DI GALLO Frédéric

COURS DE RESEAUX ET SYSTEMES

Cycle Probatoire

CNAM BORDEAUX 1999-2000

Cnam
CONSERVATOIRE NATIONAL DES ARTS ET METIERS

RAPPEL SUR LA THEORIE DU SIGNAL

CIRCUITS ET LIAISON DE DONNEES

SUPPORTS & MODE DE TRANSMISSION

LES SUPPORTS PHYSIQUES

PROTECTION CONTRE LES ERREURS DE TRANSMISSION

PROTOCOLES DE COMMUNICATION

LE NIVEAU LIAISON DU MODELE OSI

LE NIVEAU RESEAU DU MODELE OSI

INTERCONNEXION DE RESEAUX

TCP/IP

LE NIVEAU TRANSPORT DU MODELE OSI

LES COUCHES HAUTES DU MODELE OSI

PROCESSUS & APPLICATIONS REPARTIES

SNA ET OSI/DSA

ADMINISTRATION DE RESEAUX

LA SECURITE DANS LES RESEAUX

EXERCICES SUR LA SECURITE

SECURISATION D'UN RESEAU LOCAL

PROTOCOLES TRANSPORT SECURISES

EVALUATION DE RESEAUX ET DE PROTOCOLES DE COMMUNICATION

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

I. INTRODUCTION

Les réseaux prennent de plus en plus d'importance car ils permettent de minimiser les coûts de transport des informations. Ils permettent aussi de réduire les durées de circulation de l'information (forums, courriers électroniques, …).

Les techniques réseaux évoluent.

- Liaisons point à point.
- Adaptation du signal aux supports (modulateurs : adaptation analogique ⇔ numérique).
- Protection du signal contre des perturbations (électriques, mécaniques, électromagnétiques, …) : protection électrique (blindage) et protection logique (code détecteur ou correcteur).

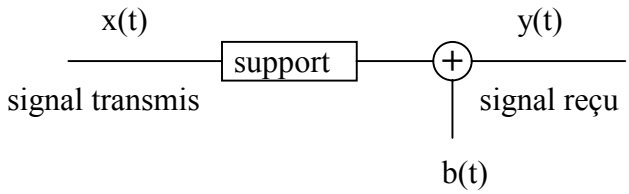
- Logiciels pour décrire des applications.

THEORIE DU SIGNAL

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

RAPPEL SUR LA THEORIE DU SIGNAL

Les défauts sur un support analogiques peuvent être le bruit ou l'affaiblissement ;

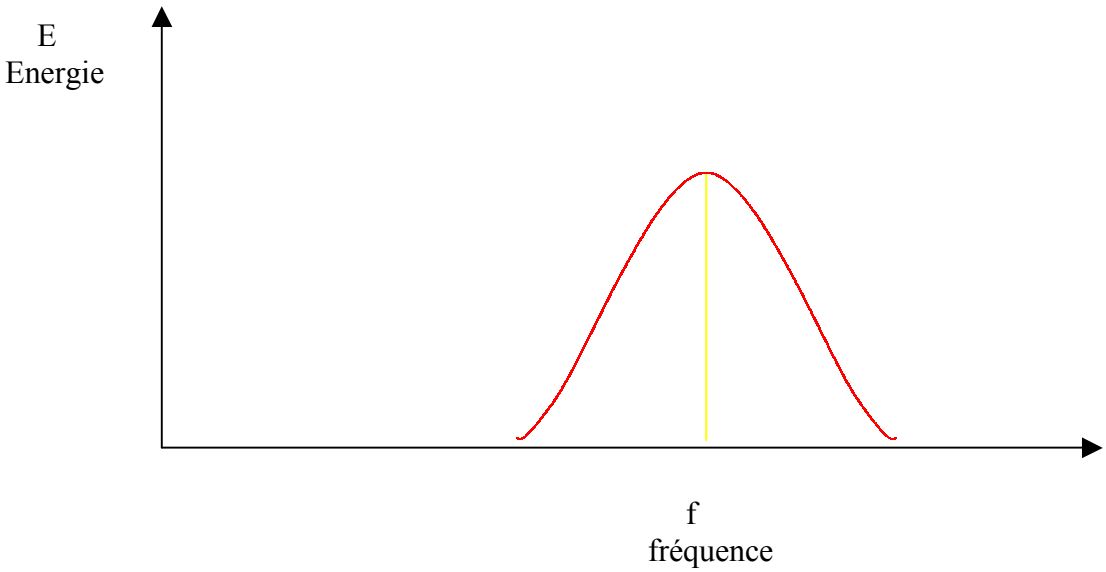


filtre linaire : $x(t) \rightarrow y(t)$

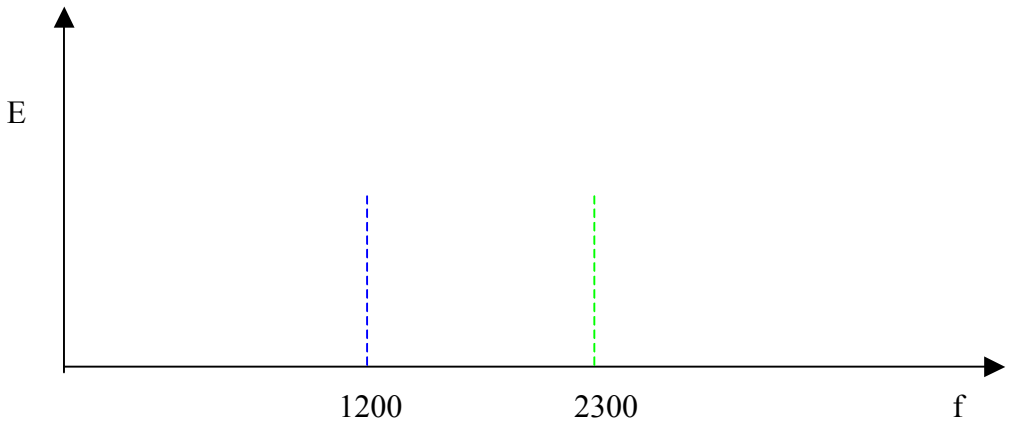
$x(t-t_0) \rightarrow y(t-t_0)$

$ax_1(t) + bx_2(t) \rightarrow ay_1(t) + by_2(t)$

I. NOTION DE SPECTRE :

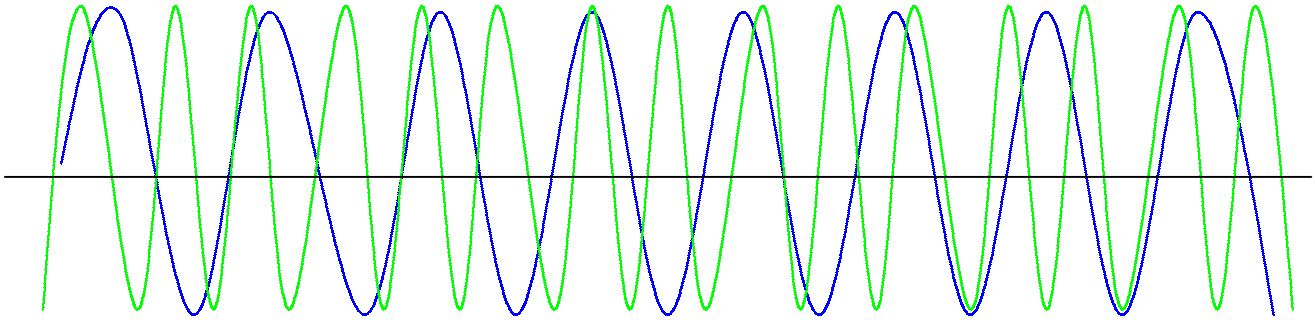


en jaune : sinusoïde parfaite de fréquence f
en rouge : sinusoïde réelle de fréquence f

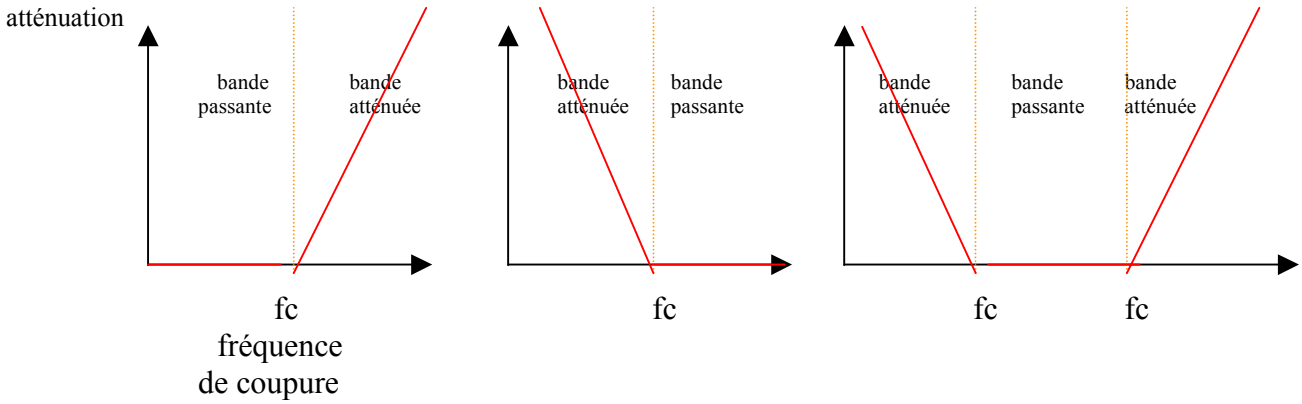


spectre de raies

(modem : 1200 Hz en émission
2300 Hz en réception)



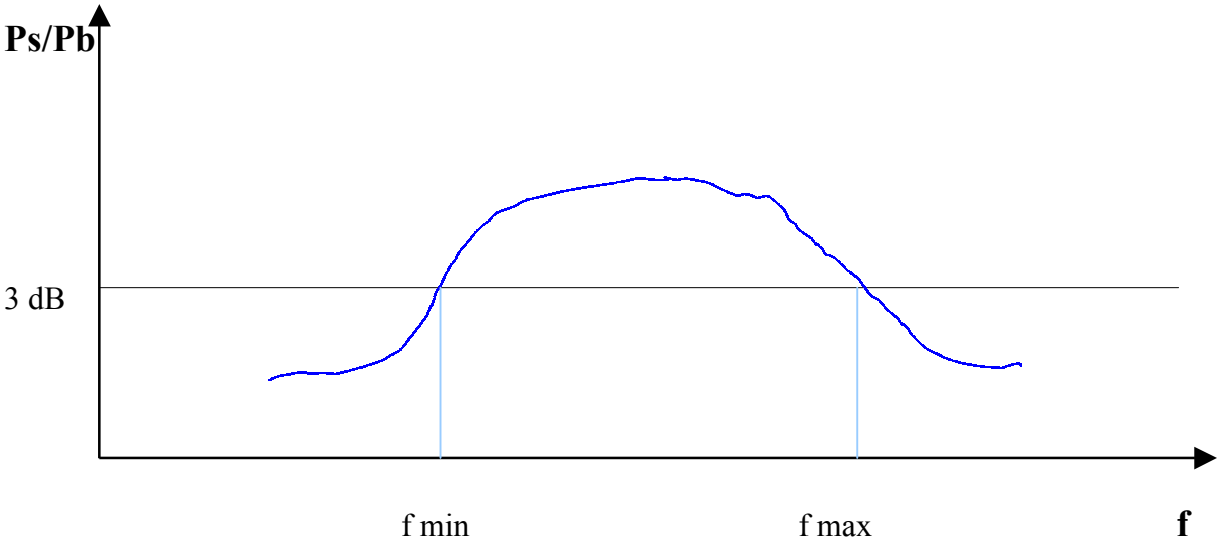
Filtres : passe bas passe haut passe bande (passe haut et passe bas mis en série)



Un filtre c'est une bande passante à n décibels
L'atténuation se mesure en dB(décibels) :

$$N = 10 \log_{10} \left(\frac{Ps}{Pb} \right)$$

Ps = puissance du signal
Pb = puissance du bruit



pour N = 3 dB, il faut que Ps/Pb ≥ 2

II. CANAL DE TRANSMISSION

- L'information est une grandeur mesurable donc calculable mathématiquement.
- Les performances d'un canal sont définies en termes probabilistes.
- La source transmet des séquences de symboles d'un alphabet donné (codage de l'information).
- Le collecteur de données possède également un alphabet non obligatoirement identique.

Le débit linéaire d'un support de transmission (formule de Shannon) :

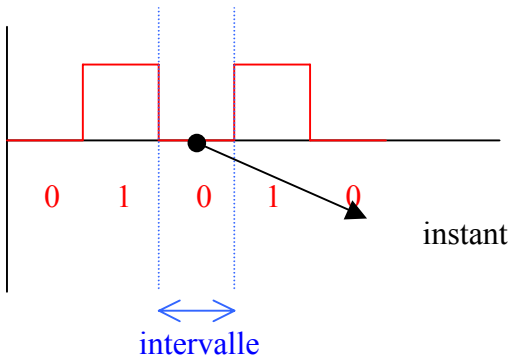
$$D = W \log_2 (1 + \partial)$$

D = débit binaire en bit/seconde
W = bande passant (fréquence en Hz)

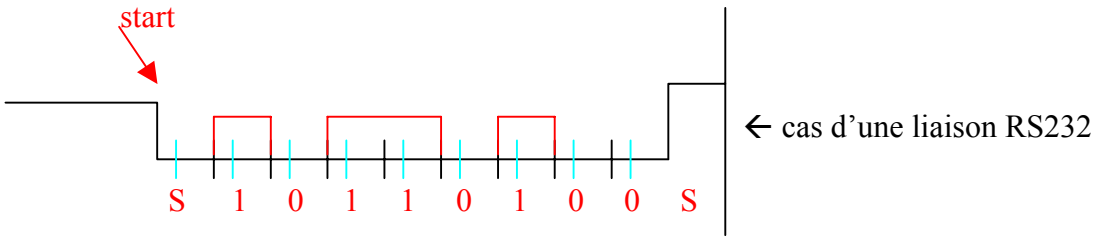
S = énergie du signal
B = énergie du bruit
∂ = rapport d'énergie

Le débit binaire dépend de la façon dont on code le signal.

