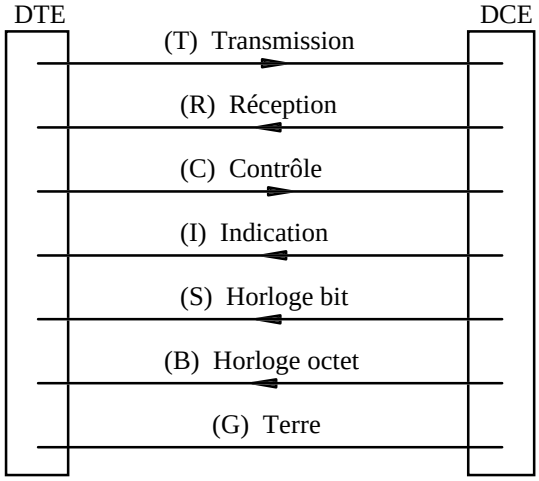
Les 9 signaux sont en général véhiculés à travers un connecteur DB 9.
Interfaçage directe ETTD ↔ ETTD via port série (PC ↔ PC, PC ↔ imprimante) = *null-modem* :



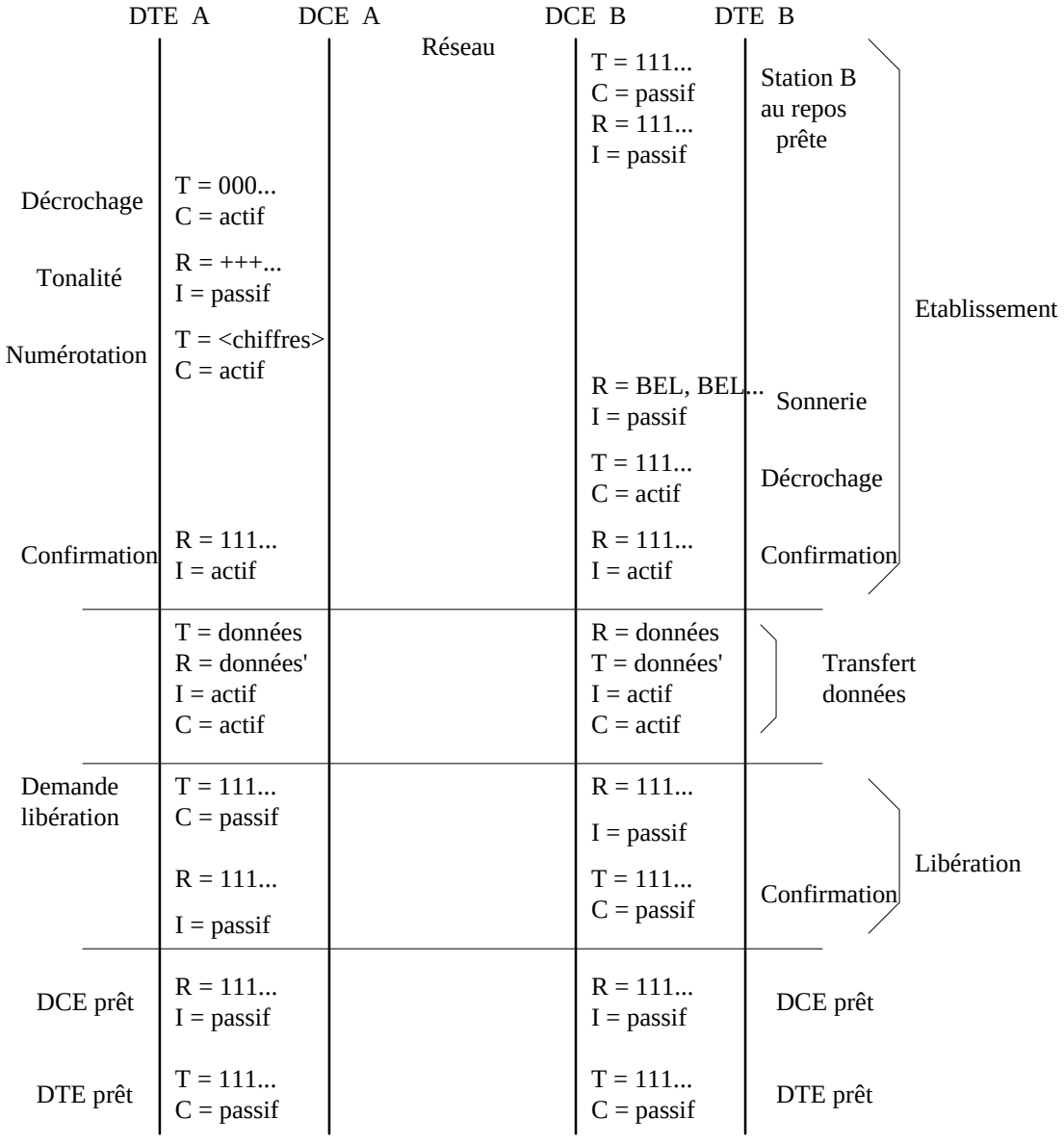
Appel automatique : V 25 / 25 bis ; plus utilisées sont les commandes Hayes.

2.2. Interface X21

Interfaçage ETTD ↔ ETCD selon X 21 (interface **synchrone** préconisée pour Transpac) :



Dialogue X 21 :



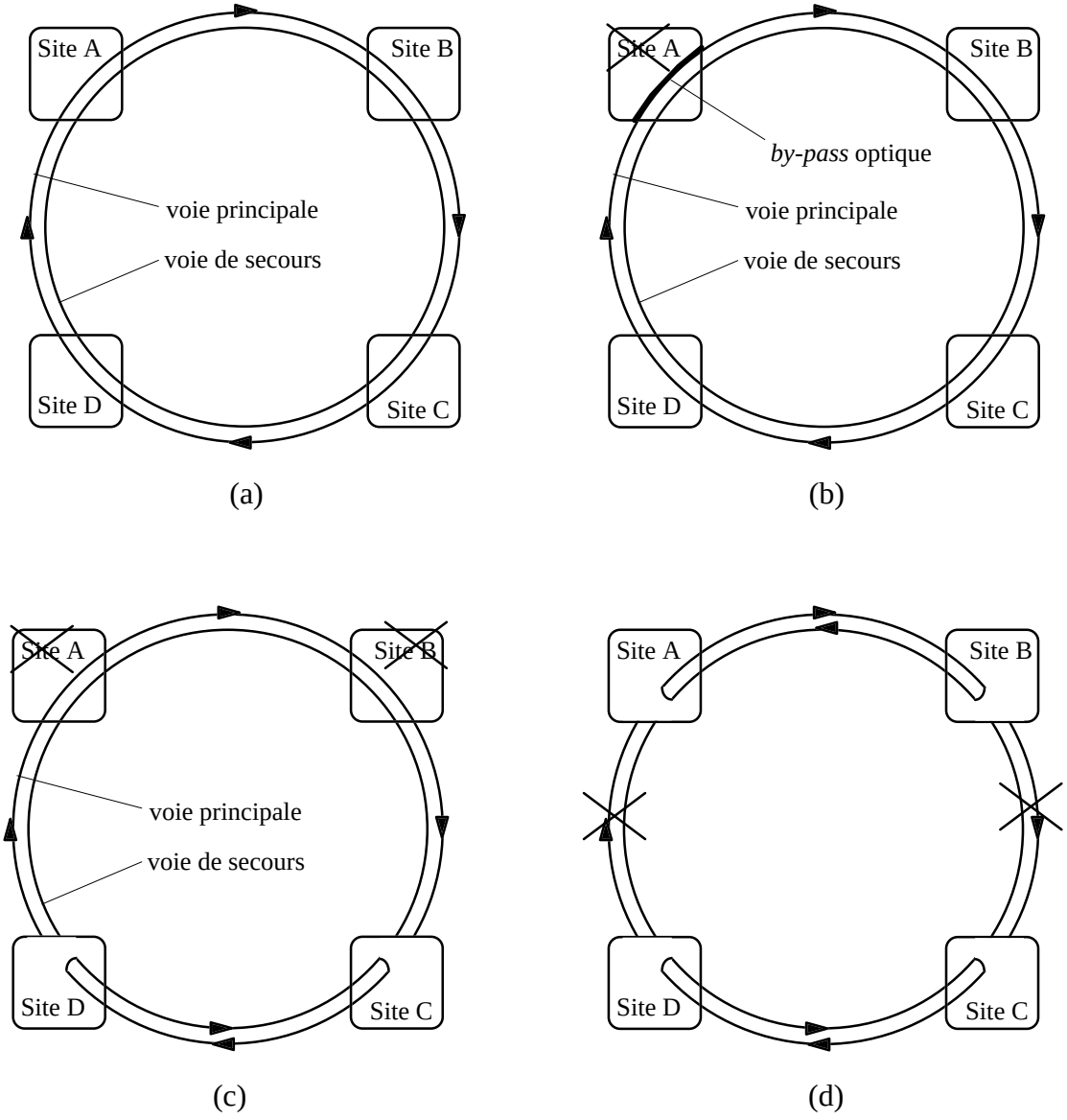
X 21 bis : X 21 à travers V 24.

2.3. La couche physique de FDDI

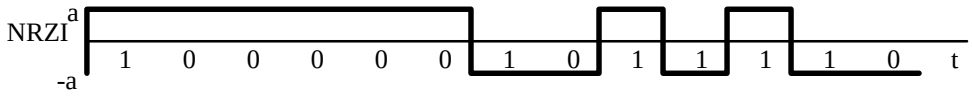
Fiber Distributed Data Interface (FDDI) est une technologie développée pour les réseaux métropolitains. La norme FDDI couvre les deux premiers niveaux de la hiérarchie OSI : la couche physique et la couche liaison. Pour la couche liaison, FDDI s'inspire de la définition de l'anneau à jeton (IEEE 802.5) qui est à la base des réseaux locaux Token Ring.

En ce qui concerne la couche physique, FDDI fait appel à une structure en double anneau (a) en fibre optique multimode 62,5/125 µm, à gradient d'indice (des fibres 50/125 µm, 85/125 µm ou 100/140 µm peuvent également être employées). La longueur d'onde du signal lumineux est de 1300 nm. Avec la fibre préconisée, la longueur maximale de l'anneau est de 200 km et la distance maximale entre deux noeuds successifs de 2 km.

Chaque station sur le réseau attend son tour pour émettre. Les trames émises sont copiées par le(s) destinataire(s) mais continuent leur chemin jusqu'à l'émetteur, qui est chargé de les éliminer. Si un noeud FDDI tombe en panne (ou n'est plus alimenté), un *by-pass* optique passif (micro-mirroir pivotant) prend la relève (b). Si deux noeuds successifs tombent en panne, le signal sur ce segment (devenu purement passif) est trop atténué pour que la boucle FDDI puisse être maintenue ; dans ce cas, la voie de secours est utilisée pour reconstituer la boucle (c). Dans certains cas, des pannes multiples provoquent une reconfiguration avec disparition de la boucle unique à la faveur de plusieurs boucles distinctes (d).



Le débit binaire préconisé est de 100 Mbit/s. La transmission est en bande de base (aucune modulation). Pour limiter la fréquence d'horloge employée, un codage NRZI est utilisé (une transition est présente pour un "1", aucune transition pour un "0") :



Afin de garantir la synchronisation entre l'horloge de l'émetteur et celui du récepteur, les groupes de plus de 3 bits successifs à "0" sont éliminés de la façon suivante : tous les groupes de 4 bits successifs de données sont traduits, avant le codage NRZI, en groupes de 5 bits ayant au maximum 3 bits successifs à "0" (codage 4B/5B). Sur les 32 combinaisons binaires possibles sur 5 bits, 8 sont employées pour la signalisation, 16 pour les données (correspondant aux 16 combinaisons binaires possibles avec 4 bits) et les 8 combinaisons restantes sont déclarées illégales. Le tableau suivant présente, pour les configurations légales, la correspondance entre l'information à transmettre et le groupe de 5 bits associé :

Code signalisation	Code 5B	Donnée	Code 5B	Donnée	Code 5B
I*dle*	11111	0000	11110	1000	10010
J (début)	11000	0001	01001	1001	10011
K (début)	10001	0010	10100	1010	10110
R (*reset*)	00111	0011	10101	1011	10111
S (*set*)	11001	0100	01010	1100	11010
Q*uiet*	00000	0101	01011	1101	11011
H*alt*	00100	0110	01110	1110	11100
T (fin)	01101	0111	01111	1111	11101

Signification des symboles de signalisation (voir la trame FDDI plus bas) :

Idle = symboles émis en continu (en l'absence de trames à envoyer) sur la liaison, doivent maintenir la synchronisation entre les horloges

J = premier délimiteur de début de trame

K = second délimiteur de début de trame

Reset = indique une condition logique "off" ou "faux"

Set = indique une condition logique "on" ou "vrai"

Quiet = indique l'absence de transitions sur la fibre (situation anormale)

Halt = séquence de contrôle couche physique

T = délimiteur de fin

Trame FDDI (niveau physique) :

Une trame physique débute par 16 symboles Idle (normalement, en l'absence de trames, des symboles Idle circulent en permanence sur le support). Le délimiteur de début est composé d'une séquence JK. Ce délimiteur est suivi par l'en-tête liaison FDDI et les éventuelles données de niveau liaison. La trame physique se termine par un symbole T, un champ E (erreur détectée dans la trame) et un nombre variable de paires de champs A (adresse du destinataire reconnue) et C (trame copiée par le destinataire). Les champs E, A et C peuvent prendre comme valeur les symboles R et S. L'émetteur positionne trois champs à la valeur R, les stations successives peuvent repositionner les champs à la valeur S ou en ajouter d'autres (positionnés). En effet, si le destinataire est unique, une seule paire AC est présente, sinon chaque destinataire ajoute sa propre paire AC. Toutes les stations traversées par la trame effectuent une détection d'erreurs et peuvent positionner le champ E. Si l'émetteur (chargé de retirer sa trame de l'anneau) constate que le champ E a la valeur S et au moins un des destinataires a laissé son champ C la valeur R, il sait qu'il doit reémettre la trame (l'erreur s'est produite avant la recopie par un des destinataire). Si E a la valeur S, mais tous les champs A et C sont positionnés à S (et leur nombre correspond au nombre de destinataires), l'émetteur sait que l'erreur s'est produite après la recopie correcte de la trame par tous les destinataires (sur le chemin de retour), donc la reémission est inutile. Si les champs A et C sont absents, un second T doit être présent.

3. La couche liaison de données

L'accès au support physique d'une liaison point à point est partagé uniquement par les deux interlocuteurs et le contrôle de cet accès pose peu de problèmes. En revanche, le support physique d'un réseau à diffusion est partagé par un nombre très important de stations et donc le contrôle de l'accès est complexe (les réseaux à diffusion sont employé principalement pour les réseaux locaux, mais aussi pour certains réseaux métropolitains, pour les communications par satellite et pour des réseaux de communications radio). Une sous-couche spécifique de la couche liaison de données — appelée MAC (*Medium Access Control*) — regroupe ces fonctions de contrôle d'accès pour les réseaux à diffusion (sous-couche absente pour les liaisons point à point).

3.1. Réseaux à diffusion — sous-couche MAC

Caractéristique générale des protocoles : le contrôle est décentralisé ; quand un rôle de gestionnaire — à attributions limitées — existe (protocoles à jeton), il peut être rempli par une machine quelconque.

3.1.1. Allocation statique des canaux

Possibilités :

1° Multiplexage en fréquence : une bande de fréquences est réservée à l'émission de chaque intervenant. Désavantages : faible efficacité (le taux d'utilisation de chaque canal est en général faible) ; délais d'attente proportionnels au nombre de stations (pour une même bande passante totale du support) ; les stations doivent posséder un récepteur pour chaque bande...

2° Multiplexage temporel statique : un intervalle temporel est réservé à l'émission de chaque intervenant. Désavantages : faible efficacité (un nombre important d'intervalles temporels restent inutilisés) ; délais d'attente proportionnels au nombre de stations (pour une même bande passante totale du support) ; les stations doivent être capables de se synchroniser afin de détecter leurs intervalles temporels respectifs.

3.1.2. Allocation dynamique des canaux

Le droit à émettre est accordé en fonction des demandes formulées par les stations, il n'y a pas de réservation permanente. A priori plus efficace que l'allocation statique, mais la résolution des conflits d'accès utilise un pourcentage de la bande passante disponible...

3.1.2.1. Protocoles à collisions

3.1.2.1.1. Protocoles ALOHA

Le premier protocole ALOHA (pur ALOHA) a été développé à l'Université de Hawaï pour permettre la communication par ondes radio entre des sites répartis sur plusieurs îles et un site central. Le protocole est très simple : chaque station émet sans se soucier des autres stations ; les messages qui entrent en collision ne sont pas acquittés par la station réceptrice, donc chaque émetteur sait quels messages il doit réémettre. Toutefois, avant de réémettre, une station observe un délai aléatoire afin minimiser le risque de nouvelle collision avec la (les) même(s) station(s). Si S est le nombre moyen de trames correctement transmises par durée de trame et G le nombre moyen de tentatives de transmission de trame par durée de trame, pour le protocole pur ALOHA nous avons la relation $S = G \cdot e^{-2G}$. La valeur maximale $S = 1/2e \cong 0{,}184$ est atteinte pour $G = 0{,}5$.

L'efficacité du protocole à été multipliée par 2 grâce à une idée simple (qui en revanche n'est pas applicable dans tous les cas) : les émissions de toutes les stations sont synchronisés par un signal unique "début d'émission" (d'où le nom d'ALOHA discrétisé) ; la synchronisation permet de réduire de moitié l'intervalle de vulnérabilité. La relation entre S et G devient $S = G \cdot e^{-G}$. La valeur maximale $S = 1/e \cong 0{,}368$ est atteinte pour $G = 1$.

3.1.2.1.2. Protocoles CSMA

Dans certains cas — comme celui des réseaux locaux sur bus — les stations peuvent "écouter" tout ce qui est transmis sur le canal et peuvent donc attendre pour émettre que le canal soit disponible. Une telle technique porte le nom de CSMA (*Carrier Sense Multiple Access*) — accès multiple avec écoute de la porteuse — et permet de réduire considérablement le nombre de collisions. Dans le protocole CSMA 1-persistant une station qui écoute le canal essaie de transmettre dès que le canal devient disponible ; plusieurs stations ayant des trames à émettre peuvent donc essayer d'émettre en même temps et donc une collision est probable. Si une collision se produit, chaque station attend un intervalle aléatoire avant de réessayer. Dans le protocole CSMA non persistant, une station qui désire émettre écoute le canal ; si le canal est disponible la station essaie d'émettre, sinon elle attend un intervalle aléatoire avant d'écouter à nouveau le canal dans le but d'émettre ; une telle politique minimise le risque des collisions, en revanche les délais de transmission sont en moyenne plus importants. Dans les protocoles CSMA *p*-persistants, quand le canal est libre la probabilité pour que la station émette est *p* et la probabilité pour qu'elle attende l'intervalle suivant est 1-*p* (et ainsi de suite) ; plus *p* est réduit, plus la probabilité de collisions diminue mais le délais moyen de transmission augmente.

Une autre amélioration consiste à arrêter une transmission dès qu'une collision est détectée et non plus à la fin de la trame ; un tel protocole porte le nom de CSMA/CD (*Collision Detection*) – à détection de collision. Un tel protocole est employé par exemple dans les réseaux Ethernet et les réseaux respectant la norme IEEE 802.3. Quelques paramètres caractéristiques à IEEE 802.3 et Ethernet : *Time slot* : temps nécessaire au signal pour parcourir deux fois la distance qui sépare les stations les plus éloignées. **La durée minimale d'une trame doit être supérieure au *time slot* pour que la détection des collisions soit possible**. En effet, après un retard de 1L la première trame arrive à l'autre bout, moment auquel la station à l'autre bout peut encore l'ignorer et émettre la sienne, plus un retard de 1L pour que la station ayant émis la première détecte la collision – car la station ayant émis la première doit être encore en émission – donc durée minimale trame > *time slot* (correspondant à 2L). L'intervalle d'attente après détection d'une collision : $n \times timeslot$, avec n tiré au sort dans l'intervalle $\left[0,2^{\min(\text{nombre collisions successives},10)}\right]$.

CSMA/CD ne permet pas d'obtenir des délais garantis, ni implémenter une gestion de priorités.

3.1.2.2. Protocoles sans collisions

3.1.2.2.1. Protocole BRAP

BRAP (*Broadcast Recognition with Alternating Priorities*) alterne des périodes de contention à durée variable avec des transmissions sans collision de trames. Pendant une période de contention, chaque station connectée possède un numéro d'ordre et émet à un moment bien défini un bit à 1 si elle désire émettre ; la station émet dès qu'elle a positionné ce bit ; après émission, les numéros subissent une permutation circulaire. Un tel protocole est utilisé dans des réseaux locaux à ondes radio (peu de stations connectées).

3.1.2.2.2. Protocole MLMA

Pour MLMA (*Multi-Level Multiple Access*) une période de contention consiste en l'émission successive par les stations qui désirent émettre des digits successifs de leur numéro (attribué une fois pour toutes) ; afin d'éviter les collisions pendant les périodes dites de contention, à chaque digit correspond un groupe de 10 bits. Après chaque digit, seules les stations dont le numéro correspond à la valeur maximale du digit peuvent continuer. De cette façon, plusieurs stations sont "élues" pendant chaque période de contention et un ordre d'émission est établi entre elles ; toutes ces stations transmettent leurs données avant une nouvelle période de contention.

3.1.2.2.3. Protocoles à jeton

Un "jeton" (une trame spécifique) permet d'accorder le droit d'émettre successivement aux différentes stations qui le demandent : la station qui entre en possession du jeton à le droit d'émettre pendant un certain temps et doit ensuite céder le jeton (émettre une trame avec le jeton). Le protocole peut être employé sur un bus (norme IEEE 802.4 pour les réseaux en milieu industriel) ou sur un anneau (*Token Ring* d'IBM ou IEEE 802.5).

Un anneau est constitué de plusieurs lignes point à point reliées bout à bout par des dispositifs qui assurent la continuité de l'anneau lorsque la station connectée au répéteur est hors service ; la mise en service/hors service d'une station a toutefois des conséquences néfastes sur les signaux transmis et engendre la retransmission du message affecté. Circulation du jeton sur un anneau : la station qui entre en possession du "jeton" (message spécifique) a le droit d'émettre pendant un certain temps et doit ensuite céder le jeton (émettre un message avec le jeton). Chaque station connectée retransmet toutes les trames qu'elle reçoit, en introduisant un **retard fixe** (1 bit pour IEEE 802.5). Des techniques de gestion de priorités et de gestion du jeton sont implémentées. L'efficacité est raisonnable et **le délai de livraison est garanti** (contrairement à l'accès CSMA/CD). Les protocoles à jeton sont malheureusement beaucoup plus complexes que CSMA/CD.

3.1.2.3. Protocoles mixtes

Les protocoles mixtes essaient de combiner les avantages des protocoles à collisions – bonne efficacité et délais réduits à faible charge – et ceux des protocoles sans collisions – bonne efficacité et délais contrôlables à charge élevée. Différentes techniques ont été proposées, nous en regarderons une seule.