- Intervalle significatif : Temps pendant lequel les caractéristiques du signal sont stables. Intervalle compris entre deux temps significatifs.



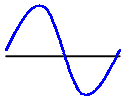
- L'instant significatif : instant choisi pour prélever l'information



Instant significatif

Même s'il y a un décalage de l'instant significatif, ce n'est pas grave car au bout de 10 caractères, on remet un bit de start pour bien repartir.

- Valence : nombre des états significatifs distincts dans une modulation (2π)



- Rapidité de modulation : s'exprime en bauds, c'est l'inverse de l'intervalle significatif

T = intervalle significatif

R= rapidité de modulation (baud)

- Débit binaire : quantité d'information par seconde. Exprimé en bit par seconde.

D = débit binaire (en bit/seconde)
R = rapidité de modulation (en baud)
V = valence ($\in$ N)
16,32,64..

$D = R \log_2 V$

EXERCICE 1 :

Trouver les rapports d'énergie pour 10 dB, 3 dB, 40 dB et 37 dB.

$N = 10 \log_{10} (R)$

$\log_{10} (100) = 2$ car $10^2 = 100$

10 dB → R= 10

3 dB → R= 1,995… $(10^{0,3})$

40 dB → R= 10^4

37 dB → R= 5011,87…. $(10^{3,7})$

EXERCICE 2 :

A combien de dB correspondent les rapports de puissance suivants : 2000, 500, 100 000

2000 : 1000 → 30
2000 → 30 +3 = 33

500 : 500 = 1000 : 2 → 30-3 = 27

100 000 : 100 000 = 10^5 → 50

EXERCICE 3 :

a) Quel est le débit binaire d'une voie qui émet un signal à chaque période T ? T=10 ms

$D = R \log_2 V$
$D =$ **Erreur !** $\log_2 V$ → par défaut il faut savoir que l'on a une valence de 2

$D =$ **Erreur !** $\log_2 (2) =$ **Erreur !Erreur !Erreur !** = **Erreur !Erreur !** $\frac{1}{10}$ =100 b/s

$n = \log_2 2^n$

b) Δ = intervalle significatif. Quelle est la rapidité de la modulation disponible ?
Erreur ! = rapidité de modulation T = intervalle significatif

$$\frac{1}{\Delta} = \frac{1}{100} = 10 \text{ bauds}$$

c) Le signal a une valence V. quel est le débit binaire disponible ? Exprimer cette grandeur en fonction de Δ et de V.

$D = R \log_2 V$ $R =$ **Erreur !** $= \frac{1}{\Delta}$

$$D = \frac{1}{\Delta} \log_2 V \rightarrow D = \frac{\log_2 V}{\Delta} = \frac{4}{0{,}01} = 4000 \text{ bits/secondes}$$

EXERCICE 4 :

1) Quelle est la modulation nécessaire (en baud), D = 2400 b/s, signaux binaires (V=2)

$$R = \textbf{Erreur !} = \frac{\log_2 V}{D} = \frac{\log_2 2}{2400} = \frac{1}{2400} = 2400 \text{ bauds}$$

2) Bande passante 1000 Hz, débit binaire = 2400 b/s, quel est le rapport **Erreur !** ? en rapport de puissance puis en dB

$D = W \log_2 (1 +$ **Erreur !**$)$

$2400 = 1000 \log_2 (1 +$ **Erreur !**$) \rightarrow \log_2 (1 +$ **Erreur !**$) = 2{,}4 \rightarrow 1 +$ **Erreur !** $= 2^{2,4} \rightarrow$ **Erreur !**$= 2^{2,4} - 1 = 4{,}278..$

3) 2400 b/s, V=4, Quelle est la modulation ?

$$R = \textbf{Erreur !} = \frac{\log_2 V}{D} = \frac{\log_2 4}{2400} = \frac{2}{2400} = 1200 \text{ bauds}$$

EXERCICE 5 :

- support qui varie de 60 kHz → 108 kHz (bande passante)
- 37 dB

a) débit binaire théorique ?
b) même question avec 40 dB ?

a)

$D = W \log_2 (1 + \textbf{Erreur !})$

$D = 48.10^3 \log_2 (1+5000)$ (48.10^3 car kHz)

$D = 589\ 810$ b/s

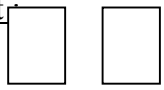
b) $D = W \log_2 (1 + \textbf{Erreur !})$

$D = 637\ 810$ b/s

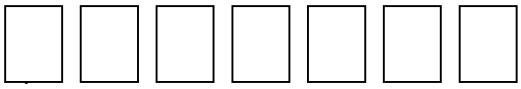
EXERCICE 6 :

La base 2 est elle la meilleur base pour stocker l'information ? R : non ! pourquoi ?
En fait quelle est la base minimale que l'on doit adopter pour avoir le minimum d'état stable ?

Au basket :

 → 20 plaques pour score de 00 à 99 en décimal

Panneaux pour afficher le score

en binaire  → 14 plaques pour score

Réponse :

np = b * d np = nombre de plaques
b = base
d = nb de chiffres nécessaires

d ? $b^d \geq N$
$\log_b b^d = d \geq \log_b N = \textbf{Erreur !}$

$d \geq \textbf{Erreur !}$

np(b) = b **Erreur !** = k $\frac{b}{\ln b}$ $\Rightarrow$ np'(b) k $\frac{\ln(b)-1}{\ln(b)^2}$

or ln(b) – 1 → 0 d'où ln(b) = 1
b = e

D = W $\log_2$ (1 + **Erreur !**)

D = R $\log_2$ V

Si on rapproche ces 2 formules, on s'aperçoit que la valence est liée au rapport signal sur bruit

III. Modulation d'un signal :

Signal binaire

Signal analogique

signal émis signal reçu

$\Rightarrow$ non adapté pour bande passante étroite

La modulation, c'est transformer un signal binaire en signal analogique. Elle utilise une onde sinusoïdale (porteuse) dont la fréquence est supérieure à l'onde modulante.

$P(t) = A_p \cos(\varpi_p t + \varphi_p)$

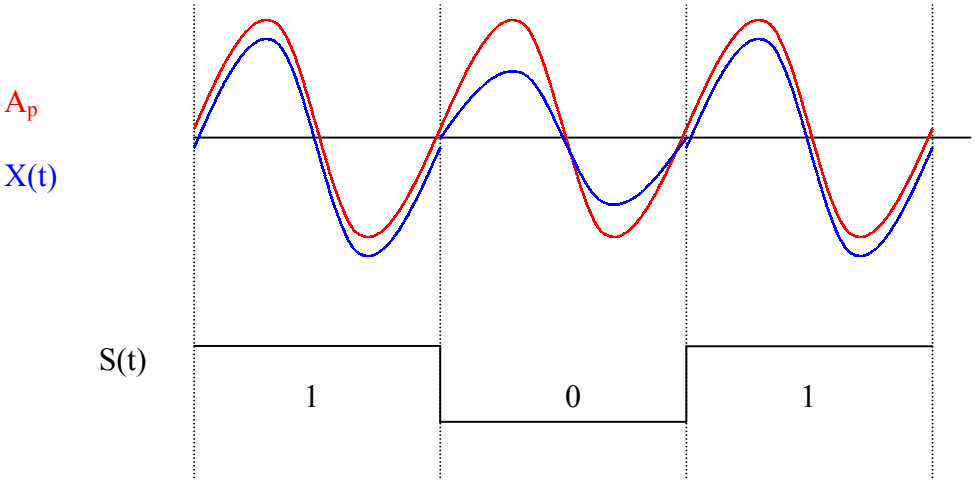
A_p = amplitude
ϖ_p = pulsation
φ_p = déphasage

3.1) 1ère technique de modulation : modulation d'amplitude

$x(t) = A_p (K + m\, s(t)) \cos(\varpi_p t + \varphi_p)$

m = taux de modulation
s(t) = signal variant en fonction du temps

m s(t) > -K

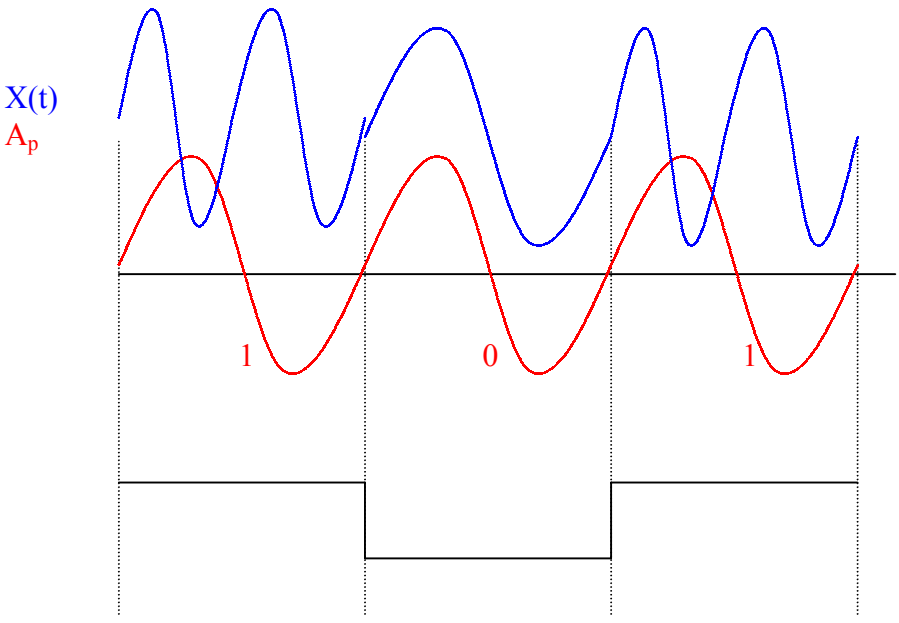


Utilisé pour : BLU : bande latérale unique (K=0)
MAQ : modulation de 2 porteuses en quadrature
Inconvénients : parasites

3.2) 2ème technique de modulation : modulation de fréquence

signal modulé : $x(t) = A_p \cos (\varpi_p t + \varphi_p + 2 \pi W \int s(u)\, du)$

$2 \pi W$ = excursion de fréquence (décalage de fréquence rapport à fréquence du signal émis)



Ici pas de perte dues au signal.

3.3) 3ème technique de modulation : modulation de phase

La plus employée actuellement pour la transmission de signal. Modulations à 2, 4 ou 8 états de phase.

Modulation de phase à 2 états sur un intervalle de modulation T.

$x(t) = A_p \cos(\varpi_p t + \Theta + \varphi_i)$ avec $\varphi_{i} = \pm \pi$ (teta)

Modulations combinées d'amplitude et de phase

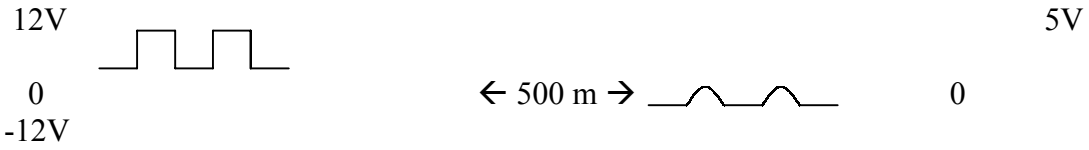
Objectif : meilleur densité et meilleur qualité du signal

$x(t) = A_i \cos(\varpi_p t + \varphi_i)$ sur $[iT,(i+1)T]$ (MAQ)

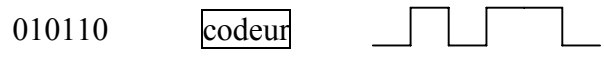
Il existe aussi un codage numérique → bande de base

Transmission en bande de base

On transmet les signaux numériques sur le support et sur des distances limitées (< 30Km).



L'émetteur à un codeur bande de base :

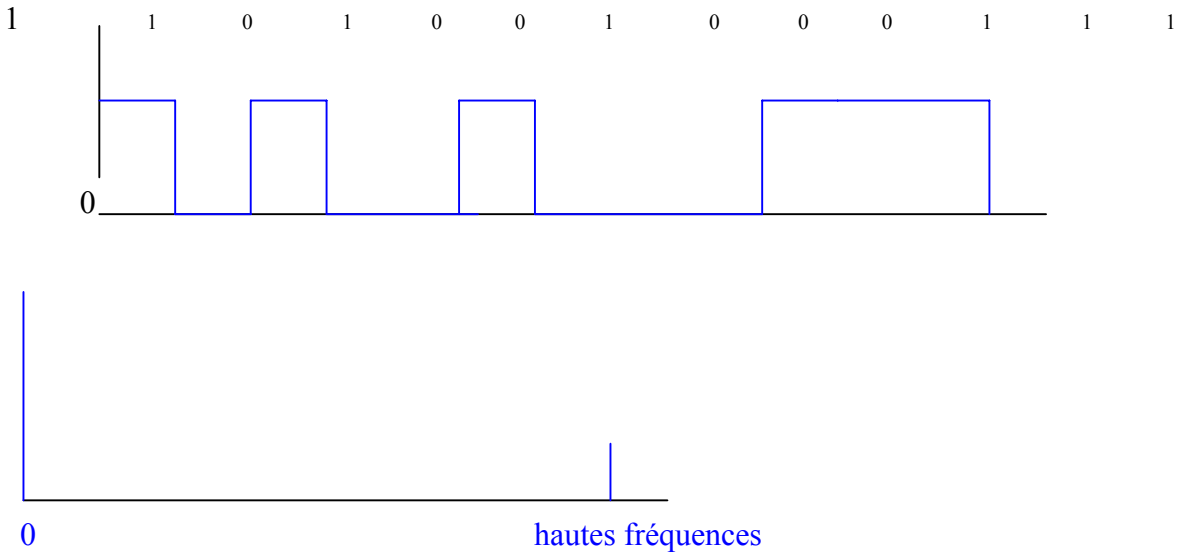


Ce codeur transforme une suite de bits $\{a_i\}$ en une suite de symboles $\{d_k\}$

$d_k \in \{\alpha_1, \ldots, \alpha_q\}$

Les d_k ont le même intervalle significatif $\Delta \geq T$ (intervalle élémentaire).

Ne peut être employé que si le support n'introduit pas de rapport de fréquence.
Nécessité du codage en bande de base



IV. SPECTRE DE FREQUENCE

- Le spectre est illimité concentré sur f =0
- Valeur moyenne = **Erreur !** (on préfère 0 car + économique)
- Longues suites de bits identiques (problème d'échantillonnage)

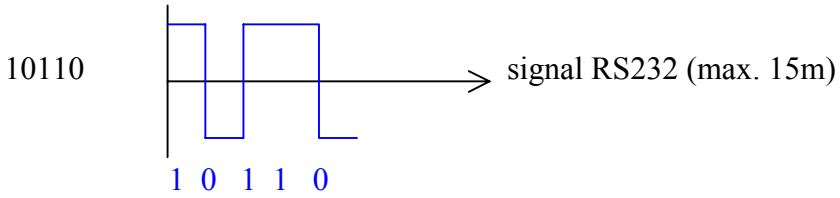
Pour éviter ces problèmes d'échantillonnage, il existe 2 méthodes de transmission d'horloge :

1) On émet le signal d'horloge en le superposant aux données.
2) On déduit l'horloge à partir des transitions du signal.

4.1) Codages à 2 niveaux

NRZ (non-retour à zéro)

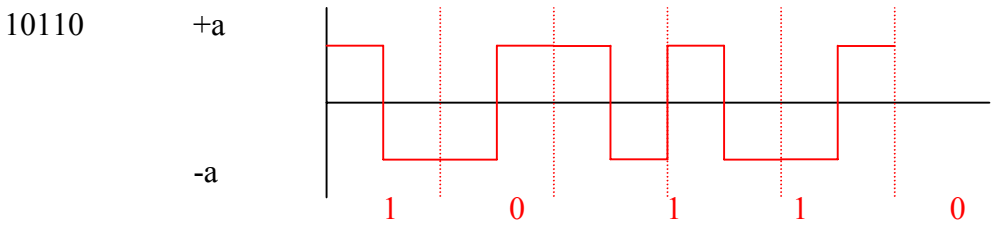
Si $a_i = 0$ alors le signal vaut –a
Si $a_i = 1$ alors le signal vaut +a



avantages : intéressant car il ne revient jamais à zéro et résistant aux bruits
inconvénients : spectre concentré autour de fréquences extrêmement basses et problème d'horloge

4.2) Codage biphase (Manchester)

Introduire des transitions au milieu de l'intervalle significatif.
Front montant → $a_i = 0$
Front descendant → $a_i = 1$



pas de problème d'horloge
problème si on inverse les deux câbles (les 0→1 et les 1→0)

4.3) Codage Manchester différentiel (Ethernet)

On code la valeur du bit par rapport à la valeur précédente.

Front montant au milieu de Δ si $|a_{i-1} - a_i| = 0$ (bit actuel et précédent identiques)
Front descendant au milieu de Δ si $|a_{i-1} - a_i| = 1$ (bit actuel et précédent différents)

4.4) Codage de Miller

Déduit du précédent en supprimant une transition sur deux.

Décodage :
- Si sur un intervalle Δ, le signal ne présente pas de transition, la donnée vaut 0 ($a_i = 0$).
- Si sur un intervalle Δ, le signal présente une transition, la donnée vaut 1 ($a_i = 1$).

EXERCICE :

On désire transmettre une suite de bits : 00101101, dessiner la suite de signaux transmis par un modem.

a) en modulation de phase quadrivalente (valence = 4).

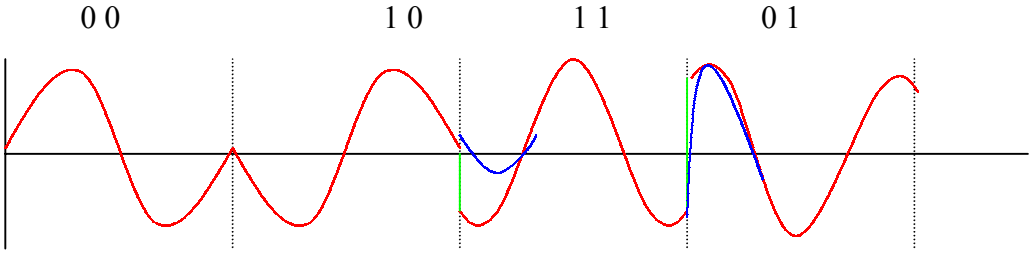
4 états par modulation

$00 \rightarrow \varphi = 0$

$01 \rightarrow \varphi = \frac{\pi}{2}$

$10 \rightarrow \varphi = \pi$

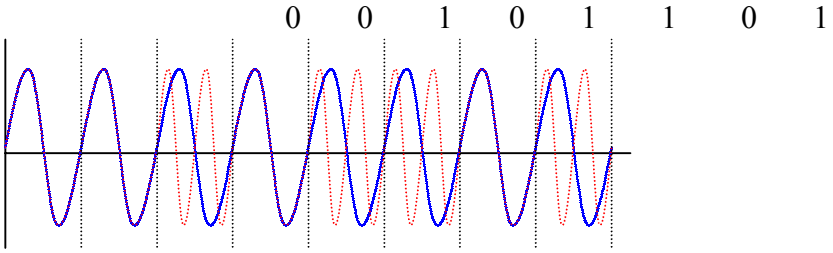
$11 \rightarrow \varphi = \frac{3\pi}{2}$



« Cassure » du signal

Signal Réel

b) En modulation de fréquence bivalente.



Signal

Porteuse

0 → idem porteuse
1 → fréquence doublée par rapport à la porteuse

EXERCICE :

Transmettre 00101101, liaison avec bande passante à 1200 bauds, bande de base, je veux transmettre 2400 bits/seconde, comment faire le codage des données ?

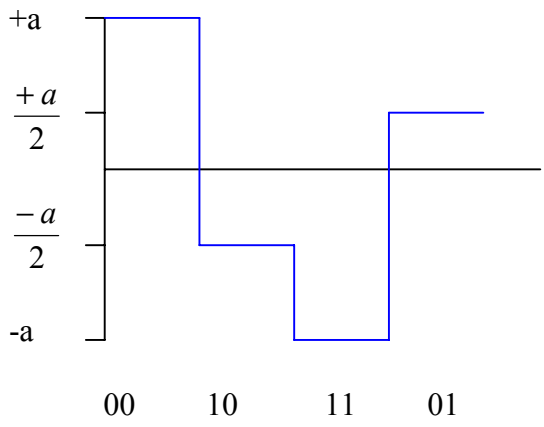
Solution : On joue sur l'amplitude.

00 → +a

01 → $\frac{+a}{2}$

10 → $\frac{-a}{2}$

11 → -a

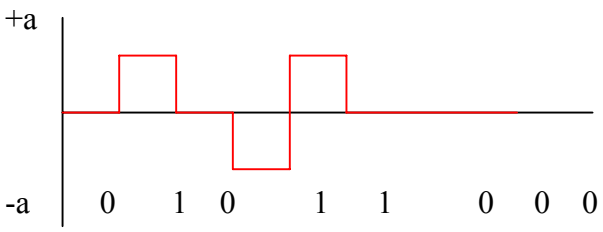


4.5) Codages à 3 niveaux

- Fréquence réduite.
- S'annule pour fréquence nulle.

4.6) Codage bipolaire simple (d'ordre 1)

$a_i = 0$ le signal vaut 0
$a_i = 1$ le signal vaut +a ou –a selon le signal précédent.



- Plus sensible au bruit que le codage à 2 niveaux.
- Les problèmes ne sont pas résolus pour des sites de zéros.
- Pour résoudre ce problème d'horloge→codage BHDn (Bipolaire à Haute Densité d'ordre n).

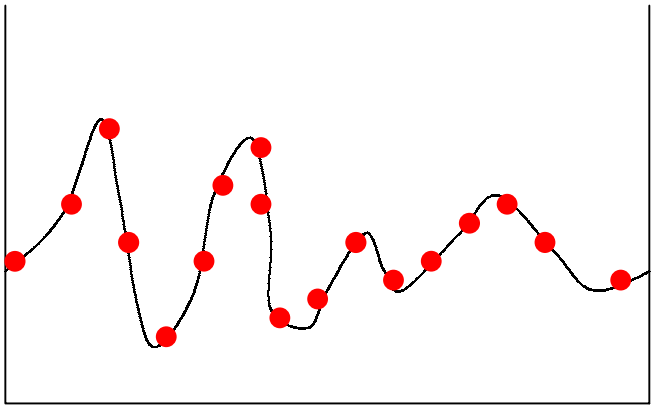
4.7) Critères de choix d'un codage

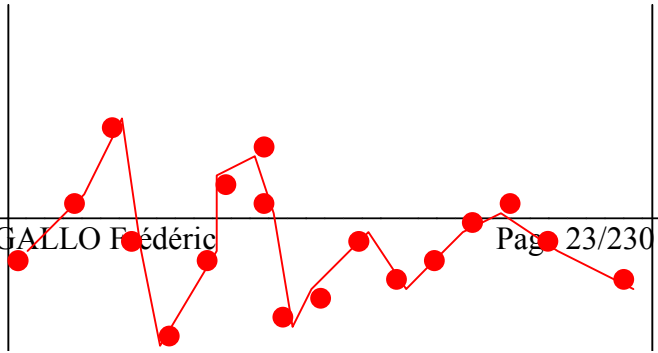
- Choisi en fonction des paramètres connus du support.
- Les supports de transmission (la plupart) coupent brutalement la fréquence quand elle passe au voisinage de zéro, le plus mal adapte est le NRZ. Le codage biphase nécessite de larges bandes.
- On code en fonction de la résistance au bruit. Les parasites sont liés au nombre de niveaux du signal. Les codes bipolaires de niveau 3 sont donc plus sensibles que les codages à 2 niveaux.
- On code aussi en fonction des problèmes d'horloge. Le décodage des données devient impossible en cas d'erreur (avec le BHD par exemple) quand l'horloge est transmise dans le signal (de manière contextuelle).

Ecart au cours : sur un micro, la probabilité d'erreur non détectée avec ECC (erreur control correcteur) est de 10^{-12} et elle passe à 10^{-6} sans ECC.

- Numérisation du signal :
  - Transmission analogique du signal analogique (TSF)(paroles, images).
  - Transmission analogique d'information numérique (modem).
  - Transmission numérique d'information analogique.
  - Transmission numérique d'information numérique.

Pour le codage analogique → numérique, il faut faire un échantillonnage.





Pas parfait ! ! !

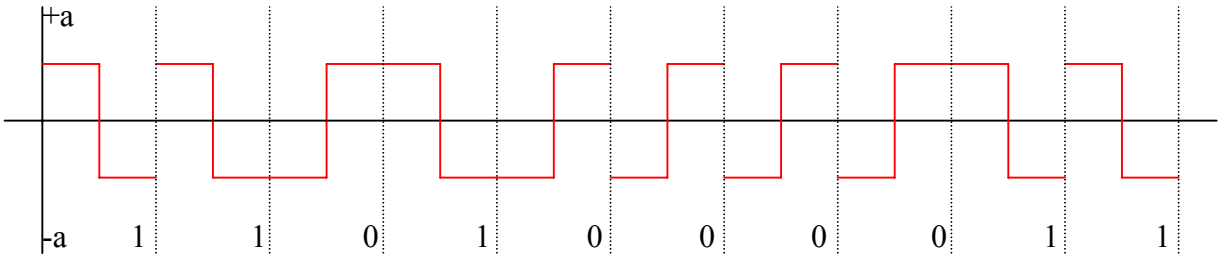
Si B est la fréquence la plus élevée contenue dans le spectre A(f) du signal a(t), la fréquence d'échantillonnage doit être ≥ 2B.

Un échantillonnage à 8KHz → 8000 mesures/seconde.