Une technique qui utilise à la fois CSMA/CD et un jeton a été proposée pour l'accès à un bus : les stations essaient d'obtenir le droit à émettre comme dans CSMA/CD ; si une collision se produit, la station qui possède le jeton est la seule qui a le droit d'émettre dans l'intervalle qui suit la détection de la collision. A faible charge il n'y a quasiment pas de collision et le jeton intervient peu ; à charge élevée le jeton permet d'avoir des délais de transmission assurés.

3.2. La couche liaison

La couche liaison assure la transmission fiable entre des machines connectées au même support physique. Le contrôle de l'accès au support est simple pour les connexions point à point et est assuré par une sous-couche spécifique (MAC) dans les réseaux à diffusion.

3.2.1. Types de services offerts à la couche réseau

Les types de services prévus pour la couche liaison :

1° Service sans connexion et sans acquittement : utilisé quand le support est très fiable et le contrôle des erreurs est assuré par une couche supérieure ou quand le trafic est isochrone (par exemple transmission de signaux téléphoniques).

2° Service sans connexion et avec acquittement : plus fiable que le précédent (la livraison est assurée), mais des trames peuvent être dupliquées et/ou peuvent arriver dans un mauvais ordre.

3° Service avec connexion : canal fiable entre deux machines offert à la couche réseau ; livraison garantie, pas de duplication de trames, respect de l'ordre.

3.2.2. Fonctions de la couche liaison

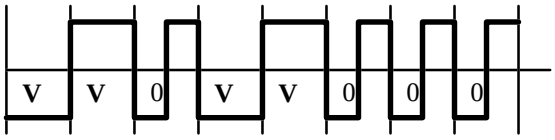
3.2.2.1. Séparation des trames

La couche liaison utilise les services offerts par la couche physique. Souvent la couche physique transmet une succession continue de bits — les intervalles entre les trames sont remplis par des séquences dont le but est de maintenir la synchronisation entre les horloges des interlocuteurs — sans connaître leur signification. Le découpage en trames revient alors à la couche liaison. Différentes techniques sont employées dans ce but :

1° Utilisation de caractères spécifiques de début et de fin de trame (protocoles orientés caractère, par exemple BSC). Quand dans les données à transmettre on rencontre un caractère utilisé pour marquer le début ou la fin d'une trame, l'émetteur doit insérer avant un caractère spécial (DLE pour le protocole BSC), caractère que le récepteur élimine (si DLE est rencontré dans les données, il sera doublé).

2° Utilisation de fanions (séquences particulières de bits) de début et de fin de trame (protocoles orientés bit, comme SDLC ou HDLC). Le fanion employé par HDLC est 0111 1110 ; quand une succession de 5 bits à 1 apparaît dans les données à transmettre, l'émetteur insère un bit à 0 et le récepteur l'élimine.

3° Violation du codage employé par la couche physique (par exemple l'anneau à jeton IEEE 802.5). La convention de codage Manchester ou Manchester différentiel (pour IEEE 802.5) n'est pas respectée, ce qui permet au récepteur de détecter le début et la fin d'une trame : pour IEEE 802.5, par exemple, c'est la séquence VV0VV000 (V = bit de viol) qui marque le début de trame



La plupart des protocoles utilisent aussi un champ longueur de trame, afin de déterminer de façon plus fiable la fin d'une trame.

3.2.2.2. Contrôle des erreurs

La transmission fiable est assurée par l'utilisation conjointe de codes détecteurs d'erreurs et d'un mécanisme d'acquittement et de retransmission en cas d'erreur. La retransmission se produit non seulement en cas de réception d'acquittement négatif, mais aussi en cas de non réception d'acquittement après un délai préétabli.

3.2.2.3. Contrôle de flux

Les ressources (place mémoire pour le stockage, temps de calcul) allouées par le récepteur à une communication étant limitées, un mécanisme de synchronisation entre l'émetteur et le récepteur doit être présent afin d'éviter la perte de trames.

3.2.2.4. Gestion de la liaison

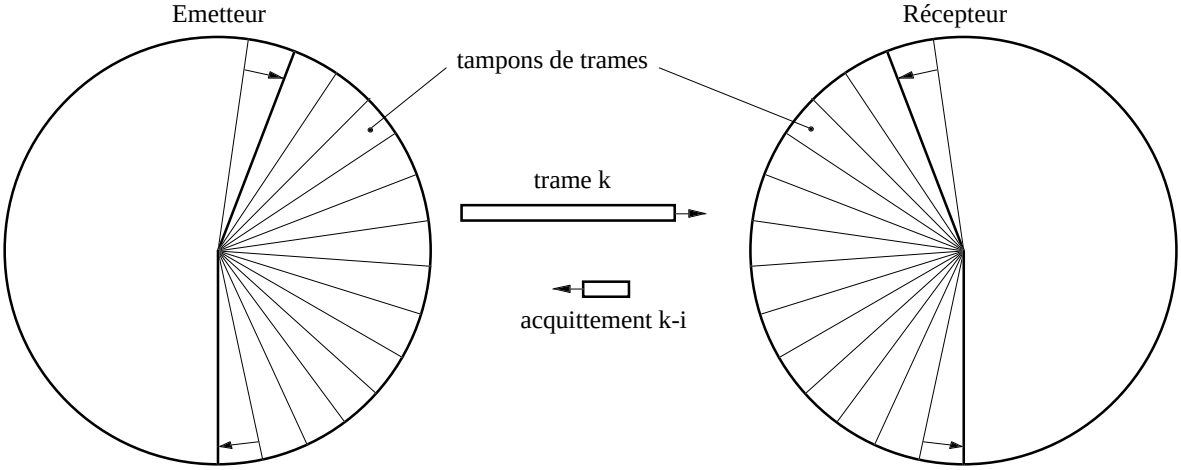
La gestion est minimale pour un service sans connexion, mais peut comporter différentes activités pour un service avec connexion : choix du sens de transmission, choix des paramètres de communication, initialisation de numéros de séquence et de la taille de la fenêtre, réinitialisation de la connexion en cas d'erreur, etc.

3.2.3. Protocole générique à fenêtre d'anticipation et rejet sélectif

Un tel protocole offre un service fiable, en mode connecté. Les trames sont numérotées en utilisant *n* bits de l'entête. L'émetteur et le récepteur possèdent chacun une "fenêtre" (de taille inférieure à 2^{n-1}) dont le fonctionnement est le suivant :

La fenêtre d'émission contient les trames émises pour lesquelles des acquittements positifs n'ont pas encore été reçus. Chaque trame dans la fenêtre est gardée dans un tampon en mémoire. Dès qu'un acquittement positif pour une trame *m* est reçu, toutes les trames dont le numéro est inférieur ou égal à *m* sont considérées acquittées (avantage : si un acquittement positif est perdu, la réception d'une trame sera confirmée par l'acquittement d'une trame suivante) et la borne inférieure de la fenêtre est déplacée.

Quand la couche réseau demande l'émission d'une nouvelle trame, la borne supérieure de la fenêtre est déplacée afin d'inclure celle-ci, mais uniquement si la taille maximale définie pour la fenêtre n'est pas dépassée. Chaque tampon de trame garde aussi un compteur initialisé au moment où la trame est émise et décrémenté ensuite périodiquement ; si le compteur atteint 0 alors que la trame est encore dans la fenêtre d'émission, la trame est émise à nouveau. Aussi, une trame (dans la fenêtre) est émise à nouveau dès qu'un acquittement négatif portant le numéro de la trame est reçu.



La fenêtre de réception (dont la taille est égale à la taille maximale de la fenêtre d'émission afin d'assurer la synchronisation entre l'émetteur et le récepteur) contient les trames reçues et non encore transmises à la couche réseau ainsi que les tampons encore vides. Chaque trame reçue est vérifiée (détection des erreurs) et son numéro est comparé avec les numéros qui définissent la fenêtre ; si son numéro ne correspond pas à la fenêtre ou si la trame est déjà présente dans la fenêtre, la trame est rejetée ; sinon, si une erreur est trouvée le récepteur envoie un acquittement négatif avec le numéro de la trame à l'émetteur ; sinon la trame est gardée dans un tampon de la fenêtre et des acquittements négatifs sont transmis pour chaque trame dont le numéro se trouve encore dans la fenêtre et est inférieur au numéro de la trame gardée (à condition qu'un acquittement négatif n'ait pas été déjà envoyé pour une même trame — la perte d'un acquittement négatif n'est pas très importante grâce aux compteurs maintenus par l'émetteur). Dès que la trame qui porte le numéro inférieur de la fenêtre est disponible dans un tampon, une indication est transmise à la couche réseau pour la récupérer ; après cette opération un tampon devient disponible et les deux bornes de la fenêtre sont augmentées de 1. L'envoi d'un acquittement positif est conditionné, d'une part, par la réception correcte de toutes les trames dont le numéro est inférieur **et**, d'autre part, **par la récupération de toutes ces trames par la couche réseau du récepteur**.

Pour chacune des deux fenêtres, le déplacement de la borne inférieure conditionne le déplacement de la borne supérieure. La borne inférieure de la fenêtre de réception peut avancer dès que la trame inférieure, correctement reçue, est récupérée par la couche réseau. Le déplacement de la borne inférieure de la fenêtre de l'émetteur est conditionné par la réception d'un acquittement positif concernant la trame inférieure. Si la couche réseau du récepteur ne peut pas reprendre les trames disponibles, des acquittements positifs ne peuvent pas être envoyés à l'émetteur et, par conséquence, la borne inférieure de la fenêtre de l'émetteur ne peut pas avancer. Cela empêche l'évolution de la borne supérieure de la fenêtre de l'émetteur. Ce mécanisme permet donc la synchronisation entre l'émetteur et le récepteur (le contrôle de flux). Malheureusement, comme il ne reçoit aucun d'acquittement positif, l'émetteur reprend (après expiration du délai d'acquittement) l'émission des trames présentes dans la fenêtre d'émission ; le support de transmission est donc occupé inutilement.

Si des données doivent être transmises dans les deux sens, les acquittements sont "collés" aux trames de données (procédé qui porte le nom de *piggybacking*). Dans ce cas, un compteur est associé aussi aux tampons des trames de la fenêtre de réception ainsi qu'aux trames erronées : le compteur est initialisé à la réception de la trame et, s'il atteint 0 avant qu'une trame de données soit prête pour émission, l'acquittement est transmis dans une trame d'acquittement.

3.2.4. Protocole BSC (pour info…)

Le protocole *Binary Synchronous Communication* (BSC) est un protocole "envoyer et attendre" orienté caractère, en mode semi-duplex, avec connexion, exploitable en point à point ou en multipoint ; développé par

IBM principalement pour le dialogue entre un ordinateur et les terminaux. Le contrôle et la supervision sont assurés par des caractères ASCII spécifiques (codes CCITT n° 5) :

SYN (*SYNchronous idle*) — caractère de synchronisation au niveau des caractères (pour la synchronisation au niveau bit l'octet 0101 0101 est employé).

ENQ (*ENQuiry*) — indique une demande de réponse (ouverture connexion, demande de renseignements).

EOT (*End Of Transmission*) — indique la fin d'une transmission (déconnexion).

SOH (*Start Of Heading*) — indique le début de l'en-tête du message.

STX (*Start of TeXt*) — indique le début des données dans une trame.

ETB (*End of Transmission Block*) — indique la fin d'un bloc (zone de données d'une trame).

ETX (*End of TeXt*) — indique la fin du message.

ACK (*ACKnowledge*) — accusé de réception positif.

NAK (*Negative AcKnowledge*) — accusé de réception négatif.

DLE (*Data Link Escape*) — commande supplémentaire de transmission.

Le contrôle d'erreurs utilise un caractère BCC (*Block Check Character*, code de parité) ou une clé CRC ajoutée à la fin du bloc de données.

La retransmission d'une trame se fait dans deux conditions : un acquittement négatif est reçu par l'émetteur ou le délai d'attente d'un acquittement expire. Deux accusés de réception (acquittements) positifs doivent alors être utilisés, ACK0 et ACK1, afin d'éviter la retransmission d'une trame correctement reçue mais pour laquelle l'acquittement a été perdu. En cas de *n* retransmissions successives sans succès, l'émetteur envoie une demande ENQ et en cas d'absence de réponse termine la transmission avec EOT.

3.2.5. Protocoles HDLC

Plusieurs protocoles HDLC (*High level Data Link Control*) ont été normalisés par l'ISO et le CCITT. Ces protocoles sont orientés bit (contrairement à BSC qui est orienté caractère) et utilisent le fanion 01111110 pour délimiter les trames. Quand la liaison n'est pas utilisée, des fanions sont transmis sans arrêt ; l'abandon de la trame en cours peut être indiqué par la transmission de plus de 7 bits 1 consécutifs ; si le nombre de bits 1 consécutifs est ≥ 15 la liaison passe à l'état inactif. Certains protocoles (comme HDLC-LAPX) permettent d'exploiter la liaison uniquement en mode semi-duplex, d'autres (comme HDLC-LAPB) autorisent aussi le duplex intégral. Le service est assuré en général en mode avec connexion, mais un mode sans connexion est possible (trames **UI**).

Types de trames :

Trames d'informations : assurent le transfert des informations (PDU de niveau **réseau**) et, éventuellement, des accusés de réception. La commande correspondante est **I**. Toutes ces trames d'informations sont numérotées modulo 8 ou 128 et sont employées pour les transmissions en mode avec connexion.

Trames de supervision : contrôle des transmissions sur la liaison établie. Commandes correspondantes :

RR — *Receive Ready*, trames qui portent les accusés de réception quand le récepteur n'a pas de trame **I** à envoyer.

RNR — *Receive Not Ready*, le récepteur arrête l'émetteur, en lui indiquant le numéro de la trame suivante à envoyer.

REJ — *REJect*, le récepteur demande la retransmission de **toutes** les trames à partir du numéro spécifié.

SREJ — *Selective REJect*, le récepteur demande la retransmission sélective de la trame spécifiée.

Trames non numérotées : contrôle de l'établissement et de la libération des liaisons. Quelques commandes importantes :

SABM(E) — *Set Asynchronous Balanced Mode*, demande l'ouverture de connexion (ou le passage) en mode équilibré (**E** — étendu, champs de contrôle de 2 octets).

SARM(E) — *Set Asynchronous Response Mode*, demande l'ouverture de connexion (ou le passage) en mode non équilibré dans lequel la station secondaire peut émettre sans la permission de la station primaire (**E** — étendu, champs de contrôle de 2 octets).

SNRM(E) — *Set Normal Response Mode*, demande l'ouverture de connexion (ou le passage) en mode non équilibré dans lequel la station secondaire ne peut émettre qu'avec une permission reçue de la station primaire (**E** — étendu, champs de contrôle de 2 octets).

UA — *Unnumbered Acknowledge*, acquittement d'une commande portée par une trame non numérotée.

UI — *Unnumbered Information*, trame d'information non numérotée, utilisé pour les transmissions en mode sans connexion.

FRMR — *FRaMe Reject*, rejet de trame signalant une anomalie qui ne peut se suffire d'une simple retransmission (la cause est indiquée dans le champ d'informations de la trame : champ de commande incorrect, champ d'informations incorrect, champ d'informations trop long ou numéro de séquence anormal).

DISC — *DISConnect*, demande de déconnexion.

DM — *Disconnect Mode*, indique que la station est déconnectée.

XID — *eXchange IDentification*, échange d'identifications des stations (identifications dans le champ d'informations, utilisé par la couche réseau !).

Modes de dialogue entre les interlocuteurs :

Mode non équilibré : une station primaire qui gère la liaison et une ou plusieurs stations secondaires (liaison multipoint).

Mode équilibré : stations combinées qui peuvent se partager de façon égale la gestion de la liaison.

3.2.5.1. HDLC-LAPB

HDLC-LAPB (*Link Access Protocol Balanced*) permet un fonctionnement en mode équilibré (ABM, *Asynchronous Balanced Mode*), avec connexion, sur une liaison point à point et autorise le duplex intégral. HDLC-LAPB a été choisi pour le niveau 2 (liaison) des réseaux à commutation X 25 (comme TRANSPAC en France).

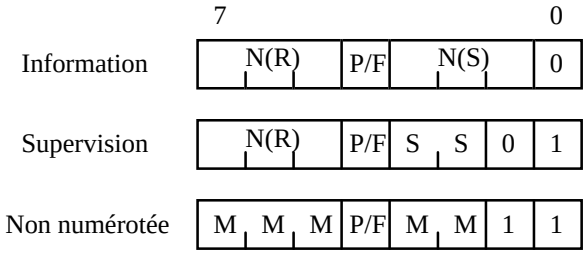
Format des trames :

Fanion (01111110)	1 octet
Adresse	1 octet
Commande	1 ou 2 octets (format étendu)
Information	variable, dépend de l'implémentation
Détection erreurs	2 octets

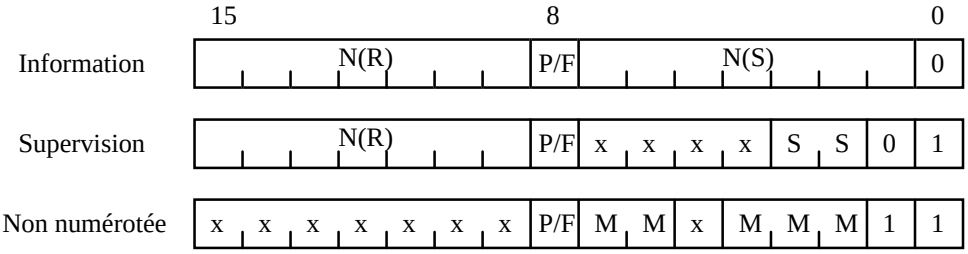
Adresse — sur les liaisons point à point deux valeurs uniquement sont utilisées (01H et 03H), pour indiquer si la trame est une commande ou une réponse (ce champ est employé aussi dans les tests par rebouclage de la ligne).

Commande — chaque type de trame possède son propre format

sur 1 octet :



sur 2 octets :



N(S) = numéro de la trame courante envoyée.

N(R) = numéro de la prochaine trame attendue (acquittement positif pour les trames antérieures).

bit P/F (*Poll/Final*) — bit P mis à 1 par une station oblige l'autre station de répondre immédiatement (pour LAP non équilibré joue le rôle de "jeton"), et avec le bit F à 1.

bits SS — codent les 4 types de trames de supervision (RR, RNR, REJ, SREJ).

bits M — codent les 32 types de trames non numérotées.

Information — PDU niveau réseau (SDU liaison) pour les trames d'informations (numérotées ou non), informations de contrôle pour les autres trames non numérotées.

Détection erreurs — sur la trame entière, le polynôme utilisé est $g(x) = x^{16} + x^{12} + x^5 + 1$ (CCITT V41). Peut être étendu à 4 octets en utilisant un polynôme générateur de degré 32 (option).

Conditions d'erreur :

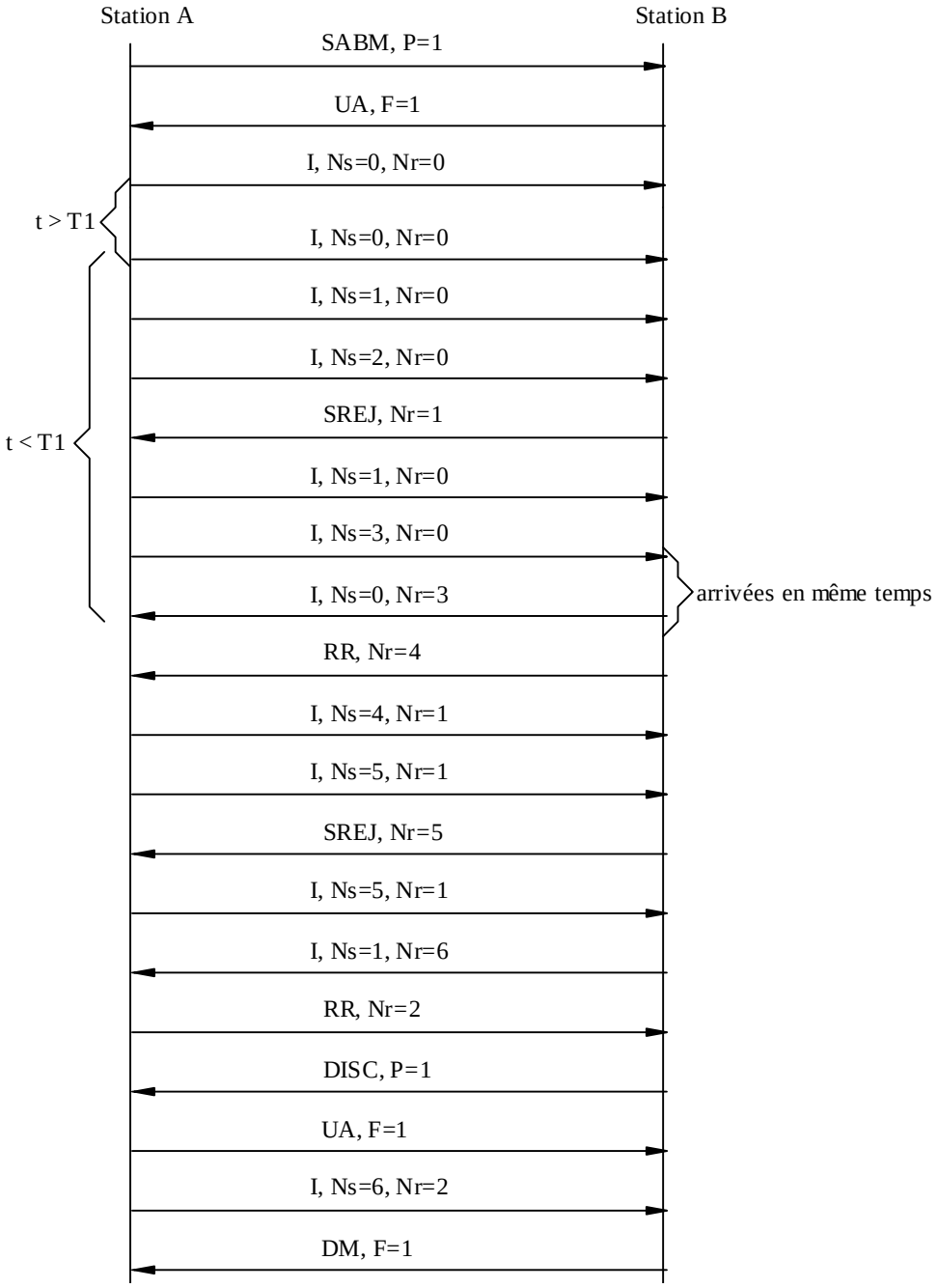
Détection d'erreurs grâce au code polynomial.

Adresse anormale.

Dépassement du délai de garde dans l'attente d'un accusé de réception.

Commande/ réponse anormale.

Exemple d'échange HDLC-LAPB :



Paramètres du protocole :

Délai de garde (T1) — à l'émission d'une trame, un horloge de garde est armé avec T1 ; si le délai expire avant l'arrivée de l'acquittement, la trame est retransmise.

Délai d'acquittement — après la réception d'une trame, la station émet un acquittement (en utilisant une trame **RR** s'il n'y a pas de trame **I** en attente) avant expiration de ce délai.

Taille de la trame (N1) — nombre maximum de bits dans la trame à l'exclusion des fanions.

Nombre maximum de retransmissions (N2) — si ce nombre est atteint suite à une successions de transmissions erronées ou sans acquittement, la liaison est fermée.

Délai d'ouverture — le produit entre le délai de garde et le nombre maximum de retransmissions.

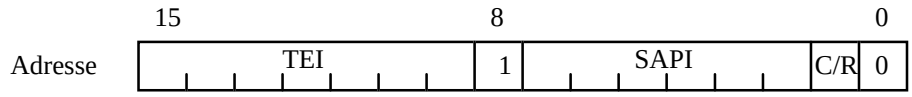
Valeur du crédit (K) — nombre de trames qui peuvent être émises sans attendre les accusés de réception correspondants ; la valeur est ≤ 7 pour le format normal et ≤ 127 pour le format étendu.

Une collision peut survenir et dans ce cas les trames sont retransmises avec des délais différents.

3.2.5.2. HDLC-LAPD

HDLC-LAPD ajoute aux caractéristiques de LAPB la possibilité de gestion des liaisons multipoint et une utilisation possible en mode sans connexion (trames **UI**). Ce protocole a été retenu pour le niveau liaison des réseaux numériques à intégration de services (RNIS) classiques (NUMERIS en France).

Format des trames — seule différence par rapport à LAPB, le champ d'adresse est sur 2 octets :



TEI — *Terminal End point Identifier*, identifie le terminal logique ou la destination finale pour l'information de niveau 3 (réseau), permettant ainsi l'utilisation sur des liaisons multipoint.

SAPI — *Service Access Point Identifier*, identifie l'entité réseau à laquelle la trame est destinée :

00 : la trame transporte l'information de signalisation ;

16 : la trame transporte des données ;

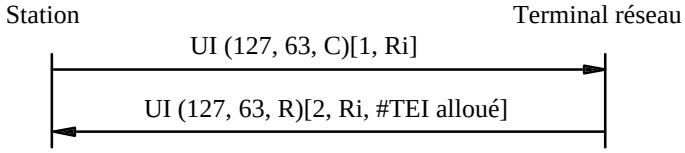
63 : la trame transporte l'information de gestion du terminal.

bit C/R — *Command/Response*, fonction similaire à celle de l'adresse employée dans LAPB (un seul bit utile).

L'allocation du TEI peut être automatique (TEI = 64-126) ou non (TEI = 0-63) ; si l'allocation n'est pas automatique il faut vérifier l'unicité du TEI sur le réseau multipoint. Allocation automatique :

1° L'entité de gestion du terminal (SAPI = 63) transmet une trame non numérotée **UI** avec TEI = 127 (adresse de diffusion) à l'ensemble des stations sur la liaison multipoint. Un nombre aléatoire (Ri) est également présent dans la trame afin d'éviter d'éventuels conflits avec d'autres terminaux demandant à obtenir une identification en même temps.

2° L'entité de gestion côté réseau cherche un numéro de TEI libre et transmet l'information, encapsulée dans une trame **UI** et avec le même nombre aléatoire pour identifier le destinataire, à l'entité de gestion du terminal.



3.3. RNIS et ATM

Les Réseaux Numériques à Intégration de Services (en France : NUMERIS) ont été développés par les opérateurs de télécommunications afin d'offrir aux utilisateurs des points d'accès uniques à des services de types très différents comme la téléphonie, le transport des données ou la vidéoconférence. L'utilisation d'un réseau numérique unique permet une meilleure intégration entre ces différents services, ainsi que de nombreuses facilités supplémentaires pour des services classiques comme la téléphonie.

3.3.1. RNIS classique

Issu de l'environnement de la téléphonie, le RNIS utilise une signalisation **hors bande** : on trouve sur le réseau un multiplexage temporel entre des canaux dédiés aux données et des canaux permettant de contrôler le transfert de ces données. Ainsi, les canaux **B** sont dédiés au transport des données utiles (des services de téléphonie ou de transport de données par commutation de circuits ou de paquets), les canaux **D** étant employés pour la signalisation et, éventuellement, pour le transport de données à faible débit.

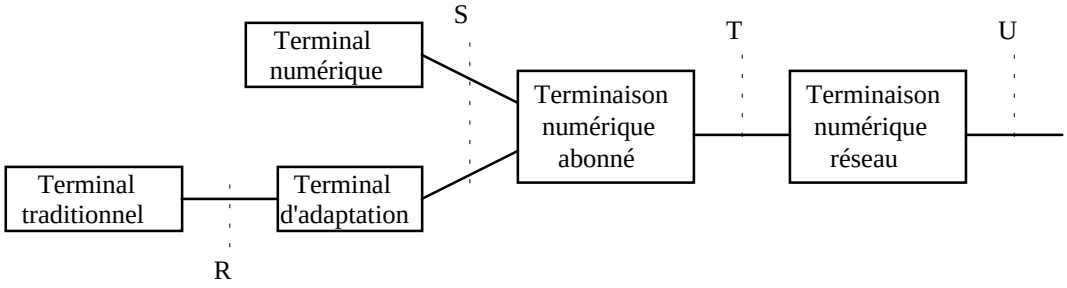
Types d'accès :

De base : 2 B + D ; débit en ligne 192 kbits/s, dont 144 kbits/s débit utile (2 canaux B à 64000 bits/s et 1 canal D à 16000 bits/s).

Intermédiaire : 9 B + D ; débit en ligne 704 kbits/s, dont 640 kbits/s débit utile.

Primaire : 30 B + D ; débit en ligne 2048 kbits/s, dont 1984 kbits/s débit utile.

Types d'interfaces (fonction du terminal pour R, recommandation I 431 pour S et T, U non normalisée) :



Les trames physiques de l'accès de base (codage bipolaire : niveau 0 pour les bits 1, niveaux -a et +a, alternativement, pour les bits 0) :

Trame terminal → réseau

F	L	**B1 (8 bits)**	L	**D**	L	F	L	**B2 (8 bits)**	L	**D**	L	**B1 (8 bits)**	L	**D**	L	**B2 (8 bits)**	L	**D**

2 bits de décalage

Trame réseau → terminal

F	L	**B1 (8 bits)**	E	**D**	A	Fa	N	**B2 (8 bits)**	E	**D**	S	**B1 (8 bits)**	E	**D**	S	**B2 (8 bits)**	E	**D**

(250 microsecondes)

F, Fa = bits de synchronisation et verrouillage de trames,
L = bits d'équilibre de la composante continue,
A = activation ou désactivation de l'équipement,
E = écho canal **D** terminal → réseau.

Protocoles employés (à partir du niveau liaison, pour le canal D uniquement) :
Niveau physique : interfaces S et T sur bus, résolution des conflits selon CSMA/CR.
Niveau liaison : HDLC-LAPD (recommandation Q 920 / I 441).
Niveau réseau : protocole X 25 PLP au service du protocole de signalisation de niveau supérieur.
Niveau supérieur : recommandation Q 930 / I 451 (CCITT 7) pour la signalisation sur le canal D.

3.3.2. ATM : RNIS large bande

ATM (*Asynchronous Transfer Mode*) est une technique de transport asynchrone utilisant des paquets de 53 octets, développée pour servir de support au RNIS large bande (*Broadband* ISDN). Ceci signifie que ATM devrait permettre de transporter à la fois des informations isochrones (voix, sons, images animées) et asynchrones (données).

Structure trame (5 octets en-tête, 48 octets données) :

[*Generic Flow Control*	4 bits]
Virtual Path	12 [ou 8 bits]
Virtual Circuit	12 bits
Payload Type	3 bits
réservé	1 bit
Cell Loss Priority	1 bit
Header Error Control	1 octet
Données	48 octets

Generic Flow Control : utilisé uniquement dans l'interface utilisateur-réseau (absent au-delà), fournit des niveaux de priorité entre les différentes connexions de l'utilisateur qui partagent le même accès réseau.

Virtual Path : identifie une voie virtuelle qui peut contenir plusieurs circuits virtuels.

Virtual Circuit : identifie un circuit virtuel. Les numéros de voie et de circuit virtuel ont une signification locale au lien et sont donc modifiés dans chaque brasseur/commutateur, selon des tables de routage dynamiques ; certaines valeurs sont réservées pour des usages spécifiques.

Payload Type : définit la nature de la charge utile (cellule utilisateur, cellule gestion réseau, cellule gestion congestion, etc.)

Cell Loss Priority : si le bit est 1, la cellule peut être jetée en priorité en cas de saturation d'un commutateur.

Header Error Control (HEC) : contrôle d'erreurs sur l'en-tête ; le polynôme employé permet de détecter certaines erreurs multiples et de corriger les erreurs simples. Les numéros de voie et de circuit virtuel

étant modifiés dans chaque brasseur/commutateur, le HEC n'est pas calculé/vérifié par la couche ATM mais par la sous-couche de convergence (TC) de la couche physique !

Architecture :

<table>
<tr><td>Fonction des couches supérieures</td><td colspan="2">Couches supérieures</td></tr>
<tr><td>Convergence</td><td>CS</td><td rowspan="2">AAL</td></tr>
<tr><td>Segmentation et réassemblage</td><td>SAR</td></tr>
<tr><td>Contrôle de flux générique
Génération en-tête cellules
Commutation (traduction VPI/VCI)
Multiplexage/démultiplexage</td><td colspan="2">ATM</td></tr>
<tr><td>Génération/vérification contrôle erreur en-tête
Adaptation à la trame de transmission
Génération/récupération trame transmission</td><td>TC</td><td rowspan="2">Couche physique</td></tr>
<tr><td>Synchronisation bit</td><td>PM</td></tr>
</table>

Caractéristiques :

Le service offert par la couche ATM est en mode connecté (un circuit doit être établi avant que des données ne puissent être envoyées) mais sans acquittement, donc non fiable. Les cellules successives d'un même circuit virtuel arrivent toujours dans l'ordre d'émission (car elles prennent toutes le même chemin), mais certaines cellules peuvent être perdues (saturation d'un commutateur). C'est le niveau supérieur (AAL) qui, selon le type de service assuré, doit éventuellement s'occuper des retransmissions. Aussi, dans la couche ATM le contrôle d'erreurs porte uniquement sur l'en-tête, l'intégrité des données transportées doit être vérifiée par les niveaux supérieurs.

Chaque circuit virtuel est unidirectionnel.

Le routage est assuré par des tables dont le contenu est mis à jour de façon dynamique, en fonction des circuits virtuels ouverts. A l'ouverture d'un circuit virtuel, des ressources lui sont réservées dans chaque brasseur/commutateur traversé.

Des fonctions d'adaptation très différentes (couche AAL) doivent être utilisées pour différents services offerts : un transfert asynchrone de données n'impose pas les mêmes exigences qu'un transfert isochrone d'images, par exemple. Catégories de fonctions d'adaptation :

Type 1 : trafic isochrone — la source produit à débit et rythme fixe des unités de données qui doivent être délivrées au même rythme à destination. Il faut pouvoir identifier la perte d'information sans chercher à y remédier.

Type 2 : trafic de source isochrone — le rythme de la source est fixe mais le débit variable, le rythme doit être maintenu de façon **approximative** à destination. Il faut pouvoir identifier la perte d'information sans chercher à y remédier.

Type 3 : trafic asynchrone entre entités identifiées (en général applications transactionnelles). Le débit demandé peut être contraint dans les bornes négociées et les délais doivent être raisonnables. Les pertes de données (perte de cellules, erreurs de transmission) doivent être corrigées.

Type 4 : trafic asynchrone en rafale (datagrammes sur réseau local). Le débit instantané demandé peut être très élevé, les délais sont peu importants. Les pertes de données doivent être corrigées.

Avantages :

La granularité fine facilite le multiplexage des informations asynchrones avec des informations isochrones (une trame isochrone doit attendre au maximum une trame asynchrone avant d'accéder au réseau), ce qui n'est pas le cas pour des techniques comme *Frame relay* qui utilisent des trames de longueur variable, en général élevée.

La taille réduite et fixe des trames permet aussi un traitement rapide dans les commutateurs, et donc des délais de routage réduits.

Enfin, des cellules de taille réduite permettent un taux de remplissage plus élevé que des cellules de taille importante.

Problèmes :

La taille réduite des cellules fait baisser le rendement (le rapport entre la taille des données utiles et la taille de la cellule). La taille choisie est un compromis entre le rendement et le délai d'acheminement (avec la contrainte de pouvoir acheminer des signaux isochrones).

Le non respect par un utilisateur du débit négocié a une influence négative sur tous les utilisateurs qui emploient le même support. Une fonction de "police" complexe doit être implémentée.

4. La couche réseau

La couche réseau assure l'acheminement entre la machine source et la machine destination à travers une succession de connexions physiques, en utilisant sur chaque connexion les services offerts par une couche liaison spécifique.

4.1. Types de services offerts à la couche transport

Les types de services prévus pour la couche réseau :

1° Service sans connexion et sans acquittement (appelés aussi *datagram*) : utilisé quand le support est fiable et le contrôle des erreurs et de flux est assuré par la couche transport (exemple : protocole IP).

2° Service avec connexion (en général circuit virtuel) : canal fiable entre deux machines offert à la couche réseau ; livraison garantie, pas de duplication de trames, respect de l'ordre (exemple : protocole X 25).

4.2. Aspects de la couche réseau

4.2.1. Adressage

Types d'adresses :

Adresse "physique" ou adresse logique — l'adresse "physique" identifie de façon unique la connexion d'une machine à un type de réseau (par exemple l'adresse Ethernet, câblée dans chaque carte Ethernet), l'adresse logique est indépendante de la machine support et identifie un programme ou un utilisateur (par exemple l'adresse IP).

Adresse "plate" ou structurée — l'adresse structurée peut être décomposée en champs à signification individuelle bien définie, éventuellement en rapport avec la localisation géographique (exemples : numéros de téléphone, adresses IP), contrairement à une adresse "plate" (par exemple l'adresse Ethernet).

Les adresses employées dans la couche réseau sont structurées mais peuvent être "physiques" (en général les adresses X 121) ou logiques (adresses IP).

Les adresses X 121 (norme CCITT pour les réseaux publics de transmission de données) sont structurées, sur 14 chiffres décimaux : 3 pour le pays (certains pays possèdent plusieurs codes), 1 pour le réseau à l'intérieur du pays, 10 chiffres pour l'adresse de l'hôte (7 pour la région, 3 pour le numéro local).

4.2.2. Routage

Types de routage :

Routage non-adaptatif ou routage adaptatif — pour le routage adaptatif la route peut être modifiée **en temps réél**, en fonction de l'indisponibilité temporaire (pannes, congestion) de certaines parties du réseau.

Routage centralisé ou routage distribué — pour le routage centralisé les décisions sont prises par un noeud (ou un nombre réduit de noeuds) superviseur(s), pour le routage distribué les décisions sont prises dans chaque noeud ; des algorithmes mixtes (voir plus loin) sont souvent employés.

Quelques techniques de routage (utilisant ou non des tables de routage dans chaque noeud) :

1° Routage centralisé non-adaptatif : en cas d'indisponibilité le superviseur est informé, il calcule des nouvelles tables de routage et envoie la table correspondante à chaque noeud ; reconfiguration très coûteuse, avec risque d'obsolescence (pratiquement toujours le cas dans de grands réseaux).

2° Routage centralisé synchrone : les noeuds envoient des comptes rendus au superviseur de façon périodique, à des instants bien déterminés ; temps de réponse est inférieur à celui de la technique précédente, mais charge du réseau par des comptes-rendus périodiques.

3° Routage centralisé asynchrone : un noeud envoie un compte rendu au superviseur dès que la différence par rapport au compte rendu précédent dépasse un seuil ; plus efficace que le précédent. Utilisé sur des sous-réseaux (nombre relativement réduit de noeuds), de façon hiérarchique : le contrôle est centralisé dans chaque sous-réseau et distribué entre les superviseurs des différents sous-réseaux.

4° Routage par inondation : chaque noeud transmet chaque message reçu par tous ses ports de sortie (chaque message a une durée de vie limitée afin d'éviter l'effondrement du réseau) ; très robuste par rapport aux modifications de la topologie du réseau et ne nécessite aucune communication pour le contrôle, mais très inefficace (surcharge du réseau).

5° Routage aléatoire : chaque noeud transmet chaque message reçu sur un port de sortie choisi aléatoirement (chaque message a une durée de vie limitée) ; robuste et ne nécessite aucune communication pour le contrôle, mais très inefficace (délais moyens très importants).

6° Routage par la source : la liste des adresses des noeuds à traverser est présente dans l'en-tête du message ; très efficace sur un réseau idéal, mais assez inefficace dans un réseau réel (toute

reconfiguration est très coûteuse) ; utilisé uniquement à la demande, pour contourner la technique de routage choisie.

7° Routage distribué avec utilisation du plus court chemin : les tables de routage sont mises à jour grâce à la communication entre les noeuds voisins, la longueur des chemins est déterminée dans chaque noeud et transmise aux voisins. Des informations concernant la congestion peuvent avoir une influence sur la "longueur" d'un chemin.

Etant donnée la complexité du problème du routage, l'utilisation d'adresses structurées est indispensable. Choix d'un algorithme : compromis entre la qualité de l'algorithme et la surcharge qu'il introduit sur le réseau.

4.2.3. Contrôle de la congestion

Toutes les connexions entre les noeuds du réseau n'utilisent pas un protocole de niveau liaison avec contrôle de flux, ce qui rend indispensable un contrôle de la congestion.

Techniques de contrôle de la congestion :

Techniques à seuil : la plus simple — le nombre de clients et le débit de chaque client sont limités. Technique isarythmique : un message peut entrer dans le réseau uniquement si un jeton est disponible ; le jeton devient disponible à l'arrivée à la destination finale du message auquel il était collé. Autre possibilité : utilisation de fenêtres pour la communication de bout en bout (souvent au niveau transport). En raison de l'indéterminisme des communications aucune technique à seuil ne peut empêcher la surcharge de certains noeuds.

Techniques à préallocation des ressources : utilisées en mode avec connexion, des ressources jugées suffisantes sont allouées à l'ouverture de chaque circuit virtuel. Une surallocation des ressources est en général employée afin d'améliorer l'efficacité.

Destruction de paquets ou de circuits virtuels : le noeud surchargé détruit un nombre suffisant de paquets ou de circuits virtuels, la reprise est à la charge de la couche réseau (protocole réseau avec acquittement) ou de la couche transport (protocole réseau sans acquittement).

Plusieurs techniques sont en général employées ensemble dans un même réseau.

4.2.4. Primitives couche réseau OSI ↔ ULP (*Upper Layer Protocol*)

Protocole réseau orienté connexion :

N-CONNECT.request, .indication, .response et .confirm — paramètres : adresse appelé, adresse appelant, paramètres qualité de service (à négocier), 16 octets de données.

N-DATA.request et .indication — paramètres : demande de confirmation, données.

N-DATA-ACKNOWLEDGE.request et .indication — aucun paramètre.

N-EXPEDITED-DATA.request et .indication — paramètres : données.

N-RESET.request et .indication — paramètres : cause Reset.

N-RESET.response et .confirm — sans paramètres.

N-DISCONNECT.request et .indication — paramètres : cause déconnexion, données.

4.3. Protocole IPv4

Paquet IP (*Internet Protocol*) :

Version	4 bits
Longueur en-tête	4 bits
Type service	1 octet
Longueur totale	2 octets
Identification	2 octets
Fanions	3 bits
Offset fragment	13 bits
Durée de vie	1 octet
Protocole	1 octet
Contrôle en-tête	2 octets
Adresse source	4 octets
Adresse destination	4 octets
Options	variable
SDU	variable

Version : la version du protocole IP qui a créé le paquet (4 actuellement, 6 bientôt).

Longueur en-tête : multiple de 4 octets ; concerne aussi les éventuelles options.

Type service : identifier le type de service assuré ; différentes options (délais, débit, priorité, fiabilité) peuvent être demandées, des champs "Options" seront alors présents. Bits 0÷2 — priorité (7 = contrôle réseau, 6 = contrôle inter-réseaux, 4 = supérieur à *flash*, 3 = *flash*, 2 = immédiat, 1 = prioritaire, 0 =

standard). Bit 3 — délai (0 = normal, 1 = réduit). Bit 4 — débit (0 = normal, 1 = élevé). Bit 5 — fiabilité (0 = normale, 1 = élevée). Les bits 6 et 7 sont réservés.

Longueur totale : longueur totale du paquet, en-tête + SDU.

Identification, fanions, offset fragment : permettent la segmentation (fragmentation) des messages. Les fragments (1 fragment = 8 octets) d'un message possèdent un même numéro d'identification mais des offsets (de 0 à 8192, correspondant donc à des paquets IP d'au maximum 65528 octets) différents. Les fanions indiquent si le message peut être fragmenté et, dans le cas affirmatif, marquent le dernier fragment. Le réassemblage des fragments est en général effectué par la couche IP du destinataire. Des fragmentations successives dans des routeurs différents sont possibles, les valeurs de l'offset correspondant à la taille des fragments. A l'arrivée les différents fragments d'un même paquet sont remis dans le bon ordre — IP n'assure pas l'arrivée des messages dans le bon ordre mais garantit le réassemblage correct des **morceaux** de messages !

Durée de vie : chaque routeur réduit de 1 la valeur et jette le paquet si le résultat est nul.

Protocole : identifie le protocole de niveau supérieur qui doit recevoir le paquet (par exemple TCP, UDP, ICMP, RDP).

Contrôle en-tête : permet de vérifier la transmission correcte de l'en-tête ; aucun mécanisme de contrôle de transmission n'est fourni par IP pour la SDU.

Adresse source et destination : adresses IP, **sans rapport avec les adresses physiques** des cartes réseau !

Options : utilisées soit pour préciser la qualité de service, soit pour des activités de management/diagnose. Un champ option contient un code d'identification (1 octet), la longueur (sur 1 octet) et les données spécifiques (plusieurs octets).

Options :

Source routing (routage par la source) : le champ option contient un pointeur et la liste des adresses IP pour les noeuds que le paquet doit traverser, fournie par l'émetteur du paquet. La liste peut être exhaustive (*strict source routing*) ou des noeuds intermédiaires supplémentaires peuvent être acceptés (*loose source routing*). Implique toujours l'utilisation conjointe de l'option suivante.

Record route : chaque noeud ajoute son adresse dans la liste contenue dans le champ option.

Internet time-stamp : en plus d'enregistrer les adresses IP des noeuds intermédiaires, le protocole enregistre pour chaque noeud le moment (GMT, valeur en milisecondes) où le paquet a quitté le noeud.

Primitives IP/ULP (IP/*Upper Layer Protocol* ; spécification abstraite, détails dépendant de l'implémentation) :

SEND = primitive utilisée par ULP pour demander à IP l'émission d'un paquet ; les paramètres transmis (en plus des données) permettent de spécifier les champs de l'en-tête IP du paquet à émettre. Le résultat de la demande de service (attention, IP est non fiable !) est retourné à l'ULP.

RECV = primitive utilisée par IP pour informer l'ULP destinataire qu'un paquet est arrivé ; les paramètres transmis (en plus des données) correspondent aux champs de l'en-tête IP du paquet arrivé.

Primitives IP/SNAP (IP/*SubNetwork Access Protocol*) :

SNAP_SEND = primitive utilisée par IP pour demander à SNAP l'émission d'une trame ; les paramètres transmis (en plus des données) sont : adresse sous-réseau locale (destination), indicateurs type service, longueur IP PDU (ou SNAP SDU).