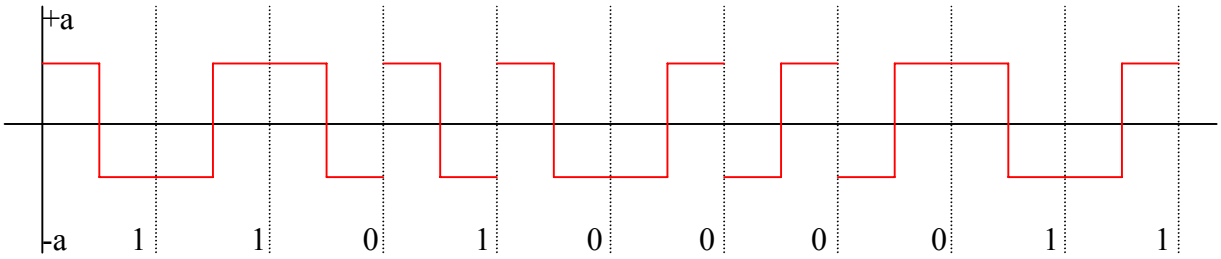
EXERCICE :

1) coder en Manchester 1101000011
2) coder en Manchester différentiel 1101000011

1)



2) hypothèse de départ : bit précédent 0



CIRCUIT & LIAISON DE DONNEES

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

CIRCUITS ET LIAISON DE DONNEES

I. DEFINITIONS

ETCD : Equipement Terminal de Circuits de Données

Equipement placé à chaque extrémité du circuit de transmission ayant pour rôle de transformer le signal en données compatibles pour le circuit (ex : modem …).

Circuit de données :

Ensemble constitué du support de transmission + ETCD.

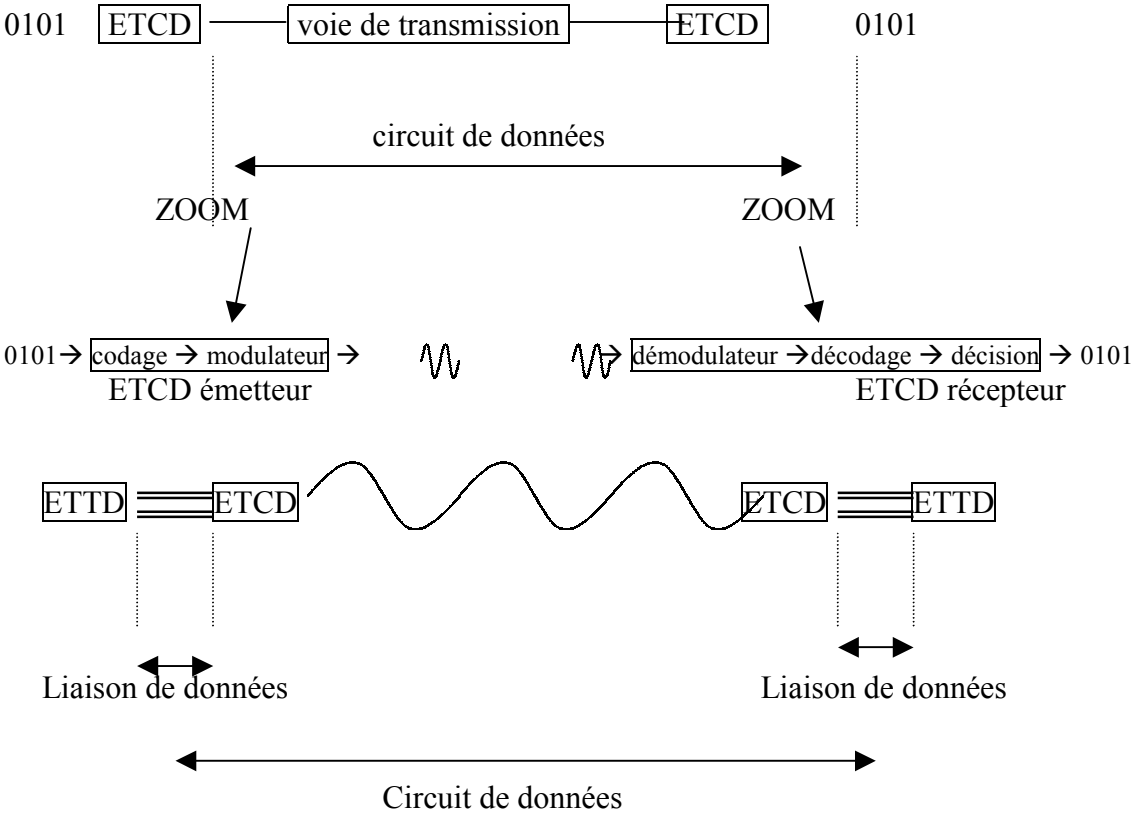
ETTD : Equipement Terminal de Transmission de Données

(ex : terminal, console, ordinateur en liaison RS232 …).

Liaison de données :

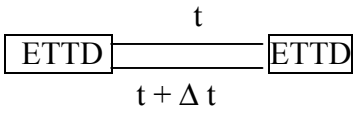
Dispositif matériel + logiciel pour acheminer des données avec un taux d'erreur minimum garanti.

Structure générale d'un outil de communication :

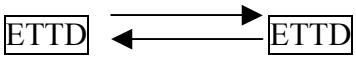


Nature des liaisons de données :

- Unidirectionnelles (simplex) : contrôle de processus ou acquisition de données (ex pour un banc de mesure). Ex : transmission par fibre optique, hertzienne, …
- Bidirectionnelles à l'alternat (half duplex) : un coup dans 1 sens, un coup dans l'autre.



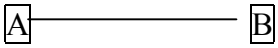
- Bidirectionnelles simultanées (full duplex) : ex : les modems actuels. Utilisation de bandes de fréquences différentes sur le canal de transmission.



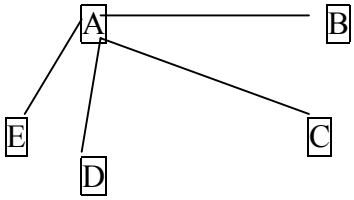
Liaison full duplex

Configurations des liaisons de données :

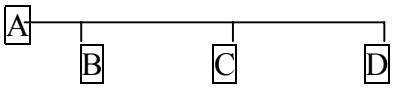
- Point à point : ne relie que deux ETTD.



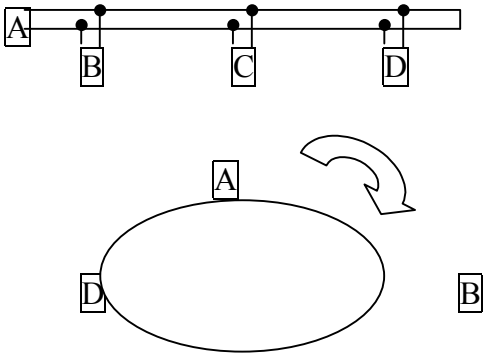
- Etoile : collection de point à point.



- Multipoints : plusieurs ETTD branchés sur le même support (gain de câbles).



- En boucle : on peut considérer que c'est une double liaison multipoints. Ex : token ring, FDDI (100mb/s).



C

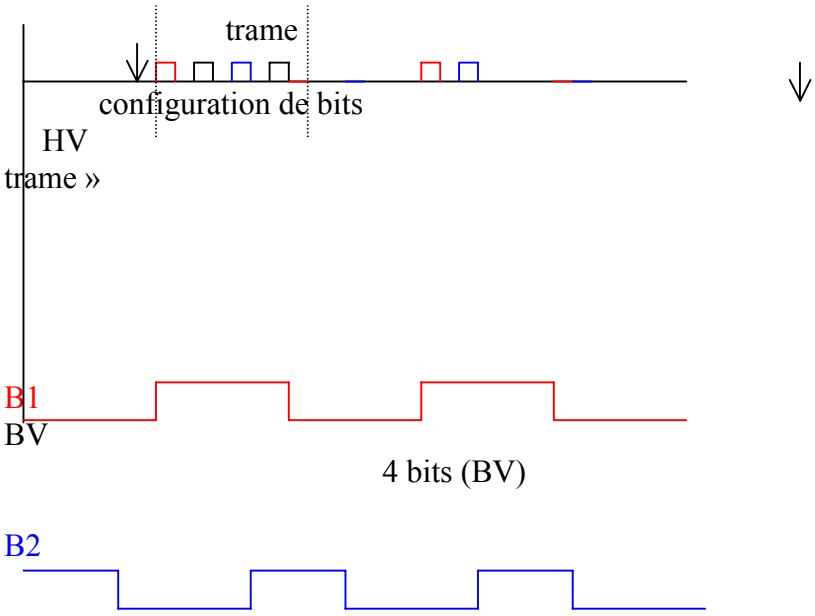
II. Les multiplexeurs (concentrateurs)

Concentrer les liaisons basses vitesses (BV) sur un seul circuit de données à plus haut débit (HV).

- Analogique : voies de communication (avant le modem) → bandes de fréquences isolées par des filtres.
- Temporel : données (après le modem).

- Multiplexage temporel par caractères

  Objectif : former une trame de bits de débit D sur une voie HV en fonction d'une voie BV.



Prévoir une synchronisation en début de trame.

Technique statique : les trames sont de longueurs fixes.
Technique dynamique : découper l'information en « paquets » ou « cellules » avec envoi à la demande.

- Multiplexage de voies MIC (téléphone)

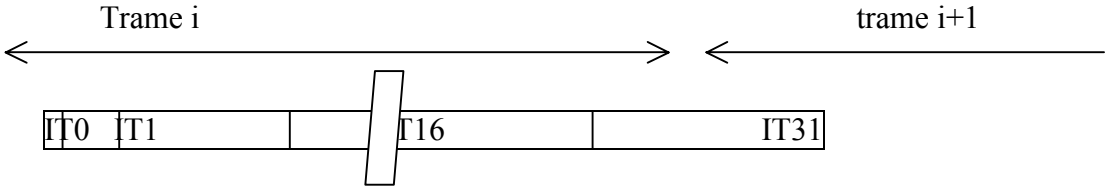
  On profite de la faible bande passante des lignes téléphoniques.
  3400 – 300 Hz = 3100 Hz (bande passante)
  → 4000 comme bande passante utile.

CEPT : Commission Européenne des Postes et Télécommunications
Ils ont normalisé un multiplexeur téléphonique à 30 voies.

Structure de trame à 32 IT de 8 bits (numérotés de 0 à 31)

IT 0 = verrouillage de trame + signaux d'alarme.

IT 16 = signalisation correspondant aux 30 voies téléphoniques.
Chez Alcatel
IT 1
Chez les autres



EXERCICES :

1) Les liaisons de données à l'alternat sont susceptibles de contentions. Comment cela se traduit-il au niveau de la transmission de données ?

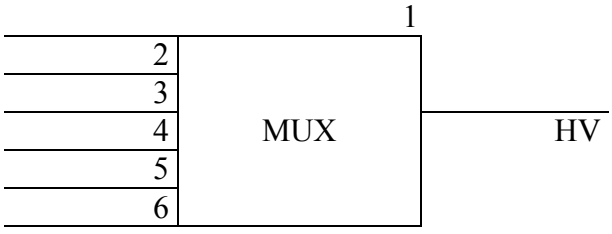
   ➔ 1er cas : les 2 attendent que l'autre émette ou que les 2 n'ont rien à dire.

   A émet juste pour dire qu'il n'a rien à dire à B et cela à intervalle régulier.

   ➔ 2ème cas : les 2 émettent en même temps.

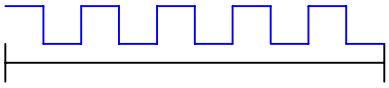
   Pour détecter cela, il faut s'écouter parler (comparer ce qu'il envoi avec ce qu'il écoute). En cas de collision, il arrête et réessaye un peu lus tard.

2) On veut multiplexer 6 voies binaires de débit différents sur une voie HV. Multiplexage temporel par caractère. Les lignes BV sont en mode asynchrone et envoient 1 bit de start + 8 bits de données + 1 bit de stop.

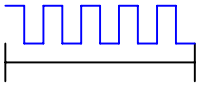


a) Indiquer les différentes solutions d'affectation des IT pour une transmission de signalisation hors bande. Quel est l'impact du débit dont on a besoin sur HV (quelle est la manière la plus efficace de faire) ?

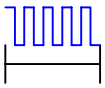
100 bits/s



200 bits/s



300 bits/s



1 →

	1	2	3	4	5	6	
	1/3	1/3	1/3	2/3	2/3	3/3	

Verrouillage Signalisation

V	1	2	3	4	5	6	S	V				4	5	6	S	V						6	S

2400 bits/s
inconvénients : trames vides

2 →

V	6	4	1	5	6	2	3	4	6	5	S

1200 bits/s pas + de 3 intervalles entre les 6 (1200/300=4)

inconvénients : trame longue pas + de 5 intervalles entre les 4 (1200/200=6)

3 →

V	6	1, 2 ou 3	4 2fois 5	S

			2fois	

V	6	1	4	S
V	6	2	5	S
V	6	3	4	S
V	6	1	5	S
V	6	2	4	S
V	6	3	5	S

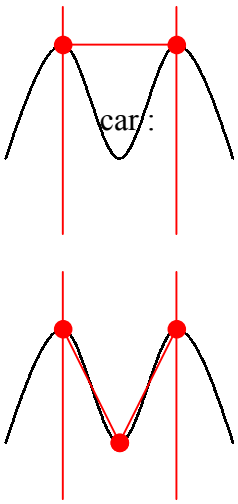
1500 bits/s

Le second cas est le meilleur si la ligne est très bonne.
Le troisième cas est le meilleur si la ligne est mauvaise.