SNAP_DELIVER = primitive utilisée par SNAP pour informer IP destinataire qu'une trame est arrivée ; en plus des données des indicateurs d'erreurs peuvent être transmis (option non standardisée...).

Primitives x/LLC (x/*Logical Link Control* IEEE 802.2) en mode sans connexion :

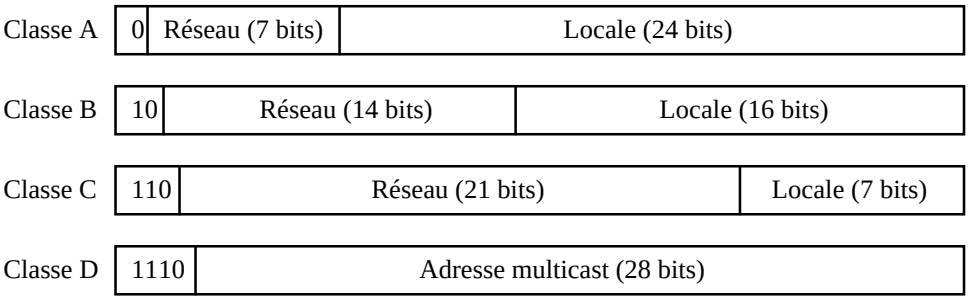
DL-UNITDATA.*request* = demande à LLC l'émission d'une trame ; paramètres : adresse physique source, adresse physique destination, données, priorité.

DL-UNITDATA.*indication* = indique au protocole x (de niveau supérieur) destinataire qu'une trame est arrivée ; mêmes paramètres que pour DL-UNITDATA.*request*.

Quand IP est utilisé avec LLC, un protocole de convergence doit implémenter la correspondance entre les primitives IP/SNAP avec leurs paramètres et les primitives x/LLC avec leurs paramètres.

4.3.1. Adressage IPv4

Les adresses IP (actuellement sur 32 bits) permettent d'identifier une connexion d'une machine (ordinateur, routeur) à un réseau indépendamment de l'adresse physique de cette connexion. Les adresses physiques sont en général (ex. pour Ethernet) câblées dans la carte réseau et changent donc avec la carte. Les adresses IP permettent donc d'identifier de façon permanente un service alors que le support matériel de ce service peut changer (évolution, maintenance). Une adresse IP consiste en un numéro de réseau et une adresse locale (sur le réseau spécifié). Traditionnellement, il y a 4 classes d'adresses IP :



Une adresse IP est représentée comme une suite de 4 valeurs correspondant aux 4 octets, séparées par des points, par exemple 128.14.155.202. Dans le champ adresse locale on peut en général distinguer entre un numéro de sous-réseau et un numéro de machine. Cette distinction est donnée, au niveau de chaque sous-réseau, par un masque de 32 bits (certaines implémentations n'offrent pas ce mécanisme !) : les bits 0 indiquent la partie réservée au numéro de machine, les bits 1 indiquent la partie réservée au numéro de réseau et de sous-réseau.

Types d'émission : à destinataire unique, *broadcast* — à toutes les machines d'un réseau ou sous-réseau (les bits indiquant le numéro de machine sont tous à 1), *multicast* — à toutes les machines situées sur un même sous-réseau ou des sous-réseaux différents et appartenant à un même groupe *multicast* (adresse classe D ; l'adresse multicast est traduite par un routeur et une liste d'adresses IP ou en une adresse *multicast* du protocole de niveau inférieur, par exemple Ethernet ; des opérations périodiques de remise à jour des listes des membres du groupe sont nécessaires).

Aujourd'hui, une solution plus flexible est adoptée pour les adresses unicast : un masque sur 4 octets est employé pour indiquer quels bits de l'adresse IP font partie du numéro de réseau (ceux qui correspondent à des bits à 1 dans le masque) et quels bits font partie du numéro de machine (ceux qui correspondent à des bits à 0 dans le masque). Quand la couche IP d'une machine reçoit des données à envoyer dans un paquet IP, le numéro de réseau du destinataire est comparé au numéro de réseau de la machine locale. Deux cas sont possibles :

1° Un message est envoyé à une destination sur le même réseau local (sous-réseau) avec une adresse IP destination ; à cette adresse IP la couche physique de l'émetteur associe une adresse physique permettant aux machines connectées au réseau de rejeter un message qui ne leur est pas destiné dès son arrivée dans la couche physique (sans avoir donc à le faire remonter jusqu'à la couche réseau IP).

2° Un message est envoyé à une destination distante avec une adresse IP destination ; sur chaque sous-réseau assurant le transfert, une adresse physique identifiant le nœud suivant (routeur) traversé par le message est associée par la couche réseau au message.

Une table de correspondances présente dans chaque machine connectée au réseau local (sous-réseau) indique l'adresse physique associée à chaque adresse IP du réseau local. Pour mettre à jour cette table (suite à la connexion d'une nouvelle machine au réseau, à une opération de maintenance, etc.) on utilise le protocole ARP (*Address Resolution Protocol*) : la machine qui constate que sa table n'est pas à jour émet (en mode *broadcast* afin d'atteindre toutes les machines) une requête ARP avec l'adresse IP en question, la machine qui reconnaît son adresse IP répond en indiquant son adresse physique. Certaines implémentations d'IP peuvent être configurées pour qu'une réponse ARP soit prise en compte par toutes les machines connectées au réseau.

Le cas inverse peut se présenter, une machine connaît son adresse physique (câblée dans la carte réseau) mais pas son adresse IP (par ex. machines sans disque qui utilisent le réseau pour se mettre en route). Le protocole RARP (*Reverse Address Resolution Protocol*) permet de résoudre ce problème : la machine qui cherche son adresse IP émet (en mode *broadcast* afin d'atteindre toutes les machines) une requête RARP avec son adresse physique et la machine qui est serveur RARP sur le réseau lui répond en lui indiquant l'adresse IP correspondante.

Les adresses IP étant difficiles à mémoriser par les utilisateurs, des noms symboliques (par exemple `cs.cmu.edu`, `balzac.univ-tours.fr`) sont associés aux adresses IP (→ niveau supplémentaire de résolution d'adresse...). Ces noms symboliques sont organisés de façon hiérarchique dans ce qu'on appelle le *Domain Name System* (DNS) ; l'adresse doit être lue à partir de la **droite**, par exemple `balzac.univ-tours.fr` indique : domaine `fr` (France), sous-domaine `univ-tours`, host `balzac` (serveur mail Centre de

calcul, sous OpenVMS...). L'association noms → adresses IP est assurée par un ensemble de serveurs de noms (en l'absence de serveur de noms sur le réseau local, il faut utiliser l'adresse IP...). Chaque station de travail possède une cache locale qui garde les associations les plus récemment utilisées ; si une association n'est pas présente dans la cache, avant d'émettre le message avec l'adresse IP destination, la station contacte le serveur local de noms afin de récupérer l'association ; le serveur local peut retransmettre la requête à d'autres serveurs (opération récursive) ou peut indiquer à la station de travail l'adresse d'un autre serveur (opération non récursive) s'il ne connaît pas l'association.

4.3.2. Routage IPv4

Exemple d'architecture (adresses IP classe B ; les carrés représentent des routeurs) :

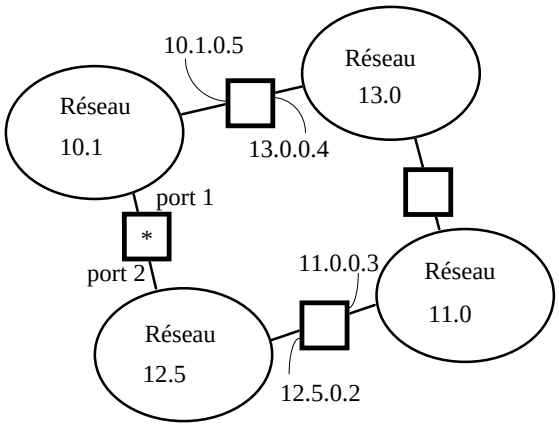


Table de routage pour le routeur * (très simplifiée ; adresses IP classe B) :

Numéro réseau destination	Routage vers	A travers le port
10.1	adresse physique correspondant à l'adresse IP destination	1
11.0	adresse physique correspondant à l'adresse IP **12.5.0.2**	2
12.5	adresse physique correspondant à l'adresse IP destination	2
13.0	adresse physique correspondant à l'adresse IP **10.1.0.5**	1

Algorithme de routage (très simplifié) :

Si destination sur un réseau directement connecté
Alors associer adresse physique destination et transmettre directement
Sinon Si routage spécifique
Alors suivre routage spécifique
Sinon Si adresse IP destination présente dans la table de routage
Alors associer adresse physique noeud suivant et transmettre
Sinon Si routage par défaut défini
Alors transmettre noeud par défaut
Sinon Erreur de routage

Construction et mise à jour des tables de routage : *Route Discovery Protocols* (RDP), ne font pas partie de IP. Les informations prises en compte pour construire et mettre à jour les tables de routage :

Nombre de sous-réseaux et routeurs traversés : technique plus ancienne et plus simple.

Métrique de l'état des liens : technique plus récente, la métrique peut concerner plusieurs caractéristiques comme le délai, le débit, etc. Le routage dépendra en général du champ "type service" de l'en-tête IP. Un routage adaptatif — les caractéristiques des liens et donc les tables de routage changent dans le temps — est souvent implémenté.

Terminologie en rapport avec RDP :

Autonomous system : réseau administré par une autorité unique ; peut être divisé en *areas.*

Interior/exterior gateway : routeurs se trouvant à l'intérieur/à la frontière d'un *autonomous system*. Les routeurs se trouvant à la frontière doivent échanger des informations (*External Gateway Protocol* commun) avec des routeurs d'autres *autonomous systems*. Les routeurs internes doivent échanger des informations uniquement entre eux et peuvent utiliser des *Internal Gateway Protocols* spécifiques au *autonomous system.*

4.3.3. ICMP

ICMP (*Internet Control Message Protocol*, RFC 792 et 1256) accompagne le fonctionnement de IP. Les messages ICMP sont encapsulés dans des paquets IP. Utilisation :

Informer l'émetteur de l'impossibilité de livraison d'un paquet — la destination ne peut pas être atteinte.

Informer l'émetteur que la durée de vie d'un paquet a expiré.

Informer l'émetteur sur les erreurs de format de paquet détectées.

Informer un noeud que le noeud suivant a épuisé sa capacité de stockage de paquets.

L'utilisation principale de ICMP est donc de créer des rapports pour les erreurs ; ceci ne rend pas IP fiable, des paquets peuvent ne pas arriver à destination alors qu'aucun message ICMP n'est reçu par l'émetteur du paquet.

4.4 Évolution de IP : IPv6

IP (*Internet Protocol*) : niveau réseau (3 OSI), mode sans connexion, non fiable et sans garantie de séquencement (*datagram*) ; permet la segmentation/le regroupement des messages.

TCP (*Transmission Control Protocol*) : niveau transport (4 OSI), mode avec connexion, fiable et avec contrôle de flux.

UDP (*User Datagram Protocol*) : niveau transport (4 OSI), mode sans connexion, non fiable et sans garantie de séquencement (*datagram*).

ARP (*Address Resolution Protocol*) : traduire une adresse IP en adresse physique.

RARP (*Reverse Address Resolution Protocol*) : traduire une adresse physique en adresse IP.

RDP (*Route Discovery Protocols*) : famille de protocoles permettant de diriger les messages et de mettre à jour les tables de routage.

4.4.1. Problèmes posés par IPv4

Quand IPv4 a été développé le nombre d'équipements connectés était relativement faible, les équipements mobiles étaient très rares et les délais de transmission n'avaient pas une grande importance car les données transmises n'étaient pas urgentes. Mais l'environnement d'utilisation a complètement changé et les caractéristiques de IPv4 posent actuellement des problèmes importants :

1° Epuisement des adresses, dû a l'explosion du nombre de sous-réseaux et d'équipements connectés.

2° Pour faire face à l'épuisement des adresses, l'utilisation des masques a remplacé les classes d'adressage, ce qui rend plus complexes les algorithmes et les tables de routage. Aucune technique de configuration automatique des espaces d'adressage n'a été définie.

3° Accommodation difficile des équipements mobiles, actuellement très nombreux : en effet, chaque mobile doit avoir une adresse IP complètement différente selon l'endroit où il se trouve, ce qui pose des problèmes très difficiles de gestion des adresses.

4° Indisponibilité de classes de service correspondant aux exigences imposées par des flots de données très divers : transfert de fichiers, sessions interactives, conversations téléphoniques, vidéoconférence, etc.

5° Certaines opérations effectuées dans les routeurs — recalcul du code de contrôle après la modification du champ durée de vie, fragmentation/réassemblage des messages — sont très coûteuses en temps de calcul et augmentent les délais d'acheminement.

4.4.2. IPv6

Paquet IPv6 (IP nouvelle génération, RFC 1752, offre commerciale disponible) :

Version	4 bits
Classe de trafic	4 bits
Etiquette de flot	3 octets
Longueur données	2 octets
En-tête suivant	1 octet
Nombre de sauts	1 octet
Adresse source	16 octets (128 bits)
Adresse destination	16 octets (128 bits)
SDU IPv6	variable

Version : la version du protocole IP qui a créé le paquet, 6 pour IPv6.

Classe de trafic : les valeurs de 0 à 7 sont employées pour les flots à paquets contrôlés (0 = trafic non caractérisé, 1 = trafic ne demandant pas de réponse, 2 = transfert de données intempestives, 3 et 5 sont réservés, 4 = transferts volumineux attendus, 6 = trafic interactif, 7 = trafic de contrôle Internet), les valeurs 8 à 15 pour les flots à paquets non contrôlés (par exemple conversations téléphoniques ; plus la valeur est élevée, moins l'utilisateur est disposé à accepter que les paquets soient jetés en cas de congestion).

Etiquette de flot : permet d'identifier un flot, qui est une séquence de paquets envoyés depuis une source particulière à une destination particulière, séquence pour laquelle la source désire un traitement particulier par les routeurs concernés.

Longueur données : longueur de ce qui suit l'en-tête, jusqu'à 64 Koct (codée sur 2 octets) ; une valeur supérieure (→ *Jumbogram*) peut être indiquée dans une option.

En-tête suivant : permet l'extension des en-têtes existants, en indiquant quelle entité de protocole doit être appelée afin de traiter l'en-tête suivant. Dans IPv6, les options sont indiquées dans des en-têtes supplémentaires, traités uniquement par le noeud identifié par l'adresse destination (à l'exception de l'option de routage par la source, dont l'en-tête est traité par chaque routeur intermédiaire). La longueur de chaque en-tête supplémentaire est un multiple de 8 octets.

Nombre de sauts (*hop count*), ancien champ durée de vie : chaque routeur réduit de 1 la valeur et jette le paquet si le résultat est nul.

Adresse source et adresse destination : adresses IPv6, voir les détails plus loin.

Quelques options proposées :

Authentification des utilisateurs et confidentialité des données.

Auto-configuration des adresses, permettant aux un stations connectées à un sous-réseau de se construire une adresse.

Routage par la source et marquage du chemin (modification des structures associées aux étiquettes de flot).

Fonctions de fragmentation.

Contrôle des erreurs.

4.4.2.1. Adressage

La définition et la gestion des adresses IPv6 doivent faciliter la tâche des utilisateurs/administrateurs des réseaux et aussi l'activité des routeurs. Cela est possible grâce notamment à la richesse de l'espace d'adressage, à la possibilité de définir de multiples niveaux hiérarchiques d'adresses, à la présence d'adresses de type *cluster* et aux mécanismes de configuration automatique des adresses.

Une adresse IPv6 est représentée comme une succession de 8 groupes de valeurs hexadécimales représentées sur 4 chiffres, par exemple 1080:222:AF45:FF:FE:143:4441:110. Une succession de 0 peut être représentée par "::", comme dans FEDC::122:AD45:4555. Une adresse IPv4 encapsulée dans une adresse IPv6 sera x:x:x:x:x:x:d.d.d.d (x = valeur hexadécimale codant 16 bits, d = valeur décimale codant 8 bits), comme dans 0:0:0:0:0:0:192.22.128.1, représentée aussi ::192.22.128.1.

IPv6 possède 22 classes d'adresses, dont 17 sont réservées pour un usage futur. Les adresses IPv6 sont de trois types :

1° *Unicast* : adresse d'un correspondant unique, bien défini. Les adresses IPv6 de noeuds utilisant IPv4 ont pour préfixe 0000 0000.

2° *Cluster* : adresse d'un groupe de noeuds qui partagent un même préfixe d'adresse. Un paquet (datagramme) envoyé à une telle adresse sera livré au routeur le plus proche situé sur la frontière du domaine. Cela permet notamment de préciser de façon simple, en utilisant l'option de routage par la source, le ou les opérateur(s) télécom dont on veut utiliser les services. Dans une adresse *cluster* la partie de poids fort est le préfixe partagé par les adresses du *cluster* et la partie de poids faible est 0 (par conséquence, toutes les adresses IPv6 ayant une succession de 0 comme partie de poids faible sont réservées et ne doivent pas être employées comme adresses *unicast*).

3° *Multicast* : adresse de diffusion utilisée pour envoyer un datagramme à tous les membres d'un groupe multicast. Les adresses de *broadcast* sont remplacées dans IPv6 par des adresses *multicast*. Toutes les adresses multicast débutent par 1111 1111. Certaines adresses *multicast* prédéfinies permettent de simplifier le fonctionnement des protocoles (par exemple, FF0E::43 identifie tous les serveurs *Network Time Protocol* de l'Internet).

Contrairement aux adresses IPv4 qui sont totalement indépendantes des adresses de niveau inférieur (liaison/physique), une adresse *unicast* IPv6 est censée incorporer l'adresse de niveau inférieur. Par exemple, pour une station connectée via un réseau local IEEE 802, l'adresse MAC sur 48 bits (dont l'unicité dans le monde est garantie par les constructeurs des cartes réseau) forme les 48 bits de poids faible de l'adresse IPv6 (la source d'inspiration a été IPX). Ceci simplifie l'autoconfiguration des adresses et l'assignation dynamique d'une adresse à un mobile. Pour un réseau local qui n'est pas connecté à Internet, des adresses "lien local" peuvent être utilisées : une telle adresse est de type FEx0::<adresse MAC> (avec x = 1000 ou 1100) ; ces adresses peuvent être configurées par les stations connectées, en l'absence de toute intervention d'un utilisateur/administrateur ou d'un routeur.

Deux mécanismes existent pour former l'adresse Internet d'une station. Suivant le premier, la station forme son adresse en ajoutant un préfixe de réseau présent dans un message ICMPv6 *Router Advertisement* envoyé

périodiquement par un routeur local à un suffixe qui est en général l'adresse MAC IEEE 802. La station peut aussi envoyer une demande ICMPv6 *Router Sollicitation* pour obtenir le préfixe sans plus attendre.

Suivant le deuxième mécanisme, l'adresse est assignée par le routeur ou un autre serveur d'adresses IP grâce au protocole DHCPv6 (*Dynamic Host Configuration Protocol*). L'administrateur du réseau local peut préciser non seulement l'ensemble de préfixes à utiliser, mais aussi les masques (pour IPv4), l'adresse du DNS, l'adresse du routeur par défaut, ainsi qu'un certain nombre de paramètres du protocole IP (taille des paquets, durée de vie conseillée, etc.). Ces différents paramètres peuvent être transmis à une station non seulement à la première connexion mais aussi ultérieurement, à la demande de la station. Les serveurs d'adresses gardent (*bindings*) la configuration IP de chaque station gérée : identification de la station, adresse IP, paramètres IP et durée de vie (*lease*) de l'association station−adresse. Un dialogue DHCP typique se déroule comme suit :

1° La station qui vient d'être connectée au réseau local envoie en *broadcast* un message *DHCP_Discover*.
2° Plusieurs serveurs d'adresse peuvent répondre par des messages *DHCP_Offer*. Chaque réponse contient l'adresse du serveur, l'adresse IP proposée à la station et plusieurs paramètres IP.
3° La station choisit le serveur DHCP qui lui convient et lui envoie un message *DHCP_Request*, en lui demandant éventuellement des paramètres IP supplémentaires.
4° Le serveur choisi sauvegarde le *binding* correspondant à la station et lui envoie les paramètres demandés dans un message *DHCP_ACK*.
5° Avant d'utiliser l'adresse IP obtenue, la station fait appel à ARP pour s'assurer de l'unicité de cette adresse.
6° Pour prolonger la durée de vie de l'adresse IP assignée, la station envoie un DHCP_Request avec la valeur de cette adresse. Si l'adresse n'est plus valide, le serveur répond par un *DHCP_NAK* et la station doit redémarrer une configuration d'adresse.

Le routage dans IPv6 est similaire, à quelques extensions près, à celui de IPv4. L'extension la plus importante concerne l'utilisation des adresses *cluster* dans l'option de routage par la source.

4.4.2.2. IPv6 et la mobilité

Un mobile est une station pour laquelle le point de rattachement à Internet peut changer souvent. Pour chaque mobile on définit un identificateur qui est similaire à une adresse (c'est en général l'adresse du mobile sur son réseau maison) et qui ne change pas dans le temps, ainsi qu'une adresse IP qui indique donc le point courant de rattachement du mobile et qui peut donc évoluer. La correspondance entre l'identificateur et l'adresse courante est gardée dans des AMT (*Address Mapping Table*). Chaque entrée dans la table contient aussi un indicateur de la durée de maintien de la validité de cette entrée, ainsi qu'un numéro de version de l'adresse, numéro incrémenté à chaque changement d'adresse. Pour la création et la mise à jour d'entrées dans une AMT, une authentification (du mobile ou de la source du message) est effectuée au préalable sur les paquets reçus.

A son arrivée sur un réseau (ou avec une périodicité donnée), un mobile se fait attribuer une adresse IP adéquate ; si l'adresse MAC IEEE 802 est employée dans l'adresse IPv6, chaque mobile possède par défaut une adresse sur chaque réseau IEEE 802. Ensuite, il envoie un paquet — avec l'option de marquage de la route et le paramètre de contrôle de la mobilité — à son réseau maison. Chaque routeur qui intervient dans ce transfert met à jour son AMT (après authentification du mobile) à partir des informations présentes dans le paramètre de contrôle de la mobilité (identification, adresse, version de l'adresse, durée de maintien, etc.), si l'entrée est absente de sa table ou si le numéro de version de l'adresse est supérieur à celui présent déjà dans la table. Le réseau maison ne peut pas déterminer la nouvelle adresse d'un de ses mobiles si celui-ci ne se fait pas connaître en utilisant le mécanisme décrit.

Le protocole de niveau supérieur (comme TCP) emploie l'identifiant d'un mobile pour lui envoyer un paquet et c'est la couche IPv6 qui retrouve (éventuellement) l'adresse à jour du mobile dans son AMT.

4.4.2.3. IPv6 et la sécurité

Contrairement à IPv4 qui fait l'hypothèse que les mécanismes de sécurité sont présents à un niveau supérieur (niveau application pour l'environnement TCP/IP ou niveau présentation pour le modèle OSI), IPv6 inclut des mécanismes de sécurité à son niveau (réseau). Toutefois, ces mécanismes de sécurité ne protègent pas contre l'analyse du trafic : le contenu des messages peut être confidentiel mais certaines adresses, informations concernant les routes, le débit et les paramètres de la qualité de service restent accessibles dans les routeurs.

L'intérêt d'introduire la sécurité au niveau réseau est de fournir aux utilisateurs des niveaux de sécurité supplémentaires. Par exemple, une société implantée sur plusieurs sites peut utiliser la sécurité au niveau réseau pour la protection (automatique, non gérée par les utilisateurs) des communication inter-sites et la sécurité au niveau présentation ou application pour la protection (non automatique, gérée par les utilisateurs) des communications intra-site. Aussi, les mécanismes de sécurité au niveau réseau permettent de renforcer la protection des réseaux internes contre les intrusions.

Le premier mécanisme défini utilise un en-tête d'authentification et permet d'assurer l'authenticité de l'émetteur et l'intégrité (mais pas la confidentialité) du contenu des paquets IPv6. Seuls les champs qui sont modifiés dans chaque routeur (comme *Hop Count* ou *Next Address*) sont exclus de la vérification de l'intégrité. Ce mécanisme permet entre autres de s'assurer que les champs qui interviennent dans le filtrage d'accès des paquets (comme les adresses, le protocole de transport, le numéro de port, etc.) sont authentiques. La technique proposée pour l'authentification et contrôle de l'intégrité est *Message Digest 5* (MD5, développée par Rivest).

Le deuxième mécanisme utilise un en-tête de confidentialité (*Encapsulating Security Payload Header*, ESPH) qui permet en plus d'assurer la confidentialité du contenu à l'aide d'une technique de chiffrement. Le *datagram* IP en entier peut être encapsulé (*Tunnel mode*) et chiffré ou uniquement le segment de niveau transport (TCP ou UDP, *Transport mode*). La technique proposée est CBC-DES (*Cipher Block Chaining Data Encryption Standard*). La gestion des clés n'est pas incorporée dans IP mais est assurée par un protocole de niveau supérieur. Les deux mécanismes de sécurité sont des mécanismes de bout en bout (sauf si l'encapsulation d'un protocole de niveau réseau dans IPv6 est employée).

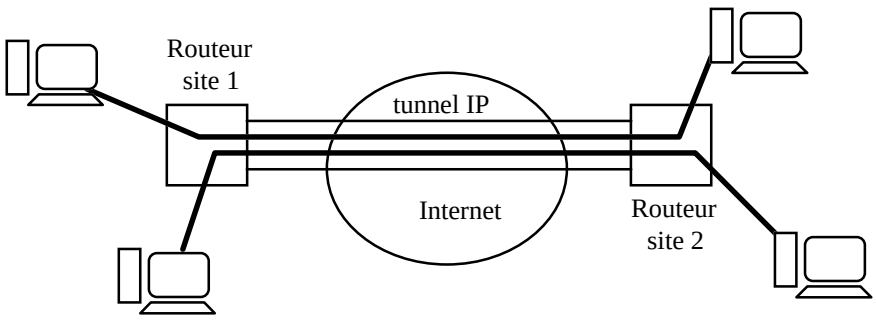
Chaque machine garde dans une table les informations de sécurité concernant ses possibles utilisateurs et interlocuteurs. A chaque interlocuteur correspond une ou plusieurs entrées (une entrée par utilisateur, usage, etc.) dans la table. Chaque entrée contient un numéro d'identification ou *Security Parameters Index* (SPI), l'adresse IP correspondante, l'identité des protocoles utilisés pour l'authentification et la confidentialité et les clés associées.

Considérons trois scénarios et les solutions correspondantes :

1° Une société désire sécuriser ses communications client↔serveur intra-site. L'utilisation de MD5 et de l'en-tête d'authentification permet de faire authentifier les clients par les serveurs et réciproquement. Ainsi, si les codes d'authentification sont différents (et confidentiels) pour toutes les machines, une machine ne peut pas se substituer à une autre en copiant son adresse IP. En même temps, grâce à la vérification de l'intégrité, une machine ne peut pas se substituer à une autre en copiant le bon en-tête et en remplaçant le contenu du message initial par des données fausses.

2° A l'intérieur d'une société, un administrateur est la seule personne habilitée à transférer des données très sensibles entre deux machines ; ces transferts doivent rester opaques. Un SPI distinct correspondra à l'utilisateur concerné et, par rapport aux autres utilisateurs des deux machines, une clé d'authentification différente sera employée. Aussi, la confidentialité sera utilisée (en-tête de confidentialité présent dans les paquets, contenu crypté en *Transport mode*) et donc une clé de cryptage sera présente.

3° Une société désire connecter ses différents sites distants via Internet, mais veut assurer une sécurité maximale au niveau réseau pour les transferts ainsi que pour les accès. Un tunnel IP sera créé : les routeurs "frontaliers" de chaque site utilisent l'authentification réciproque et la vérification de l'intégrité, ainsi que le cryptage et l'encapsulation de tous les paquets IP qui doivent circuler entre les sites (confidentialité *Tunnel mode*) dans des paquets IP inter-sites ; de cette façon, non seulement les données mais aussi les en-têtes IP (donc les informations sensibles comme les adresses IP, les paramètres de trafic, les débits, etc.) sont cachés vis à vis de l'extérieur. Sur chaque site, les machines emploient éventuellement une authentification et un cryptage poste à poste.



4.5. Protocole X 25 PLP (pour info…)

Protocoles X 25 (ex CCITT, actuel UIT-T) :

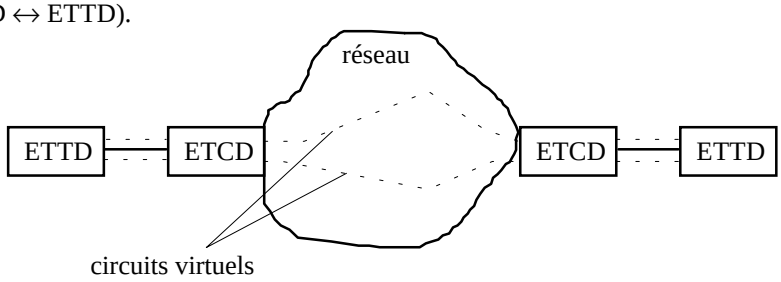
X 25 couche physique : X 21 (accès synchrone) ou X 21 bis (accès asynchrone).

X 25 couche liaison : HDLC-LAPB.

X 25 couche réseau : X 25 PLP (*Packet Layer Protocol*).

4.5.1. Caractéristiques

Le protocole X 25 de niveau réseau assure un service en mode connecté, avec acquittement local ou de bout en bout. Les deux ETTD (*DTE*) qui communiquent sont connectés à des ETCD (*DCE*) qui font partie du réseau. L'avis X 25 du CCITT définit l'interface ETTD ↔ ETCD ainsi que — pour la couche réseau — le dialogue de bout en bout (ETTD ↔ ETTD).



Pour X 25 PLP, l'établissement d'une connexion entre deux ETTD engendre l'établissement d'un circuit virtuel (CV), qui est ensuite suivi par chaque paquet jusqu'à la déconnexion des ETTD. Un CV peut être unidirectionnel ou (le plus souvent) bidirectionnel. Plusieurs CV (associés à des connexions différentes) peuvent être présents à un moment donné entre deux ETTD et chaque CV peut correspondre à un chemin physique différent dans le réseau. Les circuits virtuels peuvent être de deux types : CV permanents (CVP), établis de façon permanente entre deux ETTD (utilisés pour des volumes importants de données ou des sessions interactives ; tarification au forfait), et CV commutés, établis au coup par coup (utilisés pour des volumes peu importants de données et sessions non interactives ; tarification à la durée).

Un CV établi entre deux ETTD est identifié par une paire de numéros de voie logiques (NVL) : l'ETTD qui demande l'établissement d'une connexion utilise un NVL qui arrive à l'ETCD relié à l'ETTD récepteur ; ce NVL peut être déjà utilisé par un autre CV connecté à l'ETTD récepteur, dans ce cas l'ETCD alloue un nouveau NVL qui, avec le premier, permettra de définir le CV à établir. Pour les appels sortants le NVL est donc choisi par l'ETTD, pour les appels entrants par l'ETCD.

Considérons un ETTD relié à un ETCD et supposons qu'au même moment l'ETTD désire établir un CV pour lequel il choisit un NVL et l'ETCD reçoit une demande de connexion qui porte le même NVL. Une collision se produit et X 25 spécifie que dans ce cas c'est l'appel sortant qui l'emporte. L'appel entrant sera repris avec un NVL disponible. Afin de réduire au maximum la probabilité de collision les ETTD allouent les NVL à partir de la valeur la plus élevée et les ETCD à partir de la valeur la moins élevée.

Les mêmes problèmes (collisions, nécessité de changer un NVL) peuvent se poser entre des noeuds internes du réseau, mais leurs solutions ne sont pas concernées par l'avis X 25.

L'adressage utilisé par X 25 PLP respecte l'avis X 121. Toutefois, des adresses sont utilisées uniquement à l'établissement d'une connexion, ensuite les numéros de voies logiques (NVL) permettent de s'affranchir des adresses.

4.5.2. Types de paquets

Paquets d'établissement CALL REQUEST (ETTD → ETCD) et INCOMING CALL (ETCD → ETTD) :

- 4 premiers bits : 0001.
- Numéro de voie logique (12 bits).
- Identification : 0000 101**1** ; le LSB indique qu'il s'agit d'un paquet de contrôle (pour les paquets de données LSB = **0**).
- 4 bits 0000.
- Longueur adresse appelant (sur 4 bits) en nombre de chiffres décimaux (multiple de 4 bits).
- Longueur adresse appelé.
- Adresse appelant, adresse appelé (respectant l'avis X 121).
- Longueur champ facilités en nombre d'octets (max. 63).
- Facilités (dépendent du réseau et des équipements employés) : sélection rapide (voir plus bas), demande appel en PCV, acheminement haute priorité, circuit virtuel (CV) duplex/simplex, taille paquet, taille fenêtre, interdiction d'appels entrants, groupe fermé d'abonnés, etc.
- Données utilisateur : 16 octets de données qui peuvent être utilisés de différentes façons (indiquer un complément d'adresse, un nom de login et un mot de passe, etc.)

La sélection rapide (*Fast Packet X 25*) est spécifiée comme une facilité et permet d'inclure 128 octets de données dans un paquet CALL REQUEST/INCOMING CALL. L'appelé peut répondre par un CLEAR REQUEST (qui inclut lui aussi 128 octets de données en réponse) si l'établissement d'un CV n'est pas nécessaire ou par un CALL ACCEPTED si un CV doit être établi.

Paquets d'établissement **CALL ACCEPTED** (ETTD → ETCD) et **CALL CONNECTED** (ETCD → ETTD) :

Numéro de voie logique (12 bits).

Identification : 0000 111**1**.

Longueur champ facilités en nombre d'octets (le champ "Facilités" est présent uniquement si l'appelé n'accepte pas les paramètres spécifiés par l'appelant).

Facilités : au cas où les propositions présentes dans le champ facilités du paquet INCOMING CALL ne conviennent pas à l'appelé, il peut faire des contre-propositions (les contre-propositions doivent être plus proches des valeurs par défaut que les propositions originales).

Paquets de libération **CLEAR REQUEST** (ETTD → ETCD) et **CLEAR INDICATION** (ETCD → ETTD), utilisés aussi en cas de rejet de la demande de connexion par le réseau ou l'appelant :

Numéro de voie logique (12 bits).

Identification : 0001 001**1**.

Cause (1 octet) : libération par ETTD (00h), numéro occupé (01h), appel invalide (03h), dérangement réseau (05h), dérangement appelé (09h), numéro inconnu (0Dh), erreur locale/distante (11/13h).

Diagnose (optionnel, 1 octet) : détails éventuels concernant les problèmes détectés.

Paquets de libération (ou de rejet) **DCE/DTE CLEAR CONFIRM** :

Numéro de voie logique (12 bits).

Identification : 0001 011**1**.

Paquets de données **DCE/DTE DATA** :

Bit Q (*Qualified*) : aucune signification pour X 25 PLP, mais utilisé par le niveau supérieur (transport) en général pour distinguer entre les paquets véhiculant des données et les paquets véhiculant des messages de contrôle du protocole de niveau transport.

Bit D : D = 0 ⇒ le contrôle de flux est local (ETTD ↔ ETCD) ; D = 1 ⇒ le contrôle de flux est de bout en bout (service fiable : acquittement envoyé par l'ETTD destinataire final).

Modulo (2 bits) : 01 = le numéro de séquence (Ns) et le numéro d'acquittement (Nr) sont modulo 8 ; 10 = Ns et Nr sont modulo 128 — dans ce cas un octet supplémentaire est employé pour Nr et Ns

Numéro de voie logique (12 bits).

Identification (1 octet) : **rrr**M sss**0** ; **rrr** = numéro d'acquittement (même convention que pour HDLC), sss = numéro de séquence, M = 1 ⇒ le paquet n'est pas le dernier du message

Données : 128 octets par défaut et jusqu'à 2048 par négociation.

Paquets de contrôle de flux **DCE/DTE RR**, **DCE/DTE RNR**, **DCE/DTE REJ** (signification similaire à celle des trames HDLC RR, RNR et REJ) :

2 premiers bits : 00.

Modulo : comme pour DCE/DTE DATA.

Numéro de voie logique (12 bits).

Identification : **rrr**0 000**1** pour RR, **rrr**0 010**1** pour RNR et **rrr**0 100**1** pour REJ ; **rrr** = numéro d'acquittement.

Paquets d'interruption **DCE/DTE INTERRUPT** (permettent de transmettre 32 octets de données hors séquence, donc à l'arrivée ces données peuvent être délivrées immédiatement à la couche transport) :

4 premiers bits : 0001.

Numéro de voie logique (12 bits).

Identification : 0010 001**1**.

Données : 32 octets.

Paquets de confirmation d'interruption **DCE/DTE INTERRUPT CONFIRM** (acquittement de bout en bout des paquets d'interruption) :

4 premiers bits : 0001.

Numéro de voie logique (12 bits).

Identification : 0010 011**1**.

Paquets de contrôle **RESET REQUEST**, **RESET INDICATION** (à la demande d'un ETTD ou du réseau à travers un ETCD, remise à zéro des compteurs Ns et Nr sur un circuit virtuel spécifique après détection d'une erreur concernant l'état de ces compteurs) :

4 premiers bits : 0001.

Numéro de voie logique (12 bits).

Identification : 0001 101**1**.

Cause (1 octet) : Reset par ETTD (00h), dérangement appelé (01h), erreur de procédure locale/distante (03/05h), incident réseau (07h), etc. En cas de Reset par ETTD, celui-ci transmet un RESET REQUEST au réseau, ce qui engendre la réinitialisation du CV concerné. En cas de Reset par le réseau, celui-ci

transmet des RESET INDICATION aux deux ETTD concernés, entraînant la fermeture du CVC/la réinitialisation du CVP.

Diagnostic (optionnel, 1 octet).

Paquets de contrôle **DCE/DTE RESET CONFIRM** (confirmation de réinitialisation) :

4 premiers bits : 0001.

Numéro de voie logique (12 bits).

Identification : 0001 111**1**.

Paquets de redémarrage **RESTART REQUEST**, **RESTART INDICATION**, **DCE/DTE RESTART CONFIRM** : le problème détecté est plus grave que dans le cas antérieur, tous les CVC de l'ETTD/ETCD sont fermés et les CVP sont réinitialisés. Structure :

4 premiers bits : 0001.

Numéro de voie logique (12 bits) : tous les bits sont à 0 !

Identification : 1111 101**1** pour REQUEST/INDICATION, 1111 111**1** pour CONFIRM.

Cause (1 octet, uniquement pour REQUEST/INDICATION) : reprise par ETTD (01h), reprise par le réseau (02h). En cas de reprise par l'ETTD, le réseau transforme le RESTART REQUEST reçu en un ensemble de RESET INDICATION pour chaque CVP et CLEAR INDICATION pour chaque CVC dans lequel l'ETTD initiateur est impliqué. En cas de reprise par le réseau, celui-ci transmet un RESTART INDICATION à l'ETTD concerné et se charge d'envoyer des RESET INDICATION et CLEAR INDICATION pour les CV dans lesquels l'ETTD est impliqué.

Diagnostic (optionnel, 1 octet, uniquement pour REQUEST/INDICATION).

Paquets de diagnostic **DIAGNOSTIC** (permet au réseau d'informer un ETTD de différents incidents qui ne remettent pas en question l'existence du CV, par exemple un paquet dont l'octet identification est illégal) :

4 premiers bits : 0001.

Numéro de voie logique (12 bits).

Identification : 1111 000**1**.

Données de diagnose.

Nous pouvons remarquer qu'aucun champ permettant la détection/correction d'erreurs n'est présent dans les paquets de X 25 PLP, qui fait confiance à la couche liaison pour le contrôle d'erreurs.

Équivalence entre primitives couche réseau OSI et X 25 PLP :

N-CONNECT.request	Envoi de CALL REQUEST
N-CONNECT.indication	Arrivée de INCOMING CALL
N-CONNECT.response	Envoi de CALL ACCEPTED
N-CONNECT.confirm	Arrivée de CALL CONNECTED
N-DISCONNECT.request	Envoi de CLEAR REQUEST
N-DISCONNECT.indication	Arrivée de CLEAR INDICATION
N-DATA.request	Envoi de DTE DATA
N-DATA.indication	Arrivée de DCE DATA
N-DATA-ACKNOWLEDGE.request	Envoi de DTE RR/RNR/REJ
N-DATA-ACKNOWLEDGE.indication	Arrivée de DCE RR/RNR/REJ
N-EXPEDITED-DATA.request	Envoi de DTE INTERRUPT
N-EXPEDITED-DATA.indication	Arrivée de DCE INTERRUPT
N-RESET.request	Envoi de RESET REQUEST
N-RESET.indication	Arrivée de RESET INDICATION
N-RESET.response	—
N-RESET.confirm	—

5. La couche transport

La couche transport sert d'interface entre les couches supérieures qui demandent la transmettre des informations de bout en bout de façon fiable et économique, à travers un réseau dont les performances (dont la fiabilité) peuvent varier fortement.

5.1. Classes de service

Le protocole de transport utilisé dépendra de la qualité du service offert par la couche réseau sous-jacente : plus le service offert par la couche réseau est imparfait, plus le protocole de transport doit prendre en charge la fiabilité du transfert d'informations.

On distingue en général trois niveaux de qualité de service offerts par la couche réseau :

A : Taux acceptable (c'est à dire négligeable...) d'erreurs signalées (détectable au niveau réseau) et non signalées (non détectables au niveau réseau) — service jugé parfait. Exemple : certains réseaux X 25.

B : Taux acceptable d'erreurs non signalées et taux inacceptable d'erreurs signalées (des demandes de réinitialisation de la connexion réseau sont présentes). Exemple : X 25 dans la plupart des cas.

C : Taux inacceptable d'erreurs signalées et non signalées (pas de contrôle d'erreurs, paquets perdus, dupliqués ou arrivés dans un mauvais ordre, réinitialisations présentes). Exemple : IP dans la plupart des cas (en général à l'exception des réseaux locaux).

Principales caractéristiques des classes de protocoles de transport orientés connexion prévues par l'OSI :

Mécanismes prévus\Classes de transport	0	1	2	3	4
Segmentation et réassemblage (TSDU ↔ n TPDU)	oui	oui	oui	oui	oui
Concaténation et séparation (n TPDU ↔ NSDU)	non	oui	oui	oui	oui
Multiplexage et éclatement	non	non	oui	oui	oui
Contrôle de flux	non	non	option	oui	oui
Numérotation TPDU (séquencement)	non	oui	option	oui	oui
Données exprès	non	option	option	oui	oui
Reprise sur erreur détectée et signalée	non	oui	non	oui	oui
Utilisation de plusieurs connexions réseau	non	non	non	non	**oui**
Contrôle d'inactivité	non	non	non	non	**oui**
Détection d'erreurs sur TPDU	non	non	non	non	option

L'adéquation entre le niveau de qualité offert par la couche réseau et la classe de protocole transport :

Réseau	Transport	Caractéristiques principales de la classe de protocole de transport
A	0	Classe de base, la confiance faite à la couche réseau est totale
B	1	Numérotation TPDU, reprise sur erreur (détectée)
A	2	Numérotation TPDU, contrôle de flux, fonctions de multiplexage
B	3	Numérotation TPDU, contrôle de flux, reprise sur erreur et multiplexage
C	4	La confiance faite à la couche réseau est minimale

Adressage : l'adresse réseau permet d'identifier le correspondant et donc aussi son entité de transport ; une adresse reste toutefois indispensable pour déterminer le processus correspondant de niveau supérieur au niveau transport ; l'adresse est appelée TSAP (*Transport Service Access Point*, terminologie OSI) ou port (terminologie TCP/UDP).

5.2. TCP et UDP

Les protocoles de transport TCP (avec connexion) et UDP (sans connexion) sont utilisés en conjonction avec le protocole réseau IP.

5.2.1. TCP

Caractéristiques TCP :

Protocole orienté connexion : le transfert de *segments* est précédé de l'établissement d'une connexion et son état est suivi en permanence. Les connexions TCP sont duplex intégral (*full-duplex*).

Contrôle d'erreurs et reprise : des acquittements sont émis par le récepteur, indiquant le premier segment non encore reçu. Les segments erronés sont retransmis. L'émetteur utilise des timers afin de réémettre automatiquement les segments pour lesquels l'acquittement s'est fait attendre trop longtemps.

Les segments sont remis en ordre à l'arrivée, à partir d'un numéro de séquence. Un seul exemplaire est gardé d'un segment reçu correctement plusieurs fois.

TCP reçoit les données du niveau supérieur sous forme de *stream* d'octets (de faibles volumes de données arrivent continuellement) ; le *stream* est segmenté par TCP et les segments sont émis. Les segments peuvent être de taille variable (la taille peut être choisie à l'établissement de la connexion).

TCP implémente une fonction *push* qui permet à l'application de forcer l'émission des données transférées à TCP, sans attendre qu'un segment entier soit prêt pour l'envoi. Par exemple, dans le cadre d'une session interactive sur une machine distante, l'utilisateur peut simplement donner une commande, suivie par ↵ ; la commande doit être transmise immédiatement, car la suite du dialogue peut dépendre de la réponse du système distant. La fermeture de la connexion produit le même effet qu'un *push* explicite.

Une synchronisation est assurée entre l'émetteur et le récepteur afin d'éviter le dépassement de la capacité de stockage à la réception.

TCP permet de définir des niveaux de priorité et des niveaux de sécurité (implémentés pour une partie des produits TCP seulement).

Une connexion TCP est individualisée par la paire de *sockets* [source, destination]. Un *socket* se compose d'une adresse IP et d'un numéro de *port*. Un numéro de port identifie un processus de niveau supérieur (niveau "application") qui demande un service TCP. Certains numéros de ports sont réservés, par exemple le 21 et le 20 pour FTP (commandes et respectivement données), le 23 pour Telnet, le 25 pour SMTP, le 110 pour POP3, le 111 pour RPC, le 79 pour Finger, le 80 pour HTTP.