3) Un son HIFI 16Hz → 20 kHz de bande passante. Son numérisé (technique MIC) avec 1024 niveaux de quantification.

a) Quel est le débit nécessaire à la transmission ?

débit binaire = 2 * la fréquence max



car :

insuffisant car le signal est perdu

D = 2 * f max * n bits
$1024 = 2^{10}$

$2 * (20*10^3)*10 = 4*10^5$ b/s

b) On désire multiplexer 8 canaux de ce type sur une voie HV sur un multiplexeur temporel à caractères hors bande. Proposer une structure de trame.

Hors bande
Signalisation dans la bande

1 IT de verrouillage (10 bits) 1 IT de verrouillage (11 bits)
8 IT de 10 bits
8 IT de 11 bits
1 IT de signalisation (10 bits)

→ 100 bits → 99 bits

V									

c) Quel doit être le débit binaire de la voie HV ?

10 * 400 kb/s → 4 Mb/s
(BV)

d) Efficacité du multiplexeur.

L'efficacité, c'est le rendement.
10 cellules en tout, 8 cellules de données → efficacité = 80 %.

SUPPORT & MODE DE TRANSMISSION

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

SUPPORTS & MODE DE TRANSMISSION

Les réseaux informatiques utilisent de supports physiques très variés pour la transmission de données.

I. LES SUPPORTS PHYSIQUES

1.1) Supports filaires :

- lignes « téléphoniques », paires torsadées :
  - 0,4 à 0,6 mm^2 de ∅
  - En cuivre
  - La paire torsadée est classée en catégories :

cat 3 → cat 5

cat 5 : 100 mb/s en ethernet, 125 mb/s en ATM, (cat 3 + blindage de la torsade qui est plus nombreuse), 2F le mètre, 100 mètres maximum.

- câble coaxial :
  - Caractérisé par son impédance (50Ω, 75Ω, 100Ω).
  - Besoin d'un terminateur de câble (bouchon) : résistance qui absorbe le signal pour éviter la réflexion de celui ci (en fait n'absorbe pas mais simule un câble infini).
  - 100 Mhz en standard ethernet, 500 Mhz en télétransmission.
  - 500 mètres maximum.

- fibres optiques :
  - Onde luminaire dans une fibre de silicium.
  - On peut aller jusqu'à plusieurs kilomètres.
  - Insensible au bruit.
  - Un seul sens de circulation (simplex) .
  - Plusieurs Gigabits de capacité.

1.2) Transmission d'ondes :

- HF (hyper fréquence) :

II. RNIS : Réseau Numérique à Intégration de Services

ISDN en anglais : Integrated Services data Network

Un seul raccordement chez l'abonné.
→ téléphone
→ données

Deux interfaces possibles offrant trois gammes de débit.
→ S
→ T

canal B à 64 kbits/s
canal D à 16 ou 64 kbits/s
canal Y à 48 ou 64 kbits/s
INTERFACE S :

Universelle pour l'utilisateur (téléphone, transmission de données). Bus passif. Vers un commutateur (PABX). Portée maximale 1Km. Paires téléphoniques normales.

S0 → 2 canaux B (64kb/s) + 1 canal D (16kb/s) → 144 kb/s
(pour particulier, accès de base)

S1 → 23 canaux B (64kb/s) + 1 canal D (64kb/s) → 1536 kb/s

S2 → 30 canaux B (64kb/s) + 1 canal D (64kb/s) → 1984 kb/s
(grosses installations, accès primaire)

INTERFACE R :

RNIS vers téléphone analogique (convertisseur).

INTERFACE T :

Interface T	Canaux	Débit utile	Débit de service	Débit réel
T0 de base	2 B + D(16) + Y(48)	144	48	192
T1	9 B + D(64) + Y(64)	640	64	504
T2 primaire	30 B + D(64) + Y(64)	1984	64	2048

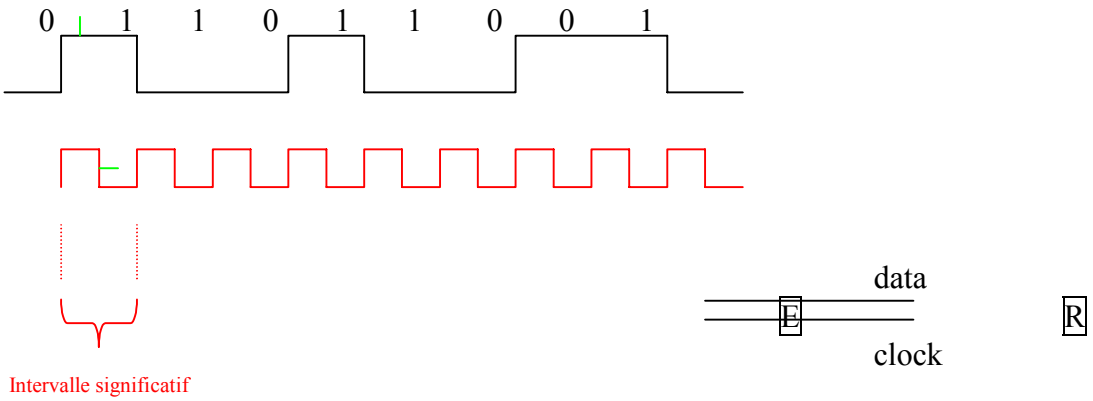
LES SUPPORTS PHYSIQUE

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LES SUPPORTS PHYSIQUES

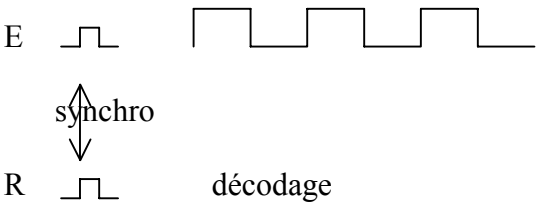
→ synchrone :

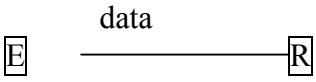
On transmet en même temps que les données un signal d'horloge.



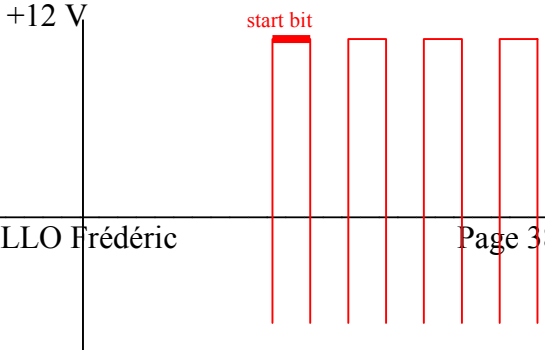
→ asynchrone :

on a une horloge coté émetteur et coté récepteur qui bat à la même vitesse ! ! Avant chaque transmission, on transmet un signal de synchro .





Exemple : transmission synchrone mode RS232, transmission caractère.



- La longueur du start bit détermine la fréquence à laquelle j'envoie des bits (largeur du signal transmis) → vitesse de la ligne.
- Stop bit : état de fin de transmission.
- Longueur du mot : 5, 6, 7 ou 8
- Parité

Mot 7 bit + parité
0 1 0 0 1 0 1 1

bit de parité : ajusté de manière à ce que le nombre total de bits positionnés à 1 soit pair.

Il peut y avoir paire, impaire, space = 0, mark = 1, sans parité (pas de bit de parité).

I. LES NORMES EXISTANTES

Normes de transmission pour les modems, analogique ;

- V21 : 300 b/s 2 fils bande de frequence=1080 et 1750 Hz
- V22 : 1200b/s 2 fils
  V22 bis
- V23 : (Minitel) 75/1200 b/s 2 fils

- V32 : 9600 b/s 2fils
- V34 : 28800 b/s
- V90 : 56000 b/s

Au niveau analogique, on devrait s'arrêter là car il y a maintenant le numérique.

Exercice : à débit binaire égale, quelles raisons feraient choisir la paire torsadée ou le câble coaxial ? ?

Paire torsadée	Câble coaxial
Avantages : - - cher	Avantages : - va plus loin - meilleure immunité au bruit
Inconvénients :	Inconvénients : - + cher - + fragile (courbure, écrasement, …)

Exercice : Quelles sont les limitations des transmissions par voie hertzienne ? ?

AVANTAGES	INCONVENIENTS
- pas de génie civile - large bande passante - longue distance	- être de vis à vis - conditions atmosphériques - problème de sécurité (interception ou brouillage possible du signal) - environnement

Exercice : Quelles sont les limitations des transmissions par infrarouge, micro-ondes, laser ? ?

AVANTAGES	INCONVENIENTS
- peu sensible aux perturbations magnétiques	- obstacles - confidentialité

II. Comparatif des différents supports de transmission

	Immunité au bruit	Prix	Contraintes d'utilisation	Affaiblissement	Vitesse de transmission
Paires torsadées	2	1	1	5	2
Câble coaxial	3	2	2	2	3
Fibre optique	4	5	5 à 6	1 à 2	4 à 5
Ondes	1	6	5	1 à 2	4 à 6

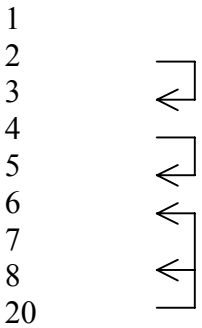
III. INTERFACES ETCD ETTD

L'avis V24 du CCITT décrit un ensemble de signaux de commandes et de signaux d'état (ressemble beaucoup à la RS232, diffère par les tensions → l'une à +12/-12, l'autre à +10/-10).

Correspond à un connecteur 25 points ou 9 points suivant le matériel.

- circuit 101 : terre de protection, broche 1
- circuit 102 : terre des signalisations, broche 7
- circuit 103 : émissions de données, broche 2 (ED)
- circuit 104 : réception de données, broche 3 (RD)
- circuit 105 : demande pour émettre, broche 4 (RTS :request to send, demande pour émettre)
- circuit 106 : prêt à émettre, broche 5 (CTS : clear to send)
- circuit 107 : poste de données prêt, broche 6 (DSR : data set ready)
- circuit 108/1 : connectez le poste de données sur la ligne, broche 20 (DTR : data terminal ready)
- circuit 108/2 : équipement terminal de données prêt
- circuit 109 : détection de signal sur la voie de données, broche 8 (CD : carrier detect)

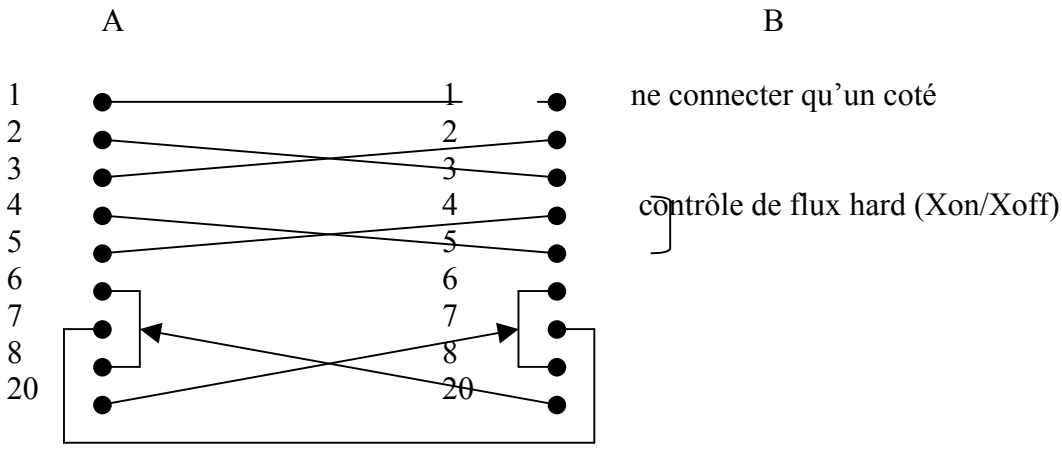
Exercice : avec l'hyper terminal Windows 95 ou un kermit du domaine public, vérifier que la liaison COM 1 fonctionne. Faire un « bouchon » qui renvoi les données émises sur la réception.



Exercice : Relier un ordinateur A à un ordinateur B.

D'habitude : A  B

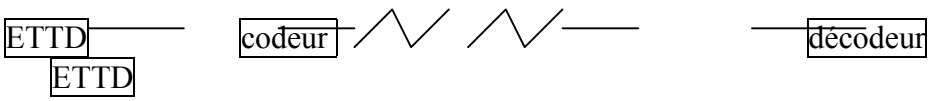
Remplacer le couple Modem/Modem par un câble (Null Modem), décrire ce câblage.



RESEAUX – Cours du CNAM BORDEAUX 1999-2000

PROTECTION CONTRE LES ERREURS DE TRANSMISSION

- à l'intérieur d'un ordinateur : 10^{-12} de probabilité d'erreur
- RTC de mauvaise qualité : 10^{-3}
- RTC de bonne qualité : 10^{-5},10^{-6}



Le codeur introduit de la redondance :

C(n,k) = r

n : bits à transmettre
k : redondance

Il existe 2 classes de codes :
- En bloc : les r bit rajoutés ne dépendent que des k bits d'information. Le plus utilisé avec détection et correction d'erreur. (détection d'erreur : codes polynomiaux, codes cycliques. Correction d'erreur : codes de Hamming, codes BCH).
- Conventionnels (ou récurrents).

I. DETECTION

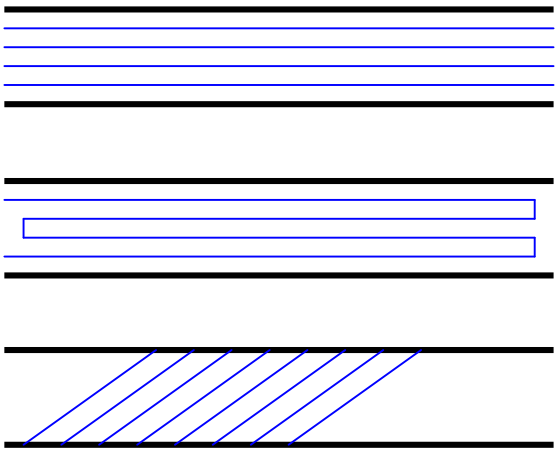
1.1) Détection d'erreur (vérif. de parité verticale et longitudinale)

A chaque caractère on rajoute un bit (bit de redondance verticale ou bit de parité, VRC :Vertical Redondancy Check). Lorsque le nombre de bits à 1 est pair, on emploi un code de parité pair (pour transmission asynchrone), sinon c'est parité impaire (pour transmission synchrone).

00110100 → parité impaire
00110101 → parité paire parité verticale

Dans ce cas, on est capable de détecter une erreur de parité, mais pas de la localiser.

La parité longitudinale était initialement utilisée pour les bandes magnétiques pour compléter la détection des erreurs de parité verticale.



avant

DLT

DAT ou hexabyte

A chaque bloc de caractère, on ajoute un champ de contrôle supplémentaire (LRC : Longitudinal Redondancy Check)

		Bit de redondance verticale
	0100011	0
	1011100	1
	11010101	
LRC	1101010	1

La combinaison de LRC et VRC permet de détecter 2 erreurs de bits dans un seul mot ou de corriger 1 erreur.

1.2) Détection d'erreur par code cyclique

Un code est cyclique s'il est linéaire (les bits de contrôle sont une combinaison linéaire des bits d'information) et si toutes les permutations d'un mot de code(suite de bits) restent un mot de code.
C = [000,101,011,100] est un code cyclique

Définition : Un mot de code est représenté sous forme polynomiale dans laquelle la suite de bits à transmettre $M=m_1m_2\ldots m_n$ est représenté par $Mx=u_1+u_2x+\ldots+u_nx^{n-1}$
Par exemple 1100101 est représenté par $x^6+x^5+0x^4+0x^3+x^2+x+1$ c'est à dire $x^6+x^5+x^2+1$

Principe de détection :
On utilise le reste R(x) de la division polynomiale M(x) par un polynôme diviseur G(x) qui donne un quotient Q(x). R(x) est calculé par l'émetteur puis transmis au récepteur. Le récepteur fait le même calcul R'(x) en divisant M(x)+R(x) (message + contrôle). Si R'=0 alors pas d'erreur, si R'≠0 alors erreur.

M(x) – R(x) est divisible par G(x) et est équivalent à M(x) + R(x) modulo 2.

Cette méthode fonctionne bien car la table de vérité de l'addition (en modulo 2) équivaut au ou exclusif. + → ou exclusif ⊕

Exemple de polynôme générateur :

L'avis V41 du CCITT est la norme.
$G(x) = x^{16}+x^{12}+x^5+1$

Ex : calcul de LRC en série
Transmission de 2 octets 01001101
01101111
00100010 = LRC

Ce mode de calcul n'est valable que si l'on transmet en parallèle les informations (ex : bande magnétique).
LRC sur 1 octet est un code cyclique de polynôme générateur x^8+1.

$M(x) = x^{14}+x^{11}+x^{10}+x^8+x^6+x^5+x^3+x^2+x+1$ (2 octets en série qui sont plus haut)
$x^8M(x)= x^{22}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8$ (on multiplie par x8 pour faciliter la division)

$$\frac{x^8M(x)}{x^8+1} =$$

$x^{22}+x^{19}+x^{18}+x^{16}+x^{14}+x^{13}+x^{11}+x^{10}+x^9+x^8$ | x^8+1

$-x^5-x$ | $x^{14}+x^{11}+x^{10}+x^8+x^5+x$

ou $+x^5+x$ |

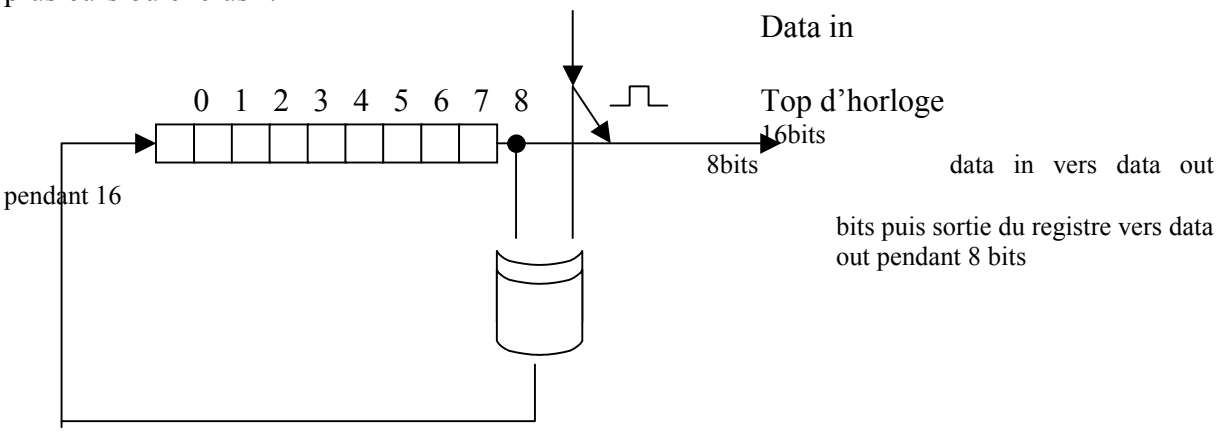
$R(x) = x^5+x$

↳ 00100010 → on trouve comme en haut ! !

On transmet donc : 010011010110111100100010

Je reçois donc cela et peut le diviser par x^8+1 pour vérifier qu'il n'y a pas d'erreur.

Pour fabriquer ces codes à la volée, on utilise donc un registre à décalage ainsi que un ou plusieurs ou exclusif.

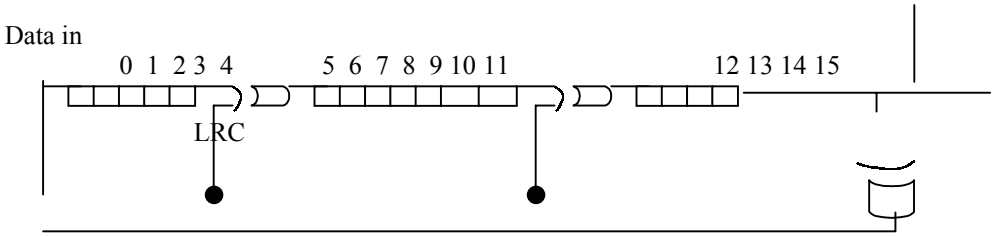


Rappel : ou exclusif
$1 \oplus 1 \rightarrow 0$
$0 \oplus 0 \rightarrow 0$
$1 \oplus 0 \rightarrow 1$

	Data in	ou exclusif	01234567
0	0		00000000
1	0		00000000
2	1		10000000
3	0		01000000
4	0		00100000
5	1		10010000
6	1		11001000
7	0		01100100
8	1		10110010
9	0		01011001
10	1		00101100
11	1		10010110
12	0		01001011
13	1		00100101
14	1		00010010
15	1		10001001
16	1		01000100
17	0		
18	0		
19	0		
20	0		
21	0		
22	0		
23	0		
24	0		
25	0		
26	0		
27	0		
28	0		
29	0		

30 0
31 0
32 0

Exercice :
Polynôme V41 : $x^{16}+x^{12}+x^5+1$
Quel serait le circuit à base de ou exclusif et de registre qui calcule le LRC ?



1.3) Procédure orientée bit (HDLC)

Initialisation du registre avec des 1.
L'émetteur transmet le FCS (Frame Control Sequence) complémenté (tous les 0→ 1, tous les 1 → 0). La valeur qui indique une transmission sans erreur est une séquence qui est toujours la même (« valeur magique »).

II. CODES CORRECTEURS

Ils permettent de détecter mais aussi de corriger une ou plusieurs erreurs. Ex :code correcteur d'erreur cyclique BCH (pour détecter des erreurs de correction ou des erreurs de synchronisation).

2.1) Code correcteur à vérification de synchronisation

BCH + traitement simple après codage.
BCH : Bose Chandhuri Hocquengheim

- Complémenter à 1 le $i^{ème}$ bit ou bien permutation du $i^{ème}$ bit et du $j^{ème}$ bit.
- A la réception, on effectue l'opération inverse.
- Si la division ne donne pas le bon résultat → erreur.

Exemple de correction d'erreur

Tout mot de code $X = (x_1, \dots, x_{15})$ ou $x_1 \dots x_7$ → données
$x_8 \dots x_{15}$ → bits de redondance
on doit vérifier $H X^t = 0$ (X^t : transposé de X) ou H est la matrice de contrôle de parité.

100010011010111
010011010111100
001001101011110
000100110101111
100011000110001
000110001100011
001010010100101
011110111101111

si les bits de données valent 0101100 $H X^t = 0$

$$1 + x^8 + x^9 + x^{11} + x^{13} + x^{14} + x^{15} = 0$$
$$x^8 + x^{10} + x^{11} + x^{13} = 0$$
$$x^9 + x^{11} + x^{12} + x^{13} + x^{14} = 0$$
$$1 + x^8 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} = 0$$
$$1 + x^{10} + x^{11} + x^{15} = 0$$
$$x^9 + x^{10} + x^{11} + x^{15} = 0$$
$$1 + x^8 + x^{10} + x^{13} + x^{15} = 0$$
$$1 + x^8 + x^9 + x^{10} + x^{12} + x^{13} + x^{14} + x^{15} = 0$$

La valeur du mot code = 010110000101010

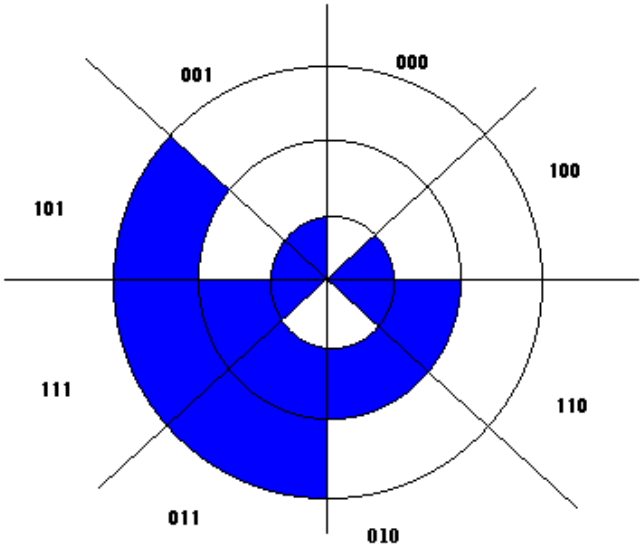
1er cas : détection d'erreur
si erreur sur le 9ème bit
Y= 0101100 01101010
Data FCS

$Y.H^t \Rightarrow$ on trouve $(10100101)^t \Rightarrow$ 9ème colonne de la matrice, on corrige donc le 9ème bit.

2ème cas : détection d'erreur de synchronisation
On décide de complémenter à 1 le 9ème bit pour détecter les erreurs de synchronisation. On reçoit 0101100011010100, le récepteur complémente à 1 le 9ème bit. Z = 101100010010100.
$H.Z^t \Rightarrow (01110110)^t \Rightarrow$ somme de la 8ème et de la 9ème colonne, on corrige donc les 2 bits concernés.

Exercice : code de Hamming

3 bits, contrainte deux codes contiguës ont 1 seul bit qui change.



2 bits	3 bits
00	000
10	100
11	110
01	010
	011
	111
	101
	001

PROTOCOLES DE COMMUNICATION

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

PROTOCOLES DE COMMUNICATION

I. GESTION DE LA LIAISON DES DONNEES

1.1) Protocoles

Logiciel de niveau II (norme ISO, modèle OSI)

- Responsable de :
  - L'acheminement sans erreur dans les blocs d'informations (messages, trames) sur une ou plusieurs liaisons physiques qui peuvent être permanentes ou non (commutées).

- Tâches à exécuter par la couche liaison :
  - Etablissement et libération de la liaison de données sur des connexions physiques préalablement activées
  - Détection des erreurs de transmission et activation des procédures de reprises en cas d'erreur, et éventuellement prévenir la couche supérieure d'une erreur.
  - Supervision du fonctionnement de la liaison de données selon le mode de transmission (synchrone ou asynchrone), selon la nature de l'échange (unidirectionnelle ou bidirectionnelle), selon le type de liaison (point à point, multi-points, boucle).
  - Définir la structure syntaxique des messages valides.

1.2) Deux familles de procédures (protocoles)

- Orienté caractères (BSC d'IBM pour caisses enregistreuse) → procédures de constructeurs souvent, fonctionne à l'alternat, de type envoyer et attendre. (ISO 1745, ISO 2111, ISO 2628, ISO 2629).
- Orienté bits. HDLC : bidirectionnel simultané haut débit.