Ouverture de ports : ouverture passive — une demande de connexion est attendue (facilite l'établissement de la connexion) ; ouverture active — une demande de connexion est effectuée (en général à un port ouvert par une ouverture passive, mais deux ouvertures actives sont acceptées, ainsi qu'une ouverture active sans ouverture passive antérieure à l'autre bout).

Segment TCP :

Port source	2 octets
Port destination	2 octets
Numéro séquence	4 octets
Numéro acquittement	4 octets
Data offset	4 bits
Fanions	12 bits
Fenêtre	2 octets
Contrôle	2 octets
Pointeur urgent	2 octets
Options	variable
Données	variable

Port source, port destination : numéros permettant d'identifier les processus source/destinataire de niveau supérieur.

Numéro séquence : la position du premier octet de données du segment dans la suite de données à transmettre (à l'aide de plusieurs segments successifs). Attention, pour TCP le numéro de séquence n'est pas le numéro du segment !

Numéro acquittement : numéro de séquence de l'octet suivant à recevoir (acquittement positif ⇒ tous les octets antérieurs ont été correctement reçus). Les acquittements concernent les segments de données transférés dans l'autre sens (*piggybacking*).

Data offset : longueur en mots de 4 octets de l'en-tête TCP.

Fanions : permettent d'établir et de contrôler la connexion

URG : le champ "Pointeur urgent" est à prendre en compte.

ACK : le champ "Numéro acquittement" est à prendre en compte.

PSH : le segment correspond à l'exécution d'une fonction *push*.

RST : la connexion doit être remise à zéro.

SYN : les numéros de séquence doivent être synchronisés (utilisé à l'établissement de la connexion : pour un segment CONNECT REQUEST, SYN = 1 et ACK = 0 ; pour un segment CONNECT CONFIRM, SYN = 1 et ACK = 1). SYN = 1 uniquement pour le segment CONNECT REQUEST, ce qui permet à un *firewall* de les reconnaître.

FIN : fin de transmission (plus de segments à transmettre).

Fenêtre : indique combien d'octets le récepteur peut encore accepter (contrôle flux).

Contrôle : détection d'erreurs sur le segment entier, plus une partie le l'entête IP (notamment l'adresse IP source et l'adresse IP destination). Ce champ est toujours utilisé par TCP.

Pointeur urgent : indique où se trouvent les données urgentes ; le niveau supérieur est informé sans délai de l'arrivée de données urgentes (interruptions, codes de contrôle, etc.).

Options : actuellement 3 sont définies : *end-of-option-list* (0), ineffectif (1), taille maximum segment (2).

Pour un transfert bidirectionnel les segments TCP ont le même format et tous les champs mentionnés sont présents. Si un des correspondants doit transmettre un acquittement et n'a pas de données à envoyer, il se contente d'émettre un segment **sans données**, avec le numéro d'acquittement correctement positionné et, comme numéro de séquence, la valeur qui correspond aux (éventuelles) données suivantes. Ce même numéro de séquence sera présent dans l'en-tête du segment qui transférera ces (éventuelles) données, dès qu'elles seront disponibles.

Étapes dans l'établissement d'une connexion TCP, par exemple entre la partie serveur et la partie cliente d'une application (situées sur des machines différentes), à l'initiative de la partie cliente :

1° Le serveur demande à la couche TCP de sa machine (nous l'appellerons TCP serveur) une ouverture passive (accepte des demandes de connexion sur un numéro de port spécifique, dédié au serveur).

2° Le client demande à la couche TCP de sa machine (nous l'appellerons TCP client) une ouverture active, en lui transférant l'adresse IP de la machine sur laquelle tourne le serveur, ainsi que le numéro de port qui identifie le serveur.

3° Le TCP client prend une valeur initiale (obtenue à partir de l'horloge de la machine) pour le numéro de séquence correspondant au transfert de données client → serveur (par exemple, 100) et envoie un segment de demande de connexion, avec SYN = 1, ACK = 0. Dans le même segment, il indique la taille de la fenêtre de réception (dans "fenêtre") et la taille du segment le plus grand qu'il peut accepter ("option 2"). La taille des segments transférés peut être différente pour les deux sens d'une connexion TCP.

4° A la réception de ce segment de demande de connexion (SYN = 1 et ACK = 0), le TCP serveur prend une valeur initiale pour le numéro de séquence correspondant au transfert de données serveur → client (par exemple, 200), positionne le numéro d'acquittement sur le premier octet attendu du client (dans notre exemple, 101), indique la taille de la fenêtre de réception (dans "fenêtre") et la taille du segment le plus grand qu'il peut accepter ("option 2"), et positionne les fanions à SYN = 1 et ACK = 1.

5° A la réception de cette réponse (SYN = 1 et ACK = 1), le TCP client envoie un segment avec le numéro d'acquittement positionné sur le premier octet attendu du serveur (dans notre exemple, 201), et les fanions positionnés : SYN = 0 et ACK = 1. Le TCP client indique ensuite à la partie cliente de l'application que la connexion TCP est établie avec le serveur.

6° A la réception de cette confirmation (SYN = 0 et ACK = 1), le TCP serveur indique à la partie serveur de l'application que la connexion avec le client est établie.

Fin normale d'une connexion TCP (même exemple, communication client ↔ serveur) :

1° La partie serveur de l'application demande au TCP serveur de fermer la connexion avec le client.

2° Le TCP serveur envoie un segment avec FIN = 1.

3° Le TCP client envoie un segment avec ACK = 1 et informe la partie cliente de l'application des intentions du serveur.

4° La partie cliente de l'application indique au TCP client que la connexion peut être fermée. Le TCP client envoie un segment avec FIN = 1.

5° Le TCP serveur répond au client par un segment avec ACK = 1, et informe la partie serveur de l'application que la connexion est interrompue.

Fin abrupte d'une connexion TCP : en cas de problème de communication grave, un des interlocuteurs envoie plusieurs segments successifs avec RST = 1. La connexion est immédiatement interrompue.

Composantes TCB (*Transmission Control Block*, permet à TCP de suivre l'état d'une connexion) : les descriptifs du *socket* local et distant, des pointeurs vers les tampons d'émission et de réception, pointeur vers la file de retransmission, valeurs de sécurité et de priorité pour la connexion, variables décrivant le transfert duplex intégral (numéros séquence, confirmation, fenêtre, etc.).

Primitives ULP (*Upper Layer Protocol*) → TCP :

UNSPECIFIED_PASSIVE = indique à TCP d'accepter une connexion en provenance d'un *socket* quelconque ; paramètres : port local, *timeout* ULP, action *timeout* ULP, priorité, sécurité, options.

FULL-PASSIVE-OPEN = indique à TCP d'accepter une connexion en provenance d'un *socket* spécifique ; paramètres : port local, *socket* distant, *timeout* ULP, action *timeout* ULP, priorité, sécurité, options.

ACTIVE-OPEN = indique à TCP d'établir une connexion avec un *socket* spécifique ; paramètres : port local, *socket* distant, *timeout* ULP, action *timeout* ULP, priorité, sécurité, options.

ACTIVE-OPEN-WITH-DATA = indique à TCP d'établir une connexion avec un *socket* spécifique et de transmettre des données sans délai ; paramètres : port local, *socket* distant, *timeout* ULP, action *timeout* ULP, priorité, sécurité, données, fanion *push*, fanion urgent.

OPEN = sollicite à TCP le résultat d'une demande d'ouverture ; paramètres : sécurité, options, connexion (nom local d'identification).

SEND = transmet à TCP des données à émettre ; paramètres : connexion (nom local d'identification), adresse tampon, longueur, fanion *push*, fanion urgent.

RECEIVE = transmet à TCP une demande d'accepter des données reçues ; paramètres : connexion (nom local d'identification), adresse tampon, longueur, fanion *push*, fanion urgent.

ALLOCATE = indique à TCP d'allouer un tampon pour des données ; paramètres : connexion, taille tampon.

CLOSE = demande à TCP la fermeture d'une connexion ; paramètre : connexion.

ABORT = demande à TCP la fermeture prématurée d'une connexion ; paramètre : connexion.

STATUS = demande à TCP l'état d'une connexion ; paramètre : connexion.

Primitives TCP → ULP :

OPEN-ID = indique l'identité d'une connexion ouverte ; paramètres : connexion, *socket* distant.

OPEN-FAILURE = indique une ouverture échouée ; paramètre : connexion.

OPEN-SUCCESS = indique une ouverture réussie ; paramètre : connexion.

DELIVER = indique la réception de données ; paramètres : connexion, adresse tampon, taille tampon, fanion urgent.

CLOSING = confirmation de fermeture d'une connexion ; paramètre : connexion.

TERMINATE = indication de terminaison des transferts sur une connexion ; paramètres : connexion, état terminaison.

STATUS-RESPONSE = réponse à une primitive *STATUS* ; paramètres : connexion, *socket* local, port distant, état connexion, fenêtres émission et réception, confirmations émission et réception, mode urgent, timeout, action timeout.

ERROR = indication d'erreur ; paramètres : connexion, description erreur.

L'interface avec la couche inférieure (presque exclusivement IP) n'est pas spécifiée par TCP mais par cette couche (voir primitives ULP/IP).

5.2.2. UDP

Caractéristiques UDP :

Protocole niveau 4 (transport de bout en bout) en mode sans connexion, non fiable et sans garantie de séquencement (*datagram*). Une information d'état concernant chaque *socket* UDP est toutefois maintenue par le système (forme primitive de connexion).

UDP sert d'interface application pour IP, permettant le multiplexage/démultiplexage du trafic grâce aux numéros de ports qui identifient les processus destinataires.

Exemples d'ULPs utilisant UDP : NFS (*Network File System*), TFTP (*Trivial File Transfer Protocol*), SNMP (*Simple Network Management Protocol*), RPC (*Remote Procedure Call*).

Message UDP :

Port source	2 octets
Port destination	2 octets
Longueur	2 octets
Contrôle (optionnel)	2 octets
Données	variable

Port source, port destination : permettent d'identifier les processus de niveau supérieur qui communiquent ; le port source est optionnel, quand il n'est pas utilisé les deux octets sont à 0.

Longueur : longueur de la partie données du message UDP.

Contrôle : détection d'erreurs sur le segment entier, plus une partie le l'entête IP (notamment l'adresse IP source et l'adresse IP destination). Contrairement à TCP, pour UDP l'utilisation de ce champ est optionnelle.

5.3. La couche transport OSI (pour info…)

Caractéristiques couche transport OSI (ISO 8072, 8073) :

Protocoles orientés connexion : le transfert de TPDU (*Transport Protocol Data Unit*) — terminologie OSI — est précédé de l'établissement d'une connexion dont l'état est suivi en permanence (à un degré variable, selon la classe de transport choisie). Les connexions sont en duplex intégral (*full-duplex*). Une connexion est individualisée par la paire d'identificateurs [source, destination], chaque identificateur étant composé d'une adresse réseau et d'une référence transport (à un même TSAP — *Transport Service Access Point* — peuvent correspondre plusieurs références transport, une pour chaque connexion ouverte).

Le service de transport OSI segmente les données du niveau supérieur et émet les segments successifs. Les segments peuvent être de taille variable (la taille peut être choisie à l'établissement de la connexion).

Comme pour TCP, l'acquittement des TPDU de données et la synchronisation entre l'émetteur et le récepteur sont assurés par des mécanismes différents, permettant de séparer la gestion des tampons de réception de la gestion des numéros de séquence.

Contrairement à TCP, les acquittements sont toujours transmis dans des TPDU distinctes des TPDU de données.

Les autres caractéristiques, comme le contrôle et la reprise sur erreur, la numérotation des segments, le contrôle de flux, etc. dépendent de la classe de service choisie.

Structure générale TPDU OSI, protocoles orientés connexion :

Longueur en-tête	1 octet
Partie fixe	variable
Paramètres	variable
Données	variable

Longueur **en-tête** : 0÷254, à l'exclusion de l'octet longueur en-tête.

Paramètres : pour chaque paramètre, 1 octet d'identification, 1 octet de longueur et la valeur du paramètre sur un nombre variable d'octets.

TPDU d'établissement : les TPDU **CONNECT REQUEST** (CR) et **CONNECT CONFIRM** (CC) sont transférées à l'aide de primitives réseau N-DATA (DCE/DTE DATA pour X 25 PLP) ou N-EXPEDITED DATA (DCE/DTE INTERRUPT pour X 25 PLP). Structure :

Partie fixe : identification TPDU (4 bits, 1110 pour REQUEST et 1101 pour CONFIRM), crédit (4 bits, permet le contrôle de flux : une valeur de *n* indiquée par le récepteur autorise l'émetteur à envoyer *n* TPDU), référence destination (2 octets, 00 00h pour TPDU CONNECT REQUEST), référence source (2 octets), classe transport (4 bits), options (4 bits : 00**EN**, **E** = format normal (0) ou étendu (1) en classe 2, 3 ou 4 (format étendu : numéros de séquence sur les 7 bits du format normal **+ 3 octets**), **N** = utilisation (0) ou non (1) du contrôle de flux en classe 2).

Paramètres négociables de la connexion : taille TPDU, identification TSAP source et destination, numéro version, sécurité, données exprès, repli, délai acquittement, taux d'erreur résiduel, priorité, temps de transit, débit de réaffectation, débit.

Données : 32 octets, non autorisé en classe 0, optionnel dans les autres classes.

TPDU de libération : TPDU **DISCONNECT REQUEST** (DR) et TPDU **DISCONNECT CONFIRM** (DC, indisponible en classe 0). Structure :

Partie fixe : identification TPDU (4 bits, 1000 pour REQUEST et 1100 pour CONFIRM), 4 bits à 0, référence destination (2 octets), référence source (2 octets), cause déconnexion (1 octet).

Paramètres : définis par l'utilisateur !

Données : 32 octets ; non autorisé en classe 0, optionnel dans les autres classes.

TPDU de données normales **DATA TRANSFER** (DT), classes 0 et 1 :

Partie fixe : identification TPDU (octet F0H) ; en classe 1, 1 octet **Fsssssss** : **F** = fin TSDU (le contenu d'une TSDU de taille importante peut se retrouver dans plusieurs TPDU successives, pour la dernière **F** = 1), **sssssss** = numéro de séquence (modulo 128).

Données (fragment TSDU).

Remarque : la référence destination est absente, en classe 0 ou 1 le multiplexage étant indisponible !

TPDU de données normales classes 2 à 4 :

Partie fixe : identification TPDU (octet F0h), référence destination (2 octets), 1 octet **Fsssssss** : **F** = fin TSDU (le contenu d'une TSDU de taille importante peut se retrouver dans plusieurs TPDU successives, pour la dernière **F** = 1), **sssssss** = numéro de séquence ; en format étendu le numéro de séquence est complété par 3 octets supplémentaires.

Paramètres : en classe 4, un paramètre optionnel permettant le contrôle d'erreurs.

Données (fragment TSDU).

TPDU accusé de réception données normales **DATA ACKNOWLEDGE** (AK, indisponible en classe 0) :

Partie fixe : identification TPDU (4 bits, 0110) ; le crédit sur 4 bits en format normal (classes 2 à 4 uniquement, 1111 en classe 1) ou 0000 en format étendu ; référence destination (2 octets) ; 1 octet 0**sssssss** = numéro d'acquittement en format normal (indique la **prochaine** TPDU attendue), complété en format étendu par 3 octets supplémentaires ; en format étendu, le crédit sur 2 octets (classes 2 à 4 uniquement).

Paramètres : en classe 4, un paramètre optionnel permettant le contrôle d'erreurs.

TPDU de données prioritaires **EXPEDITED DATA** (ED) et accusés de réception de données prioritaires **EXPEDITED DATA ACK** (EA) : disponibles en classes 3 et 4 ; optionnelles en classes 1 et 2 (négociation à l'établissement de la connexion). La structure est similaire à celle des TPDU de données/accusés de réception de données normales, mais le champ données utilisateur contient au maximum 16 octets. Les formats normal et étendu sont disponibles. Remarque : les numéros de séquence/d'acquittement utilisés sont spécifiques aux données prioritaires et n'ont aucun rapport avec les numéros de séquence des données normales !

TPDU de rejet **REJECT** (RJ) : classes 1 et 3 uniquement. Les formats sont similaires aux formats des TPDU AK, mais le numéro présent indique le rejet et non pas l'acquittement.

Une TPDU d'erreur **ERROR** (ER) rejette une TPDU invalide. Structure :

Partie fixe : identification TPDU (octet 70h) ; référence destination (2 octets) ; cause du rejet (1 octet) : type TPDU non valide (02h), code paramètre non valide (01h), valeur paramètre non valide (03h), cause non spécifiée (00h).

Paramètres : un seul paramètre décrit par {code C1h, longueur paramètre, valeur paramètre = configuration binaire de l'en-tête de la TPDU rejetée, y compris la partie qui a provoqué le rejet}.

Primitives couche transport ISO ↔ ULP (*Upper Layer Protocol*), protocole transport orienté connexion :

T-CONNECT.request et T-CONNECT.indication — paramètres : adresses appelé et appelant, option données exprès, paramètres qualité de service (à négocier), données utilisateur (maximum 32 octets).

T-CONNECT.response et T-CONNECT.confirm — paramètres : adresse en réponse, option données exprès, paramètres qualité de service (négociés), données utilisateur (maximum 32 octets).

T-DATA.request et T-DATA.indication — paramètres : données utilisateur.

T-EXPEDITED-DATA.request et T-EXPEDITED-DATA.indication — paramètres : données exprès utilisateur, 16 octets maximum.

T-DISCONNECT.request et T-DISCONNECT.indication — cause de déconnexion, données utilisateur (maximum 32 octets).

Primitives couche transport ISO ↔ ULP, protocole transport sans connexion :

T-UNIDATA.request et T-UNIDATA.indication — paramètres : adresses appelé et appelant, option données exprès, paramètres qualité de service (à négocier), données utilisateur.

6. La couche session

Une couche session distincte à été proposée pour la première fois par le modèle OSI (normes ISO 8326, 8327).

6.1. Fonctionnalités de la couche session

La couche session fournit à l'utilisateur des services facilitant le développement d'applications : gestion du dialogue (duplex, semi-duplex), points de synchronisation et opérations de resynchronisation, définition et gestion d'activités, transfert/préparation de rapports d'anomalies. Certains de ces services ne sont pas entièrement définis dans les normes ISO concernant la couche session, mais dépendent en partie des couches supérieures.

En mode orienté connexion, trois situations sont acceptées : une connexion de transport pour une connexion session, plusieurs connexions de transport successives pour une connexion session et plusieurs connexions session successives pour une même connexion de transport. En revanche, le multiplexage de plusieurs connexions session sur une même connexion de transport n'est pas permis.

6.1.1. Transfert des données

En mode orienté connexion, la libération normale d'une connexion est une libération **négociée** (contrairement au cas d'une connexion de transport) : même si un des deux correspondants fait une demande de déconnexion en utilisant la primitive S-RELEASE.request, l'autre correspondant peut refuser cette demande (refus indiqué par un paramètre de la primitive S-RELEASE.response).

Deux nouveaux types de données peuvent être transférées par la couche session (en plus des données normales et données exprès) : les données **typées** et les données **de capacité**. Les données typées ne sont pas soumises au mécanisme de synchronisation et permettent notamment de transférer des informations de contrôle des couches supérieures (leur utilisation précise n'est pas spécifiée dans les normes ISO concernant la couche session). Les données de capacité sont soumises à un mécanisme de synchronisation forte (ne peuvent être transmises qu'en dehors des activités) et permettent notamment de modifier les options et les paramètres en cours de session (leur utilisation précise n'est pas spécifiée dans les normes ISO concernant la couche session).

6.1.2. Gestion du dialogue

Les connexions session peuvent être en duplex intégral ou en semi-duplex. Certaines applications sont plus faciles à développer si les connexions fonctionnent en mode semi-duplex (cela n'a rien à voir avec les éventuelles limitations des couches inférieures !) et la couche session permet de l'assurer. La gestion du dialogue utilise un jeton, afin d'émettre une entité session doit être en possession du jeton ; des primitives spécifiques sont employées pour demander/céder les différents jetons employés (voir plus loin).

6.1.3. Synchronisation

La synchronisation entre deux interlocuteurs consiste en un maintien d'états de référence connus par les deux interlocuteurs. En cas de défaillance temporaire d'un élément **d'une couche supérieure** à la couche session (par exemple, bourrage de papier dans l'imprimante...), cela permet de reprendre la session à partir d'un état bien identifié et connu par les deux interlocuteurs — processus appelé resynchronisation. Les points de synchronisation peuvent être majeurs (deux points majeurs définissent un dialogue) ou mineurs ; les points de synchronisation majeure doivent être confirmés explicitement, ce qui n'est pas le cas pour les autres. Aussi, en cas de resynchronisation, on ne peut pas faire marche arrière au-delà du dernier point de synchronisation majeure. Enfin, deux jetons indépendants sont associés aux deux types de points de synchronisation.

6.1.4. Gestion des activités

L'utilisateur peut employer un découpage logique du flot de données en **activités** (leur utilisation précise n'est pas spécifiée dans les normes ISO concernant la couche session). Parmi les possibilités offertes par la gestion des activités est la **mise en quarantaine** (mise en oeuvre dans la couche application, mais utilisant le service de gestion des activités de la couche session) : l'accumulation de tous les messages correspondant à un traitement indivisible (par exemple une transaction) avant de traiter chaque message.

Le début de chaque activité est automatiquement délimité par un point de synchronisation majeure, donc on ne peut pas faire marche arrière au-delà du début de l'activité. En revanche, une activité peut être interrompue

et reprise après d'autres transferts ! Un jeton activité est défini, mais pour démarrer une activité il faut être en possession du jeton d'activité, du jeton de synchronisation majeure et du jeton de données (si le dialogue est en semi-duplex).

6.1.5. Rapports d'anomalies

La couche session peut transmettre des rapports d'anomalies (mis au point par des couches supérieures) entre les deux interlocuteurs et peut aussi préparer ses propres rapport d'anomalies qu'elle présente à l'utilisateur.

6.2. Types de SPDU et primitives de service

Primitives couche session OSI ↔ ULP (*Upper Layer Protocol*) :

Primitive	*Rq.*	*Ind.*	*Response*	*Confirm*	Signification
S-CONNECT	X	X	X	X	Établissement d'une connexion
S-RELEASE	X	X	X	X	Libération ordonnée d'une session
S-U-ABORT	X	X			Libération brutale à l'initiative de l'utilisateur
S-P-ABORT		X			Libération brutale à l'initiative du fournisseur
S-DATA	X	X			Transfert de données normales
S-EXPEDITED-DATA	X	X			Transfert de données exprès
S-TYPED-DATA	X	X			Transfert de données typées
S-CAPABILITY-DATA	X	X	X	X	Transfert de données de capacité
S-TOKEN-GIVE	X	X			Passage d'un jeton
S-TOKEN-PLEASE	X	X			Jeton réclamé
S-CONTROL-GIVE	X	X			Passage de tous les jetons
S-SYNC-MAJOR	X	X	X	X	Insertion d'un point de synchronisation majeure
S-SYNC-MINOR	X	X			Insertion d'un point de synchronisation mineure
S-RESYNCHRONIZE	X	X	X	X	Retour à un point de synchronisation précédent
S-ACTIVITY-START	X	X			Démarrage d'une activité
S-ACTIVITY-END	X	X	X	X	Fin d'une activité
S-ACTIVITY-DISCARD	X	X	X	X	Abandon d'une activité
S-ACTIVITY-INTERRUPT	X	X	X	X	Suspension d'une activité
S-ACTIVITY-RESUME	X	X			Reprise d'une activité suspendue
S-U-EXCEPTION-REPORT	X	X			Rapport d'anomalie utilisateur
S-P-EXCEPTION-REPORT		X			Rapport d'anomalie fournisseur

7. La couche présentation

7.1. Fonctionnalités de la couche présentation OSI

Principales fonctionnalités de la couche présentation :

Fournir une interface entre la couche application et la couche session : l'interface concerne la négociation des paramètres du contexte de présentation et la traduction de syntaxe.