1.3) La procédure BSC

- Binary Synchronous Communication :

  Fait partie des procédures constructeur. Etudié et implémenté à partir de 1960. Il y a eu aussi UIP de BULL, TMM de CII.
  Basé sur une transmission synchrone de blocs de caractères quel que soit le codage utilisé (ASCII, EBCDIC, BST) permettant d'exploiter 2 types de liaison :
  - Point à point : communication à l'alternat (Half duplex). En cas de conflit, la station «primaire» prend le contrôle au dépend de la station « secondaire ».
  - Multi-points : la station « primaire » représente le seul nœud central par rapport aux stations « secondaires ».

→ mode poolling (interrogation)
→ mode selecting (sélection) ou adressing (adressage)

- Caractères utilisés pour BSC :

  - SYN : synchronous idle. Utilisé pour la synchronisation caractère.
    début de message SYN SYN SYN message
  - ENQ : enquiry. Invite une station à émettre ou recevoir.
  - SOH : start of heading. Signale un début d'en-tête.
  - STX : start of text. Fin de l'entête et début du texte.
  - ETB : end of transmission bloc. Fin de bloc intermédiaire de donnée ou fin de message.
  - ETX : end of text. Fin de texte + début de caractère de contrôle servant à la détection d'erreur.
  - ACK : acknowledgement. Accusé de réception positif du message.
  - NACK : negative acknowledgement. Accusé négatif, il faut réémettre.
  - DLE : data link escape. Caractère de non prise en compte de caractère suivant pour le protocole (mode transparent).
  - EOT : end of transmission. Fin de transmission.
  - BCC : block check caractere. Bloc de détection des erreurs.

- Synchronisation de caractères

  Série de SYN. Toutes les secondes « SYN SYN » pour éviter une perte éventuelle de synchronisation. Au début, il y en a même de 4 à 7 à suivre.
  Il y a une séquence de remplissage (PAD) → padding pour temporiser une coupure.

- Temporisation

  A l'alternat, il faut respecter un délai maximum de réponse (3 secondes pour une station réceptrice). Au bout d'un certain temps, on passe en mode dégradé puis 2 ou 3 essais avant de déclarer la ligne Hors Service.

- Format des messages :

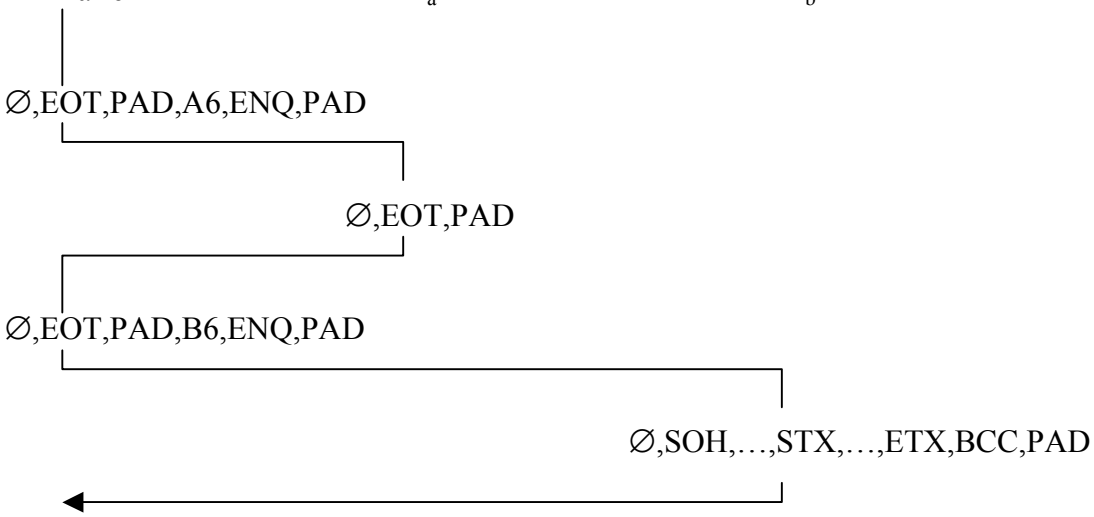
  Pour simplifier, SYN SYN SYN = ∅
  Message = bloc = groupe de caractères.
  - L'entête seule : ∅,SOH,ETX,BCC,PAD
  - Message sans entête : ∅,STX,…,ETX,BCC,PAD
  - Message avec entête : ∅,SOH, …,STX,…,ETX,BCC,PAD
  - Plusieurs blocs pour un même message sans entête : ∅,STX,…,ETB,BCC,PAD pour les premiers blocs et ∅,STX,…ETX,BCC,PAD pour le dernier bloc.

- Détection des erreurs

  Polynôme V41 $x^{16}+x^{12}+x^5+1$ = CRC16
  Dans BSC → de 12 à 16 bits pour le BCC

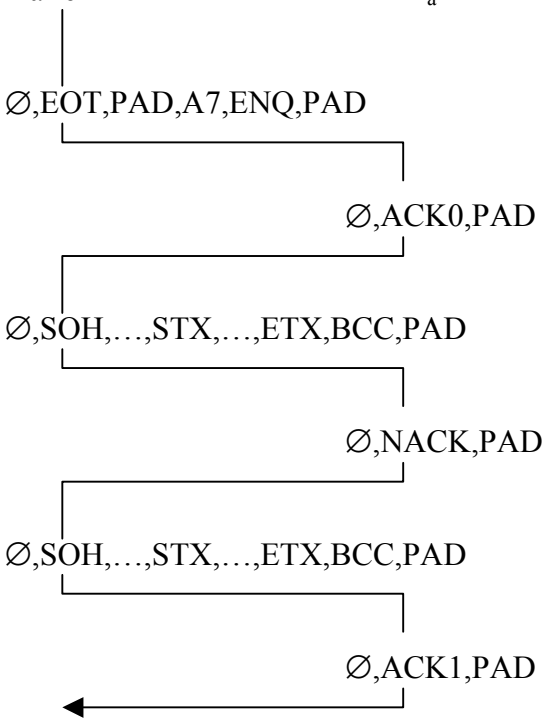
1.4) Déroulement du protocole en liaison multi-points

- Primaire $ETTD_a$ $ETTD_b$



- Sélection de $ETTD_a$ avec erreur + retransmission

Primaire $ETTD_a$ $ETTD_b$



EXERCICE :

On branche un terminal synchrone relié en multi-point à un ordinateur central. Le terminal utilise un protocole de type envoyer-attendre.

a) En phase de test, on remarque que des données arrivent mais ne provoquent aucune réaction de la part du terminal. On remarque que le terminal reçoit des caractères codés 4C (en base 16) au lieu e B3 (en base 16). A quoi est dû le problème ? ?

4C → 01001100
B3 → 10110011

Tous les bits sont inversés → on a donc dû inverser les fils d'une paire.

b) Le terminal reçoit une séquence de supervision, répond à cette séquence et reçoit à nouveau la même séquence. Pourquoi ? ?

Tout se passe en fait comme si l'ordinateur maître n'avait pas reconnu la réponse. Il y a un problème de synchronisation entre le matériel et le logiciel. Remède : tester le circuit 106 (prêt à émettre).

c) Ultérieurement, on constate que les messages de données sont systématiquement invalides alors que la syntaxe des messages émis par le terminal est OK.

On retourne le modem trop tôt avant que le dernier caractère ne soit arrivé. Remède : temporiser en envoyant du padding.

d) Par la suite, la réponse à une séquence de poolling est toujours EOT. Pourquoi ? ?

Problème d'adressage, mauvaise initialisation des adresses.

COUCHE LIAISON

LE NIVEAU LIAISON DU MODELE OSI.....58

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LE NIVEAU LIAISON DU MODELE OSI

I. Introduction

Cette couche conditionne les bits bruts de la couche physique en trames de données. La couche liaison de données est également chargée du contrôle d'erreurs qui s'effectuent en s'assurant que les bits de données reçues sont identiques à ceux qui ont été envoyées. En bref :

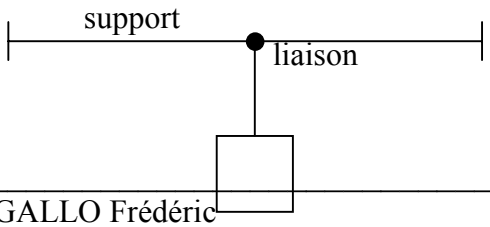
- Elle envoie les trames de donnes depuis la couche réseau a la couche physique.
- Elle regroupe les trames de bits de la couche physique et attend un accuse de réception
- Sous couche LLC : elle définit des point d'accès au services SAP
- Sous couche MAC : Elle communique directement avec la carte réseau. C'est elle qui est responsable du transfert sans erreurs des trames

La couche liaison de données prend les données de la couche physique et fournit ses services à la couche réseau. Les bits reçus sont regroupé en unité logiques appelées trames. Dans le contexte d 'un réseau, une trame peut être une trame Token Ring ou Ethernet, FDDI, ou un autre type de trame réseau. Pour les liens des réseaux étendus, ces trames peuvent être des trames SLIP, PPP, X.25 ou ATM. Les bits d'une trame ont une signification spéciale. Le début et la fin d 'une trame peuvent être marqués par des bits spéciaux. De plus, les bits de trame sont répartis en champ adresse, champ de contrôle, champ de données et champ de contrôle d'erreurs.

- Les champs d'adresses contiennent les adresses source et destination.
- Le champ de contrôle indique les différents types de trames de liaison de données.
- Le champ de données contient les données proprement dites, transmises par la trame.
- Le champ de contrôle d'erreurs détecte les erreurs dans la trame de liaison de données.

La couche liaison de données est la première couche qui gère les erreurs de transmission. En général, le champ de contrôle d 'erreur consiste en un générateur de checksum, utilisé pour détecter les erreurs dans la trame de liaison de données. Dans la plupart des cas, les réseaux modernes utilisent un contrôle de redondance cyclique (CRC). Pour les réseaux locaux, c 'est un CRC 32 bits. Pour les réseaux étendus où les liens sont plus lents, on utilise un CRC à 16 bits pour éviter de surcharger la liaison. Dans les réseau TCP/IP, les implémentations de la couche liaison de données comprennent les technologies suivantes : Token Ring, Ethernet, FDDI, Frame Relay, X.25, SLIP, PPP et ATM.

On définit plusieurs types de réseau selon leur étendue: LAN (Local Aera Network), WAN (Wide Aera Network) et MAN (Metropolitan Aera Network): plaques régionnales (fédérées par entité administrative). *Ex : en aquitaine le MAN aquarelle*.



machine

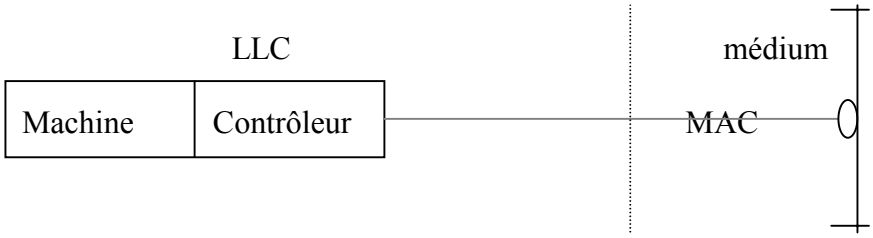
Il y a deux sous-niveaux dans la couche liaison du modèle OSI :

- Les technologies d'accès au support (Medium Access Control = MAC). Commun à toutes les stations connectées sur le réseau local.
- Procédure de communication : niveau de commande du lien logique (Logical Link Control = LLC). Cela décrit la manière de prélever les informations du support bit par bit.

NOTE : Les protocoles TCP/IP ou ATM ne respecte pas le modèle OSI.

- Normes IEEE (*reprises par l'ISO sous IS8802.x*) :
  802.1 → Fonctions de gestion des couches définies pour l'administration de LAN.
  802.2 → LLC
  802.3 → CSMA/CD (ethernet, …)
  802.4 → bus à jeton
  802.5 → anneau à jeton (token ring)
  802.6 → définition de réseaux métropolitains

- Norme FDDI (Fiber Distributed Data Interface)
  Débit à 100Mb/s, équivalant à 802.5 pour fibre optique.



II. La couche LLC

Couche dépourvue du codage analogique : on récupère les bits. Réalisé à la limite du hardware et du software (firmware EEPROM). Les services rendus par la couche LLC aux couches supérieures sont spécifié par 3 classes :

LLC1 : service sans connexion et sans acquittement. Le travail est fait dans les couches supérieures ou on accepte de perdre des données (*ex : visio conf et temps réel*)les couches supérieures assurent la reprise en cas d'erreur).
LLC2 : service avec connexion *ex : porteuse* (pour les transmissions longues de fichiers,).
LLC3 : service sans connexion et avec acquittement Cela évite de maintenir une table active : datagramme. En fait, on écoute en permanence car il y a des diffusion d'écoute (on arrose tout le monde).