Fournir une syntaxe uniforme pour les transferts entre plates-formes hétérogènes et assurer les conversions syntaxiques : une syntaxe unifiée (ASN.1, *Abstract Syntax Notation 1*, ISO 8824, 8825) est employée pour décrire les données transférées entre deux entités présentation ; chaque entité de présentation traduit les données reçues, en fonction de leur structure (définie en respectant ASN.1), en une suite binaire (respectant la syntaxe de transfert). La couche application indique, pour chaque APDU (*Application Protocol Data Unit*) à transmettre, la description formelle en ASN.1 des données. Les structures de données utilisées pour les informations à transférer sont spécifiées à l'aide de bibliothèques de structures. Types primitifs d'ASN.1 : INTEGER, BOOLEAN, BIT STRING, OCTET STRING, ANY, NULL, OBJECT IDENTIFIER. Constructeurs d'ASN.1 : SEQUENCE, SEQUENCE OF, SET, SET OF, CHOICE. Format d'un enregistrement transféré : type étiquette (2 bits : UNIVERSAL, PRIVATE, spécifique au contexte, PRIVATE), type primitif/construit (1 bit), étiquette type (5 bits, avec éventuellement des octets supplémentaires), longueur du champ de données, champ de données, fanion de fin si la longueur n'est pas indiquée.

Assurer (optionnellement) la compression des informations (la compression peut être effectuée aussi dans d'autres couches) : l'identité de l'algorithme employé fait partie des paramètres de contexte négociés par les deux entités de présentation.

Assurer (optionnellement) le cryptage/authentification/signature des messages (le cryptage peut être effectué aussi dans d'autres couches) : l'identité de l'algorithme employé fait partie des paramètres de contexte négociés par les deux entités de présentation.

7.2. Primitives de service et types de PPDU OSI

A l'exception des trois dernières, les primitives présentation (ISO 8823) traversent telles quelles la couche et sont traduites en primitives correspondantes session.

P-ALTER-CONTEXT.request, .indication, .response et .confirm : permettent à l'utilisateur du service de demander des changements de contexte (liste de bibliothèques de structures, identification méthode de compression/cryptage/authentification/signature) en cours de session, une négociation est alors menée entre les deux entités présentation. En option, la couche présentation peut conserver de multiples contextes afin de faciliter les changement de contexte.

P-U-EXCEPTION-REPORT.request et .indication : la couche présentation peut transmettre entre les deux interlocuteurs des rapports d'anomalies mis au point par les couches supérieures.

P-P-EXCEPTION-REPORT.indication : la couche présentation peut aussi préparer ses propres rapports d'anomalies (détectées dans la couche elle-même ou dans des couches inférieures) et les soumettre à l'utilisateur.

Observation : des primitives de déconnexion normale existent (P-RELEASE), mais aucune PPDU de déconnexion n'est définie ; une connexion de présentation est considérée terminée quand la connexion session sur laquelle elle repose est terminée.

8. La couche application

8.1. Fonctionnalités et structure

La couche application — la plus riche du modèle OSI — fournit des services très courants aux applications (situées au-dessus de la couche application OSI) et/ou utilisateurs, services parmi lesquels on peut citer :

Mise en association de deux processus applicatifs distants (établissement d'une connexion de niveau application) : ACSE (*Association Control Service Element*, normes X 217/227, IS 8649/8650).

Engagement, concurrence et reprise des transactions en cas de panne : CCRSE (*Commitment, Concurrency and Recovery*, normes IS 9804/9805).

Transfert de fichiers et gestion des accès à distance : FTAM (*File Transfer, Access and Management*, norme IS 8571).

Messagerie électronique : MHS (*Message Handling System*, normes X 400, IS 10021).

Transfert fiable des APDU (*Application Protocol Data Unit*), les pannes éventuelles sont transparentes à l'utilisateur : RTSE (*Reliable Transfer Service Element*, normes X 218/228 et IS 9066).

Terminal virtuel : VTP (*Virtual Terminal Processing*, normes IS 9040/9041).

Service d'annuaire : DSE (*Directory Service Element*, normes X 500, IS 9594).

Échange de documents informatisé : EDI (*Electronic Data Interchange*, norme IS 9735).

Chaque service est assuré par une composante spécifique — possédant ses propres primitives — de la couche application. Certains services (comme ACSE ou CCRSE) sont employés aussi par d'autres service de la couche d'application (voir FTAM plus loin). En général, une application utilise plusieurs services de la couche application.

8.2. Composantes application OSI (pour info…)

8.2.1. ACSE

Au niveau application, les connexions s'appellent associations. ACSE permet d'établir des associations de niveau application avec authentification des accès. La traduction primitive ACSE ↔ primitive présentation est directe. Primitives ACSE ↔ UL (*Upper Layer*) :

A-ASSOCIATE.request, .indication, .response et .confirm : établissement d'une association avec un processus distant.

A-RELEASE.request, .indication, .response et .confirm : demande de libération normale par l'utilisateur du service (application ou utilisateur).

A-ABORT.request et .indication : demande de libération brutale par l'utilisateur du service.

A-P-ABORT.indication : libération brutale à l'initiative de la couche application.

8.2.2. CCRSE

CCRSE permet de coordonner de façon fiable (même en cas de panne de certains systèmes) des interactions entre des sites multiples, dont un qui est maître et les autres sont esclaves. Dans le cadre d'une opération, le maître envoie des requêtes aux esclaves et s'assure que tous les esclaves peuvent exécuter les requêtes ; si au moins un

des esclaves est incapable d'exécuter sa requête, le maître annule toutes les requêtes de l'opération. Primitives CCRSE ↔ UL (*Upper Layer*) :

C-BEGIN : le maître indique aux esclaves le début de l'opération ; les requêtes sont envoyées immédiatement après.

C-PREPARE : le maître indique aux esclaves de préparer un engagement sur la requête.

C-READY : un esclave indique au maître qu'il s'engage à effectuer la requête (la requête est enregistrée sur un support fiable et les données sont verrouillées).

C-REFUSE : un esclave indique au maître qu'il est dans l'impossibilité de s'engager à effectuer la requête.

C-COMMIT : si tous les esclaves ont répondu par C-READY, le maître leur demande d'exécuter la requête sur laquelle ils se sont engagés.

C-ROLLBACK : si au moins un des esclaves n'a pas répondu par C-READY, le maître demande aux esclaves de ne pas prendre en compte la requête (de revenir à l'état initial).

C-RESTART : le maître ou un des esclaves indique qu'il a dû redémarrer après une panne. Dans certaines situations, l'opération entière est redémarrée.

8.2.3. FTAM

Le transfert, l'accès et la gestion de fichiers distants utilisent un modèle de fichier virtuel, mis en correspondance avec un fichier réel sur la machine hôte grâce à la spécification de contraintes. Les contraintes permettent d'adapter la structure hiérarchique très générale du modèle à la structure souvent plus particulière des fichiers réels (fichiers non structurés, fichiers séquentiels, etc.).

Le traitement d'un fichier virtuel est orienté connexion et consiste en une série de phases imbriquées :

Phase d'association : après l'établissement d'une connexion (F-INITIALIZE établit une connexion en appelant A-ASSOCIATE de ACSE). Des opérations de gestion du fichier sont possibles. La phase (et la connexion ouverte) est terminée par F-TERMINATE ou F-ABORT (utilisable à tout moment).

Phase de sélection du fichier : après la sélection (F-SELECT) ou la création (F-CREATE) du fichier ; des verrous peuvent être posés. Les attributs du fichier sélectionné peuvent être lus et modifiés (F-READ-ATTRIB, F-CHANGE-ATTRIB) et le fichier peut être ouvert. La phase se termine par F-DESELECT ou F-DELETE.

Phase d'ouverture du fichier : après l'ouverture du fichier (F-OPEN) ; des verrous peuvent être posés. Des données peuvent être transférées. La phase se termine par F-CLOSE.

Phase de transfert de données : entre F-READ ou F-WRITE et F-TRANSFER-END.

Pour le contrôle des accès concurrents, deux types de verrous peuvent être employés : verrous non exclusifs (utilisés pour la lecture) et verrous exclusifs (utilisés pour l'écriture) ; un verrou non exclusif peut être posé sur un fichier si aucun verrou exclusif n'est présent, un verrou exclusif peut être posé si aucun verrou (exclusif ou non exclusif) n'est présent.

Un service fiable même en cas de panne de l'utilisateur ou du serveur peut être demandé à l'aide de paramètres de F-OPEN ; le service fiable utilise les primitives F-BEGIN-GROUP et F-END-GROUP pour regrouper plusieurs primitives dans une action atomique, F-CHECK pour mettre en place des points de contrôle (points de synchronisation dans la couche session), F-RESTART (après perturbations mineures) et F-RECOVER (après un incident majeur comme une panne) permettent de revenir au dernier point de contrôle.

8.2.4. MHS

Le service de messagerie permet de transférer de façon fiable du courrier entre les utilisateurs, sans nécessiter la présence simultanée des utilisateurs devant leur terminal. MHS fait la distinction entre enveloppe et message.

MHS est composé de deux types d'entités, les agents utilisateurs (UA, *User Agent*) et les agents de transfert de messages (MTA, *Message Transfer Agent*). Un agent utilisateur gère l'interface avec l'utilisateur, la mémoire de messages et le dialogue avec le système de transfert. Les agents de transfert (MTA) prennent en charge l'acheminement des messages entre les agents utilisateurs de l'expéditeur et du destinataire ; plusieurs MTA intermédiaires peuvent être traversés par un message entre l'expéditeur et le destinataire.

MHS n'emploie pas directement des adresses pour identifier l'expéditeur et le destinataire, mais des "noms" (références) utilisé(e)s par le service d'annuaire DSE, comme par exemple :

```
COUNTRY=CANADA
ORG=VIDEOTRON
DEPT=PUBLIC RELATIONS
NAME=STEVE ROBSON
```

C'est le service annuaire qui est chargé de récupérer l'adresse du destinataire utilisable par les couches inférieures.

Étapes dans l'envoi de courrier : 1° sous le contrôle de l'agent utilisateur, l'expéditeur construit le message et indique les valeurs des paramètres de l'en-tête et de l'enveloppe ; 2° l'agent utilisateur passe le message à l'agent de transfert de message (primitive SUBMIT.request) ; 3° l'agent de transfert vérifie l'enveloppe et l'en-tête du message, récupère l'adresse du MTA suivant et utilise le service de transfert fiable pour lui transmettre le

message. Chaque message est envoyé dans une activité dédiée de la couche session, le mécanisme de récupération est employé en cas de panne ; des points de synchronisation mineure sont utilisés pour les messages de longueur importante.

Primitives RTSE (fonctionnement en semi-duplex) ↔ UL (*Upper Layer*) :

RT-OPEN.request, .indication, .response et .confirm : établissement d'une association (fait appel à A-ASSOCIATE). **RT-CLOSE** libère l'association.

RT-U-ABORT.request et .indication : interruption à l'initiative de l'utilisateur du service. **RT-P-ABORT**.indication : interruption à l'initiative du service.

RT-TRANSFER.request, .indication, .response et .confirm : transfert fiable d'un message.

RT-TURN-PLEASE.request et .indication : une entité RTSE demande le droit d'émettre. **RT-TURN-GIVE**.request et .indication : le droit d'émettre est accordé par l'autre entité RTSE.

8.3. Composantes application dans l'environnement TCP/IP

La couche application de l'environnement TCP/IP contient un nombre important de protocoles développés par différents constructeurs et adoptés ensuite par d'autres. Les protocoles sont présentés dans des documents appelés RFC (*Request For Comments*) mais les différences entre les implémentations sont parfois importantes.

8.3.1. TELNET

TELNET (*TELecommunication NETwork*, RFC 854) permet de se connecter à une machine distante en émulation de terminal. Par défaut, un client `telnet` établit une connexion avec un serveur `telnetd` (sur le port TCP 23), mais peut être utilisé afin de se connecter à un service différent (numéro de port différent) — par exemple `httpd` (port TCP 80). Un terminal virtuel (NVT, *Network Virtual Terminal*) est défini afin de résoudre les problèmes de hétérogénéité des équipements ; une traduction entre les caractéristiques locales et NVT est effectuée à chaque bout.

Une négociation d'options TELNET (par exemple transmission binaire, écho, etc.) **ou autres que TELNET** a lieu entre le client et le serveur à l'établissement de la connexion ; la négociation consiste en une suite de messages concernant les différentes options. Une authentification de l'utilisateur (nom et mot de passe) a aussi lieu à l'établissement de la connexion.

Format des commandes TELNET :

Premier octet : IAC (*Interpret As Command*) — indique que l'octet suivant est une commande.

Deuxième octet : code de la commande (voir le tableau).

Troisième octet, optionnel : code de l'option négociée.

8.3.2. FTP

FTP (*File Transfer Protocol*, RFC 959) permet de transférer et/ou de gérer des fichiers entre des machines distantes. Un client `ftp` établit une connexion (port TCP 21, appelé *ftp*) avec un serveur `ftpd` ; cette connexion permet d'échanger des commandes et réponses concernant les transferts. Une **deuxième connexion** (port TCP 20, appelé *ftp_data*) peut être établie entre l'ordinateur local et l'ordinateur distant permettant le transfert des données des fichiers (ou des contenus des répertoires) ; cette deuxième connexion est contrôlée par les commandes qui transitent sur la première connexion.

8.3.3. RPC

RPC (*Remote Procedure Call*, RFC 1057) permet d'effectuer des appels de procédures qui s'exécutent sur des machines distantes ; le comportement des appels distants est similaire à celui des appels locaux. RPC peut être implémenté au-dessus de UDP (cas typique) ou de TCP. Un appel RPC contient un identificateur de la procédure distante à exécuter (adresse de la machine distante, numéro de la procédure distante et son numéro de version). La réponse contient le résultat de l'appel.

8.3.4. NFS

NFS (*Network File System*, RFC 1094) permet à plusieurs machines connectées à un réseau (en général local) de partager des fichiers (les transferts de fichiers entiers entre les machines deviennent inutiles). NFS utilise les services de RPC et de XDR (*eXternal Data Representation*, protocole qui correspond approximativement à la couche présentation du modèle OSI). Remarque : NFS utilise XDR uniquement pour les éléments du protocole, pas pour les données transférées ! NFS ne garde aucune information concernant l'état des opérations en cours ; cette information doit être maintenue et gérée par les applications/utilitaires qui font appel à NFS. En revanche, un serveur peut être arrêté et redémarré sans que le fonctionnement des clients soit gravement perturbé — les clients sont tout simplement mis en attente.

NFS est construit sur le modèle client-serveur ; une machine peut être uniquement client, uniquement serveur ou client et serveur à la fois. Un système de fichiers virtuel est implémenté afin de rendre transparent l'accès à tous les fichiers, locaux ou distants. Dans une première étape, un système de fichiers distants est rendu accessible localement ; les fichiers du système de fichiers distant peuvent ensuite être manipulés comme des fichiers locaux. Les opérations qui peuvent être effectuées : 1° création de fichier, déplacement et changement de

nom ; 2° lecture et modification des attributs des fichiers ; 3° création, lecture et destruction de répertoires ; 4° lecture de/écriture dans un fichier. Un seul type de verrou peut être posé sur un fichier.

8.3.5. SMTP

Comme pour MHS dans le modèle OSI, le service de courrier électronique est composé de deux types d'entités, les agents utilisateurs (UA) et les agents de transfert de messages (MTA, `smtpd`). Contrairement à MHS, les MTA de l'expéditeur et du destinataire communiquent sans intermédiare au niveau application.

SMTP (*Simple Mail Transfer Protocol*) définit la structure des messages (RFC 822 ; il n'y a pas d'enveloppe distincte comme pour MHS) et le protocole de communication entre MTA (RFC 821).

En-tête d'un message SMTP :

La date et l'heure d'expédition du courrier (rempli par le MTA de l'expéditeur)
Le sujet (à remplir par l'expéditeur ou par le UA de l'expéditeur en cas de réponse)
L'adresse du destinataire (à remplir par l'expéditeur ou par le UA de l'expéditeur en cas de réponse)
La date et l'heure de réception (rempli par le MTA du destinataire)
Nom et identification de l'expéditeur (rempli par l'UA de l'expéditeur)
Nom et version de l'UA expéditeur
Type du contenu et jeu de caractères utilisé (rempli par l'UA de l'expéditeur)
Longueur du courrier en nombre de caractères (rempli par l'UA de l'expéditeur)

Exemple d'en-tête de message :

```
From thomas Mon May 27 10:15 1996
Subject: Current project
To: steve@media.mit.edu
Date: Mon May 27 1996 10:16:03
From: "Thomas Watson" <thomas@cs.mit.edu>
X-Mailer: ELM [version 2.4 PL22]
Content-Type: text/plain; charset=ISO-8859-1
Content-Length: 123
```

SMTP utilise un adressage de type `utilisateur@sous-domaine.domaine`, la partie de l'adresse qui suit le symbole @ étant transformée par un serveur de noms en adresse IP unicast. Les MTA prennent en compte des alias définis dans des fichiers spécifiques ; ces alias permettent d'associer un nom d'alias (par exemple `postmaster`) à un ou plusieurs utilisateurs et donc de mettre en place des listes de distribution.

8.3.6. Finger

Finger (RFC 1196) est un protocole très simple (et en général aussi le nom d'une commande utilisateur), de type question-réponse, qui permet d'obtenir des informations concernant les utilisateurs *logged* sur une station. Ces informations diffèrent en général selon les implémentations et les utilisateurs ; les implémentations sous UNIX permettent en général à un utilisateur de spécifier différentes informations dans des fichiers `.plan` et `.project`, informations qui seront fournies en réponse à `finger` <utilisateur> <station>.

8.3.7. Ping

Ping est un protocole très simple (et en général aussi le nom d'une commande utilisateur) qui emploie les messages ICMP *echo* et *echo-reply* afin de s'assurer du bon fonctionnement d'une ressource sur le réseau. Plus précisément, *ping* peut vérifier le bon fonctionnement de la couche TCP/IP sur la station locale, le fonctionnement d'une station ou d'un routeur distant, l'adresse IP de la station locale ou l'accès au réseau. Certaines implémentations permettent aussi de recueillir des statistiques concernant la connexion en question.

8.3.8. SNMP

SNMP (*Simple Network Management Protocol*, RFC 1157, 1215, 1187, 1212, 1213, 1441, 1121, 1142) a été développé comme un protocole simple pour la gestion de réseaux IP. Un autre protocole plus complexe, CMOT (*Common Management information services and protocol Over Tcp/ip*, RFC 1095), existe mais est nettement moins utilisé pour le moment.

SNMP permet le suivi et le contrôle centralisé des différentes composantes d'un réseau (routeurs, *gateways*, équipements de transmission, etc.) plus ou moins étendu. Contrairement à CMOT qui utilise les services de TCP et UDP, SNMP est implémenté au-dessus de UDP, ce qui peut poser quelques problèmes pour des réseaux complexes.

Chaque composante du réseau possède un agent SNMP et une MIB (*Management Information Base*) qui contient la description des différents attributs de la composante. Groupes d'objets (une composante est un ensemble d'objets) et leurs attributs spécifiques :

Groupe système : nom et version du hardware, système d'exploitation, logiciels réseau, dernière initialisation.

Groupe interfaces : nombre d'interfaces réseau, type d'interface dans la couche inférieure (HDLC-LAPB, Ethernet, etc.), taille du datagramme accepté, vitesse en bits/s, adresse, état opérationnel, volume de trafic reçu/livré/invalidé (avec les raisons).

Groupe IP : datagrammes en transit ou pas, durée de vie des datagrammes, volume de trafic reçu/livré/invalidé (avec les raisons), informations concernant la fragmentation, tables d'adresse (masques comprises), tables de routage et protocoles de routage employés.

Groupe ICMP : nombre des différents types de messages ICMP émis ou reçus, statistiques concernant les problèmes rencontrées.

Groupe TCP : algorithme de retransmission et valeur maximale/minimale, nombre maximal de connexions TCP supportées, informations concernant le trafic, *socket* (adresse IP + port) pour chaque connexion.

Groupe UDP : informations concernant le trafic et les problèmes rencontrés.

Groupe EGP (*External Gateway Protocol*) : informations concernant le trafic et les problèmes rencontrés, table des voisins EGP, adresses des voisins, état EGP par rapport à chaque voisin.

Les entités de gestion du réseau rassemblent dans leurs MIB l'ensemble des informations concernant les composantes du réseau se trouvant sous leur autorité de gestion ; un agent de contrôle SNMP est présent dans chaque entités de gestion. Chaque MIB définit des droits d'accès des différents agents SNMP aux différents objets de la base. Des applications de gestion spécifiques à chaque constructeur sont associées aux agents SNMP et font appel à leurs services.

Types de messages SNMP :

Get Request : demande la valeur associée à une variable ; en terminologie SNMP, une "variable" est une composante (un objet) du réseau géré et la "valeur" un descripteur complet de l'état de la composante.

Get Next Request : demande la valeur associée à la variable suivante de la MIB.

Get Bulk (SNMPv2) : demande les valeurs associées à un ensemble de variables de la MIB (équivalent à une succession de *Get Request*).

Set Request : demande la modification de la valeur d'une variable gérée (message de contrôle).

Get Response : réponse à une demande d'information (*get*) ou de modification (*set*). Des codes d'erreur ainsi que des informations additionnelles peuvent être présentes.

Trap : message permettant à une composante gérée de signaler un événement ; SNMPv2 permet, en option, d'employer des messages de confirmation de réception des *trap*.

Inform Request (SNMPv2) : messages facilitant la communication entre les administrateurs de sous-réseaux.

SNMPv2 (RFC 1441) spécifie différents mécanismes de sécurité (authentification et intégrité, renforcement de la protection des accès, confidentialité) dont le but est de rendre sûre l'activité de gestion du réseau.

8.3.9. Internet et le *World Wide Web*

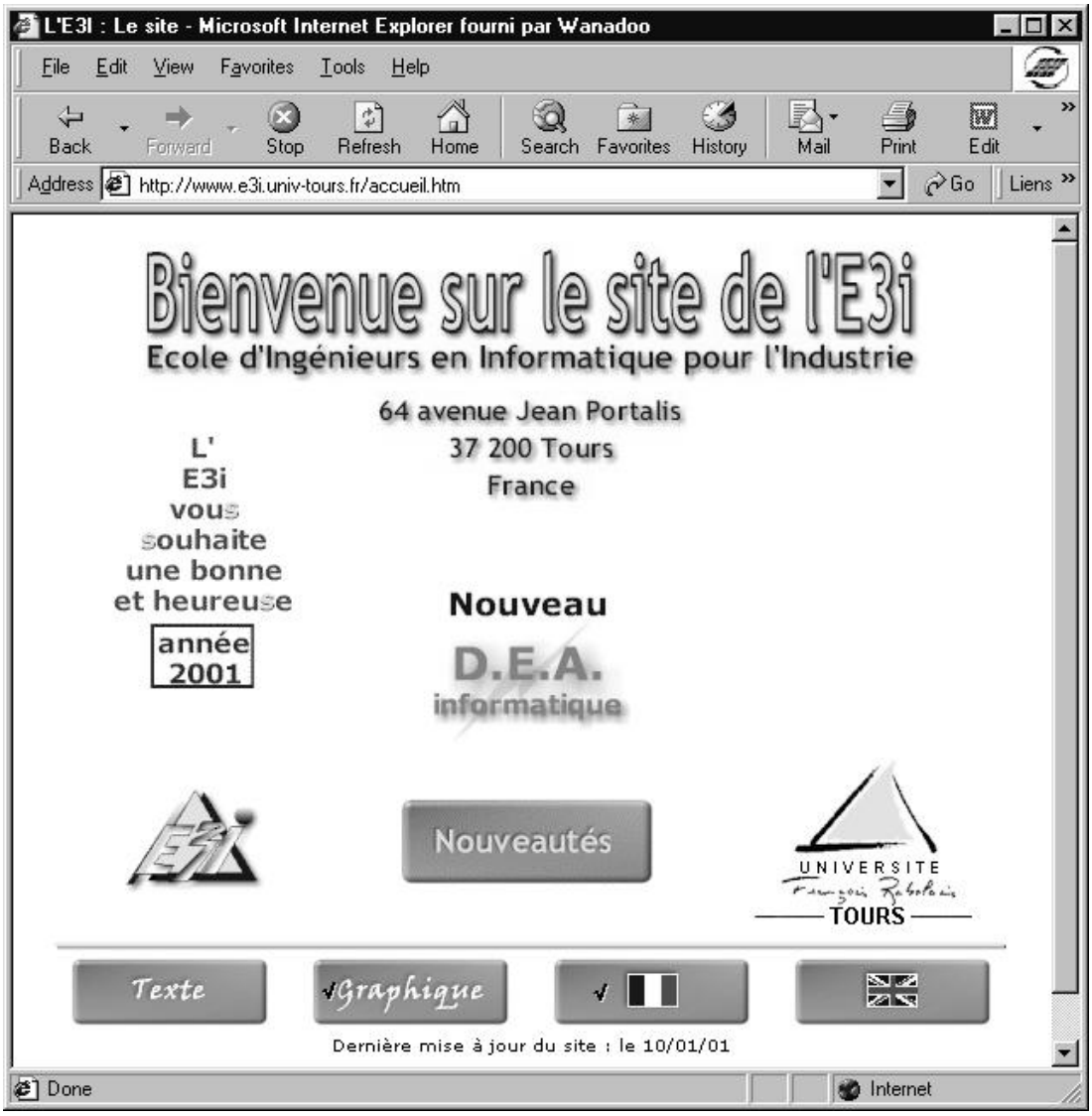
World Wide Web (*web* = toile d'araignée) : appellation donnée à l'ensemble des documents hypertexte accessibles sur Internet. Hypertexte = texte comportant des liens vers d'autres textes. HTML (*HyperText Markup Language*) est le principal langage de description de tels documents. Les documents hypertexte peuvent contenir des liens vers différents types de ressources : autres documents hypertexte, images, séquences sonores, séquences animées, programmes exécutables. Les liens contiennent l'adresse de la ressource référencée sous forme d'URL (*Uniform Resource Locator*).

Exemple de document HTML :

```
<html><head>
<title>L'E3I : Le site </title>
</head>
<body bgcolor="#FFFFFF" vlink="#808080" alink="#FF00FF">
<table border="0" width="100%">
  <tr><td width="718" colspan="3"><p align="center"><map name="FPMap0">
      <area   href="UN/Index.htm"   shape="circle"   coords="450,   8,   5"></map><img
src="images/Bnvenue2Haut.jpg" alt="Bienvenue sur le site de l'Ecole d'Ingénieurs en
Informatique    pour    l'Industrie"    border="0"    usemap="#FPMap0"    width="458"
height="82"></td>
  </tr>
  <tr>
    <td      width="35%"      rowspan="2"><p      align="center"><img      border="0"
src="images/Happy2001Anim.gif" width="105" height="161"></td>
    <td  width="30%"  valign="top"><p  align="center"><img  src="images/Bnvenue2Bas.jpg"
alt="Bienvenue sur le site de l'Ecole d'Ingénieurs en Informatique pour l'Industrie"
width="184" height="71"></td>
    <td width="35%">
      <p align="right"></td>
  </tr>
  <tr>
    <td width="30%" valign="top">
      <p align="center"> </p>
      <p align="center"><a href="http://www.e3i.univ-tours.fr/dea" target="_blank"><img
name="DEA"   src="images/DEA0.jpg"   alt="Vers   le   site   du   DEA   d'Informatique   de
l'Université    de    Tours"    onMouseOver="document.DEA.src    =    'images/DEA1.jpg';"
onMouseOut="document.DEA.src     =     'images/DEA0.jpg';"     border="0"     width="119"
height="95"></a></td>
```

```
    <td width="35%"></td>
  </tr>
</table>
<table border="0" width="100%" cellspacing="0" cellpadding="5">
  <tr>
    <td width="33%"><p align="center"><img src="images/E3iLogoShadow.jpg" width="88"
height="66"></td>
    <td width="33%"><font face="Verdana"><p align="center"><a href="NEWS.HTM"><img
src="images/New2.gif" alt="Les nouveautés" border="0" width="150"
height="52"></a></font></td>
    <td width="34%"><p align="center"><a href="http://www.univ-tours.fr"
target="_blank"><img src="images/UFRLogo.jpg" alt="Université François Rabelais -
Tours" border="0" width="136" height="100"></a></td>
  </tr>
</table>
<div align="center"><center>
<table border="0" width="75%">
  <tr>
    <td valign="top" width="100%" colspan="4"><p align="center"><img
src="images/Sep.jpg" alt width="563" height="6"></td>
  </tr>
  <tr>
    <td valign="top" width="25%" align="center"><a href="INDEXTXT.HTM"
target="_top"><img src="images/BtnTxtNo2.jpg" alt="Version texte" border="0"
width="117" height="41"></a></td>
    <td valign="top" width="25%" align="center"><img src="images/BtnGraYes2.jpg"
alt="Version graphique" width="117" height="41"></td>
    <td valign="top" width="25%" align="center"><img src="images/BtnFrYes2.jpg"
alt="Version française" width="117" height="41"></td>
    <td valign="top" width="25%" align="center"><a href="Site_E3i_GB/INDEX.GB.HTM"
target="_top"><img src="images/BtnGbNo2.jpg" alt="English version" border="0"
width="117" height="41"></a></td>
  </tr>
  <tr><td valign="top" width="100%" colspan="4"><p align="center"><font face="Verdana"
size="1">Dernière mise à jour du site : le <!--webbot bot="Timestamp" S-Type="EDITED"
S-Format="%d/%m/%y" startspan -->10/01/01<!--webbot bot="Timestamp" endspan i-
checksum="12349" --></font></td>
  </tr>
</table></center></div>
</body></html>
```

Image du document HTML obtenue à l'aide d'un *browser* (programme de visualisation et navigation) :



8.3.9.1. HTTP

L'accès à des documents hypertexte distants est assuré par HTTP (*HyperText Transfer Protocol* v1.1 : RFC 2068, `http://www.ietf.org/rfc/rfc2068.txt`), un protocole de niveau application de l'environnement TCP/IP. HTTP peut toutefois être utilisé pour le transfert de données de nature très diverse.

HTTP est construit sur le modèle client-serveur. Un module client fait partie d'un logiciel spécialisé (*browser*) ou est inclus dans un autre logiciel comme un traitement de texte ; le client prend en charge la formulation de requêtes adressées aux serveurs, ainsi que l'interprétation des réponses (dont les documents hypertexte en langage HTML). Le client fait en général appel à des outils externes afin de restituer à l'application qui en fait l'usage ou directement à l'utilisateur les informations contenues dans des fichiers de formats très divers référencés par la page HTML courante (images autres que `.GIF`, séquences sonores, séquences animées).

Un serveur (processus `httpd`, port TCP 80) répond aux requêtes des clients en retournant une réponse, après avoir éventuellement lancé en exécution un programme externe et récupéré ses résultats. Un serveur HTTP (spécialement configuré) peut aussi servir d'intermédiaire :

Proxy — intermédiaire coté client, permet de cacher l'identité de ses clients (adresse IP, port, version, etc.) aux serveurs auxquels ces clients envoient des requêtes, de gérer un cache pour plusieurs clients, etc.

Gateway — intermédiaire coté serveur, permet notamment de cacher la structure réelle du serveur à ses clients et de traduire entre des protocoles différents.

Une particularité très importante de HTTP est le fait que la connexion entre un client et un serveur (connexion qui utilise TCP pour le transport fiable) se limite à une requête et sa réponse ; pour la requête suivante adressée par le même client au même serveur, une nouvelle connexion TCP doit être établie (certains serveurs permettent

de ne pas terminer la connexion TCP immédiatement, car la probabilité est grande pour que le même client envoie plusieurs requêtes successives au même serveur). Le serveur HTTP ne garde *a priori* aucune information concernant l'état des requêtes qui ont été servies (uniquement une trace de chaque requête, sous la forme d'un enregistrement dans un *logfile*). Ce mode de fonctionnement est adapté au spécifique de la consultation de pages d'hypertexte, simplifie le protocole et réduit la charge de la machine serveur ; en revanche, l'efficacité est faible en cas de consultation de documents gérés par le même serveur et l'utilisation d'une interface Web pour accéder à certaines applications distantes (bases de données par exemple) est plus difficile.

Structure d'une requête HTTP :

```
GET http://www.ics.uci.edu/index.html HTTP/1.0
Date: Tue, 15 Nov 1995 08:12:31 GMT
MIME-Version: 1.0      version déf. MIME (Multi-purpose Internet Mail Extensions) employées
From: crucianu@limsi.fr
If-Modified-Since: Sat, 29 Oct 1995 19:43:31 GMT
User-Agent: CERN-LineMode2.15 libwww/2.17b3
[corps éventuel]
```

Structure d'une réponse HTTP :

```
HTTP 200 OK                identification du protocole, code de retour (numérique, symbolique)
Date: Tue, 15 Nov 1995 08:12:31 GMT
MIME-Version: 1.0
Location: http://www.ics.uci.edu/index.html
Server: NCSA/1.3           type du serveur
Allow: GET, HEAD           méthodes permises pour la ressource
Content-Encoding: x-gzip   modification par rapport à Content-Type
Content-Length: 3456       longueur du corps de la réponse (présent aussi avec HEAD !)
Content-Type: text/html    type MIME du contenu
Expires: Thu, 01 Dec 1995 16:00:00 GMT
Last-Modified: Tue, 8 Nov 1995 10:00:22 GMT
                           une ligne vide sépare l'en-tête du corps
<html>…</html>             la réponse à une requête HEAD ne contient que l'en-tête
```

Codes de retour :

1xx : informationnel, non défini dans HTTP/1.0.

2xx : succès (200 — réussite, 201 — créée, 202 — acceptée, 204 — pas de contenu).

3xx : redirection (300 — choix multiples, 301 — déplacée de façon permanente, 302 — déplacée de façon temporaire, 304 — non modifiée).

4xx : erreur client (401 — demande d'autorisation, 403 — accès interdit, 404 — non trouvée).

5xx : erreur serveur (500 — erreur interne serveur, 501 — non implémentée, 503 — service indisponible).

Dans certains cas, le serveur HTTP lance en exécution un programme local (sur la même machine que le serveur) et lui transmet les paramètres présents dans la requête reçue ; le résultat — en format HTML — de l'exécution du programme externe constitue le corps de la réponse que le serveur renvoie au client. La consultation devient donc interactive grâce à l'utilisation de formulaires et de programmes spécifiques de traitement sur la machine serveur.

Méthodes utilisées dans les requêtes :

GET — permet de récupérer l'information identifiée par l'URL. Si l'URL fait référence à un exécutable, le serveur HTTP envoie à cet exécutable les paramètres qui suivent, récupère le résultat de l'exécution (la sortie standard de l'exécutable) et le renvoie au client avec un en-tête HTTP correspondant. Les paramètres peuvent non seulement être présents dans l'URL, mais aussi être introduits par l'utilisateur à l'aide d'un formulaire ; dans ce cas, la chaîne des paramètres est transférée par le serveur HTTP dans la variable d'environnement QUERY_STRING. Exemples d'utilisation :

`http://www.ics.uci.edu/index.html` → récupère le document `index.html`.

`http://www.cern.ch/cgi-bin/seq_anim?file=ball.dat&name=Red+ball`
→ appel de l'exécutable `seq_anim` et transfert de paramètres (après le "?", séparés par des "&" ; les espaces sont remplacés par des "+").

POST — permet de transférer des données que le serveur HTTP doit soumettre à la ressource référencée par l'URL. Cette méthode est censée permettre d'annoter des ressources existantes (si le serveur le permet), d'envoyer un message à un *newsgroup*, liste de distribution, etc. ou de transférer les paramètres destinés à un exécutable (les paramètres ne sont pas présents dans l'URL, contrairement à la méthode GET, et donc on peut transférer des informations confidentielles sans qu'elles apparaissent à l'écran). Les paramètres sont inclus dans une chaîne de caractère dont le format est le même que pour la méthode GET : `nom_par1=val1&nom_var2=val21&nom_var2=val22&…`

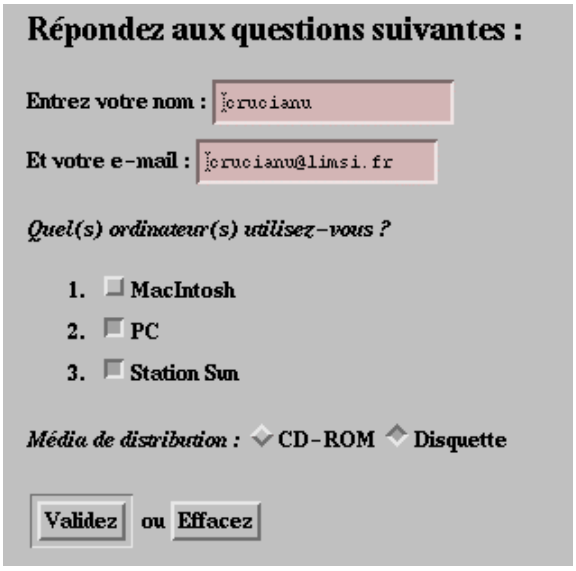
La chaîne est transférée par le serveur HTTP sur l'entrée standard de l'exécutable appelé. La longueur de la chaîne est donnée par la valeur de la variable d'environnement CONTENT_LENGTH, positionnée par le serveur.

HEAD — identique à la méthode GET à l'exception du fait que la réponse ne contient que l'en-tête ; permet d'obtenir des informations concernant la ressource référencée par l'URL sans transférer la réponse complète. Cette méthode est employée surtout pour tester les liens.

Exemple de traitement d'un formulaire :

Formulaire rempli par l'utilisateur (description HTML et image du formulaire rempli sur le *browser*) :

```
<html><head><title>Formulaire</title></head>
<body><h1>R&eacute;pondez aux questions suivantes :</h1>
<b><form method="post" action="/cgi-bin/form1">
Entrez votre nom : <input name="nom"><br>
Et votre e-mail : <input name="email"><p>
<i>Quel(s) ordinateur(s) utilisez-vous ?</i>
<ol><li><input type="checkbox" name="mc" value="MacIntosh"> MacIntosh
<li><input type="checkbox" name="mc" value="PC"> PC
<li><input type="checkbox" name="mc" value="Station Sun"> Station Sun
</ol><i>M&eacute;dia de distribution</i>
<input type="radio" name="media" value="CD-ROM"> CD-ROM
<input type="radio" name="media" value="Disquette"> Disquette <p>
<input type="submit" value="Validez"> ou
<input type="reset" value="Effacez">
</form></b></body></html>
```



Chaîne de caractères envoyée par le serveur HTTP (sur `stdin`) au programme `form1` :

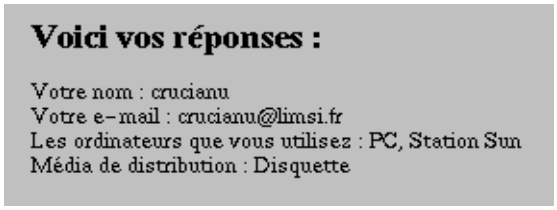
```
nom=crucianu&email=crucianu@limsi.fr&mc=PC&mc=Station+Sun&media=Disquette
```

Code en C du programme `form1(.c)` :

```
#include  <stdio.h>
#include  <stdlib.h>
#define   MAX_ENTRIES   6

typedef struct {
     char *name;
     char *value;
} entry;

/* fonctions de la librairie util.c (CERN) */
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
void unescape_url(char *url);
void plustospace(char *str);

main () {
     entry    entries[MAX_ENTRIES];
     int      cl, i;
          /* récupération de la longueur de la chaîne */
     cl = atoi(getenv("CONTENT_LENGTH"));
          /* analyse des paramètres */
     for(i=0; cl && !feof(stdin); i++) {
          entries[i].value = fmakeword(stdin, '&', &cl);
          plustospace(entries[i].value);
          unescape(entries[i].value);
          entries[i].name = makeword(entries[i].value, '=');
     }
          /* écriture de la réponse (en HTML) sur stdout */
     printf("Content-Type: text/html\n");
     printf("\n<html><head><title>Formulaire</title></head>");
     printf("<body><h1>Voici vos r&eacute;ponses :</h1>");
     i = 0;
     printf("Votre nom : %s<br>\n ", entries[i++].value);
     printf("Votre e-mail : %s<br>\n", entries[i++].value);
     printf("Les ordinateurs que vous utilisez :");
     while(!strcmp(entries[i].name, "micro"))
          printf(" %s,", entries[i++].value);
     printf("<br>\nM&eacute;dia de distribution : ");
     printf("%s<br>\n</body></html>", entries[i].value);
}
```

Réponse envoyée à l'utilisateur (document HTML et son image sur le *browser*) :

```
<html><head><title>Formulaire</title></head>
<body><h1>Voici vos r&eacute;ponses :</h1>
Votre nom : crucianu<br>
Votre e-mail : crucianu@limsi.fr<br>
Les ordinateurs que vous utilisez : PC, Station Sun<br>
M&eacute;dia de distribution : Disquette<br>
</body></html>
```



Un degré plus élevé d'intéractivité à moindre coût peut être obtenu grâce au transfert via le réseau de programmes (de taille relativement réduite, *applets*) qui s'exécutent sur la machine du client. Le langage Java (développé par Sun, enseigné en troisième année) permet d'écrire de tels programmes ; les programmes sont interprétés par une machine virtuelle qui est incorporée dans certains *browsers* ; l'utilisation d'une machine virtuelle permet d'assurer une bonne portabilité des *applets* et aussi d'imposer des restrictions sur les accès de l'*applet* aux ressources de la machine locale.

Bibliographie

Black, U. (1994) *TCP/IP and related protocols*, McGraw-Hill Series on Computer Communications, New York, 1994 (en bibliothèque, 628/5882).

En anglais. Relativement complète et souvent détaillée. Parle peu de la programmation mais présente les relations entre TCP/IP et d'autres protocoles. Les explications ne sont pas toujours claires.

Bouyer, G. (1997) *Les réseaux synchrones étendus PDH et SDH*, Hermès, Paris (en bibliothèque, 628/7580-0), 460 p.

Présentation en français des techniques utilisées comme support pour les réseaux étendus, la hiérarchie numérique plésiochrone (Plesiochronous Digital Hierarchy) et synchrone (Synchronous Digital Hierarchy).

Feit, S. (1996) *TCP/IP: Architecture, Protocols and Implementation*, McGraw-Hill, Inc., New York (en bibliothèque, 628).

En anglais. Edition revue et augmentée d'un ouvrage déjà très complet et détaillé. Les additions concernent notamment IPv6. Présente aussi la programmation (utilisation des sockets).

Händel, R., Huber, M. N., Schröder, S. (1995) *Comprendre ATM*, Addison-Wesley, Paris, 1995.

Très bonne présentation, avec des références aux normes existantes (la normalisation étant encore incomplète).

Huitema, C. (1995) *Le routage dans l'Internet*, Eyrolles, Paris, 1995 (en bibliothèque, 628/5584).

Présentation détaillée des algorithmes de routage (construction et mise à jour des tables de routage) utilisés en conjonction avec IP comme RIP, OSPF, EGP, BGP, CIDR.. Bibliographie très fournie.

Maiman, M. (1994) *Télécoms et réseaux*, Masson, Paris, 1994.

Tour d'horizon rapide, toutefois avec quelques détails techniques difficiles à trouver ailleurs. Très inégale. Un point fort : des exemples de traces (niveaux OSI 2 à 5).

Millet, M. (1987) *Transmission et réseaux locaux : architecture IEEE 802*, Masson, Paris, 1987 (en bibliothèque, 628/3688).

Présentation des aspects physiques des réseaux locaux, des techniques d'accès les plus employées et des propositions IEEE 802. Assez peu de détails et d'explications sur IEEE 802...

Montagnier, J.-L. (1998) *Pratique des réseaux d'entreprise*, Eyrolles (en bibliothèque, 628/7355-0), 525 p.

Présentation des réseaux locaux ou étendus utilisés, en laissant de côté le modèle OSI. Détails sur Ethernet et Token-Ring, beaucoup moins sur FDDI ou ATM. Les protocoles de niveau réseau sont mentionnés, certains très rapidement (NetBIOS, AppleTalk, DECNet)... Bonne introduction aux réseaux télécom et au réseau téléphonique (classique et RNIS). Les études de cas détaillées à la fin de l'ouvrage sont ***très utiles*** *!*

Pujolle, G., Seret, D., Dromard, D., Horlait, E. (1989) *Réseaux et Télématique*, Tome 2, Eyrolles, Paris, 1989 (en bibliothèque, 628/5095-2).

Tour d'horizon relativement vaste des réseaux en général et des services disponibles (en 1986...). Survol rapide des réseaux locaux (presque exclusivement les aspects technologiques). Certains chapitres sont dépassés. Assez inégale.

Rolin, P. (1995) *Réseaux haut débit*, Hermès, Paris, 1995 (en bibliothèque, 628/5891).

Présentation à jour de différents protocoles et types de réseaux : Ethernet 100 Mbps, FDDI, Relais de trame, ATM (env. 200 pages), IPv6. Présentation d'architectures d'interconnexion : relais de trame — ATM, LAN — relais de trame, X25 — relais de trame, LAN — ATM, émulation LAN.

Tanenbaum, A. (1990) *Réseaux : architectures, protocoles, applications*, InterEditions, Paris, 1990 (en bibliothèque, 628/4365).

Traduction de l'anglais, date un peu... Bonne présentation du modèle OSI. L'épaisseur n'est pas due aux détails techniques qui font parfois cruellement défaut, mais au volume très important d'explications de bonne qualité — pour comprendre le pourquoi et le comment le livre est imbattable.