2.1) Caractéristiques de LLC :

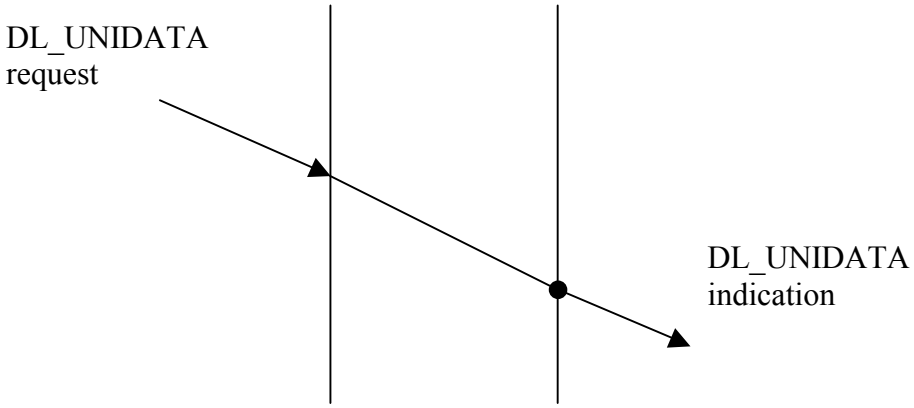
- Assure le contrôle de flux (de type « stop and wait » sans connexion pour LLC3). Mécanisme de type « fenêtre »(buffer d'anticipation) avec connexion pour LLC2.

- Assure le contrôle d'erreurs à l'aide d'un CRC de 32 bits qu'il rajoute au niveau MAC car la taile des trames émises en LAN est supérieure par rapport au WAN.

NOTE : En Ethernet, c'est la couche MAC qui détecte les collisions et qui demande la réémission.

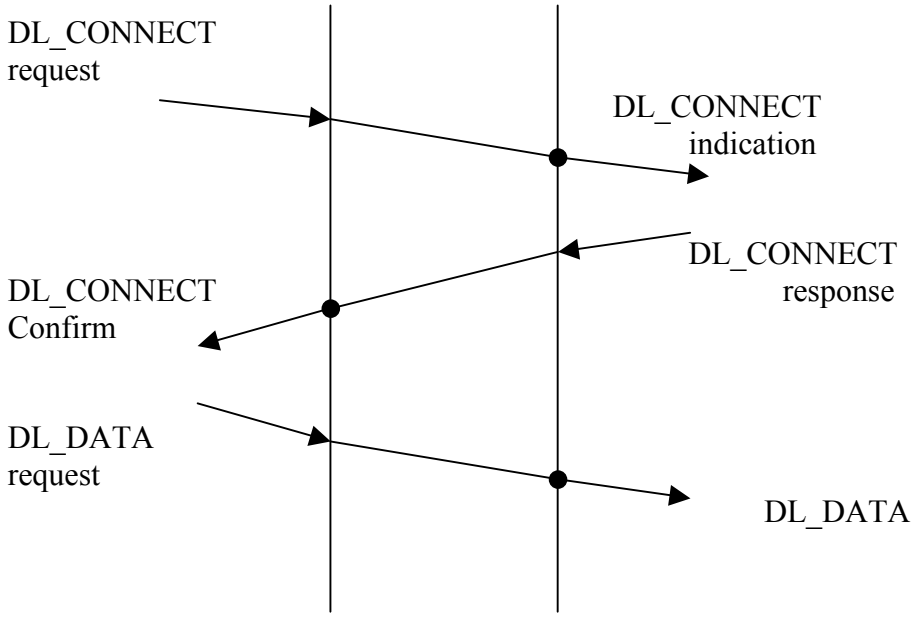
Classe LLC 1 :

2 primitives



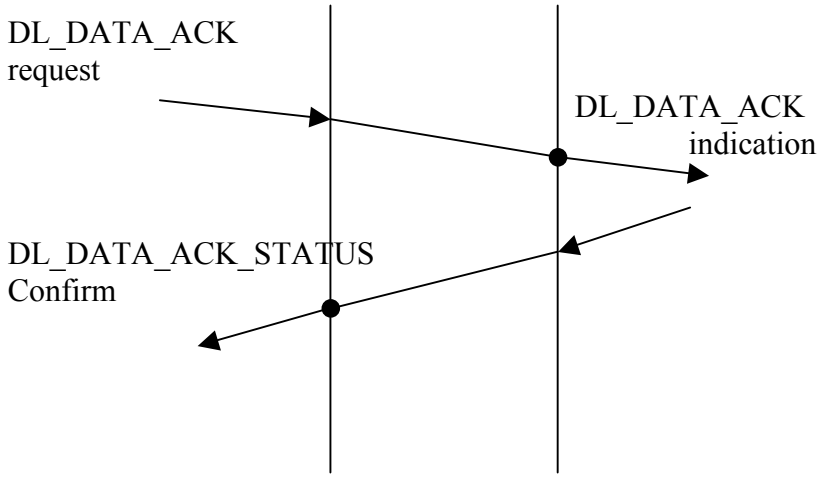
Classe LLC 2 :

3 classes successives :
. établissement d'une connexion entre 2 utilisateurs à des points d'accès au service
(service Access Point → SAP)
. Transfert de données
. Déconnexion.



Indication

Classes LLC 3 :

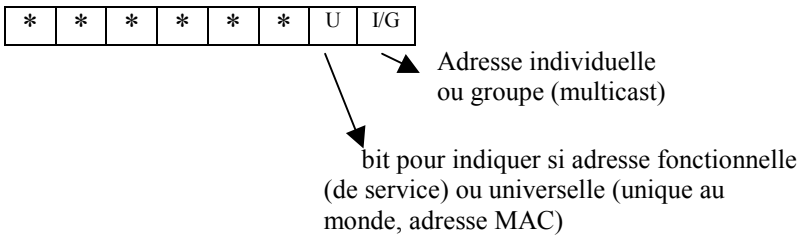


2.2) Structure des trames LLC

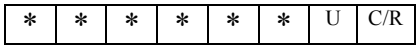
DSAP	SSAP	Commande	information

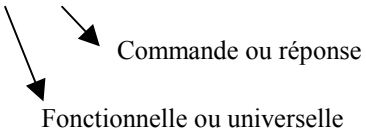
DSAP : Destination Service Access Point (1 octet)
SSAP : Source Service Access Point (1 octet)

Format DSAP



Format SSAP





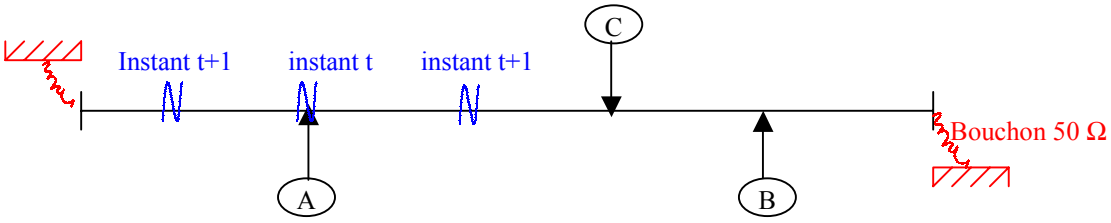
III. La couche MAC

3.1) Norme 802.3 (Ethernet) :

Introduction :

L'origine d'Ethernet vient des constructeurs Dec, Xerox, Intel en 1980. Basée sur CSMA/CD : *Carrier Sense Multiple Access / Collision Detection*. Cette technique, utilisée dans les réseaux Ethernet, est en fait basée sur la détection d'une porteuse avec accès multiple et détection de collision. C'est à dire que l'on peut avoir plusieurs éléments connectés qui peuvent prélever des bits sur le câble.

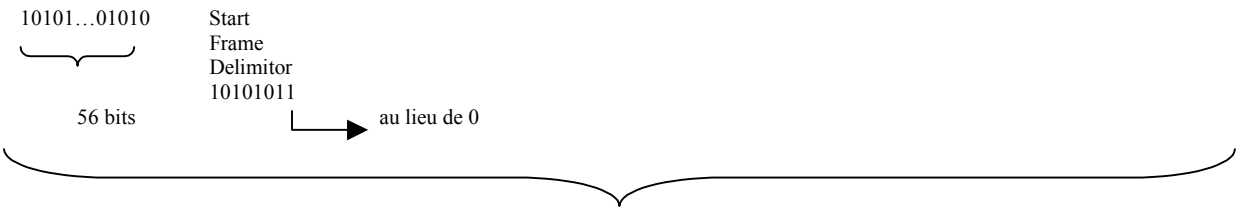
La vitesse est d'environ 0,7c (c=vitesse de la lumière).



Une collision se produit quand deux signaux sont émis en même temps (deux stations discutent en même temps). Les machines attendent le temps d'une temporisation et ensuite réémettent. C'est une technique de contrôle de flux probabiliste, c'est à dire la probabilité que la collision se produise doit être le plus faible possible. Mais les collisions dépendent donc du trafic et l'on a typiquement le "phénomène d'effondrement" qui caractérise Ethernet. Ce phénomène se produit à environ 7Mb/s pour un réseau Ethernet à 10Mb/s.

Format d'une trame Ethernet :

préambule	SFD	destinataire	source	Lg LLC	données	PAD	CRC
7 octets	1 octet	6 octets	6 octets	2 octets			4 octets

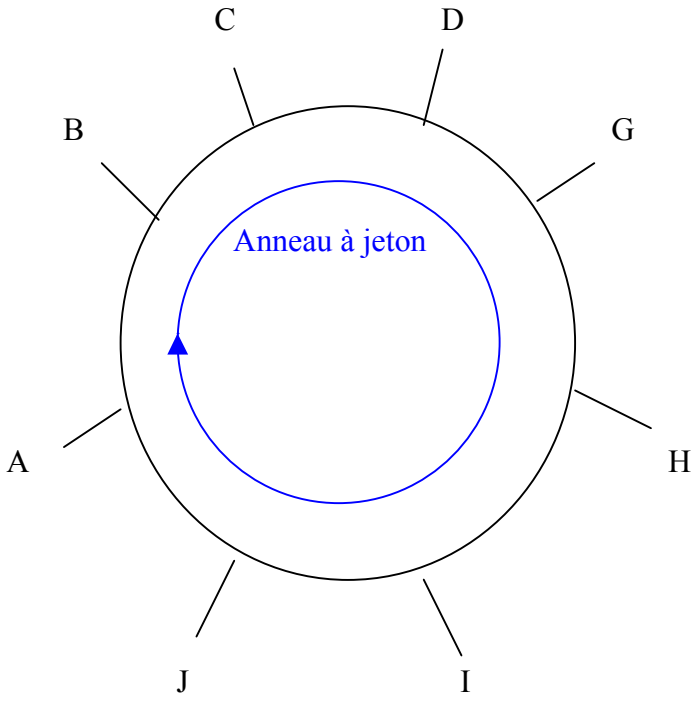


De 64 à 1518 octets

Supports Physiques :

Coaxial	- 10 BAS 2	→ Ethernet fin
	- 10 BAS 5	→ Ethernet "gros"
Paires torsadées	- 10 BAS T	→ Ethernet paires torsadées (10Mb/s)
	- 100 BAS T	→ Fast Ethernet (100Mb/s)
Fibres Optiques	- 100 BAS F	→ Ethernet fibres optiques (100Mb/s)
	-	→ Ethernet Gigabit (1Gb/s)

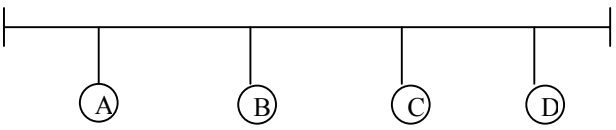
Principe de fonctionnement d'un commutateur ethernet :



En résumé, le principal inconvénient du 802.3 ce sont les collisions qui amène le phénomène d'effondrement du réseau.

3.2) 802.4 : Token Bus

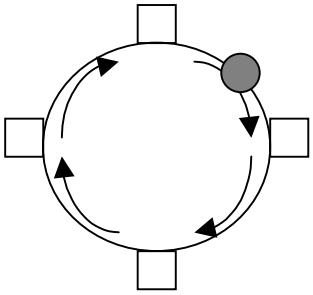
Fondé sur le droit d'émission c'est à dire qu'il dépend d'un « jeton ». On partage le support (un BUS) en « exclusion mutuelle ». En fait, on utilise le support en garantissant les délais de transmission mais l'utilisation n'est pas optimum car on charge le réseau inutilement. On a une liste de successeur avec une initialisation du réseau où est fabriqué un anneau virtuel. Le jeton est toujours transmis au successeur. En cas de panne d'une machine, on a une perte du jeton et une station "superviseur" réémet le jeton au bout d'un moment.



Ce type de réseau est peu utilisé pour des raisons de lourdeur d'initialisation du réseau. Ces avantages sont qu'il est très simple à mettre en œuvre et que sa topologie est en bus (on peut rajouter et enlever des stations sans problème : modulable). Par contre, il y a un manque de performance qui est lié au contrôle du jeton et au nombre de machines connectées.

3.3) 802.5 : Token Ring

Le fonctionnement de ce réseau est très lié à son architecture : un anneau. En fait le jeton est une sorte de « wagon » dans lequel on met des choses. Ce réseau est très compliqué et assez fragile. La vitesse est liée au nombre de stations puisque le jeton doit nécessairement faire le tour de l'anneau. Si l'on coupe le câble à un endroit : tout le réseau est bloqué. L'avantage est que l'on sait facilement où se trouve une station car on a une circulation en anneau du jeton.



A un instant T, tous les terminaux de l'anneau font la même chose.

En fait, sur cet anneau, on a une circulation de trames (contenant des informations plus l'identifiant du destinataire) permanente qu'elles soient pleines ou vides. Le débit binaire est garanti en permanence car le prélèvement de l'information dans le jeton ne ralentit pas vraiment le réseau. Chaque information est déposée de façon synchrone (a moyen d'un droit de parole : jeton). En vitesse de croisière cela marche bien mais en cas de problème c'est très lourd à gérer.

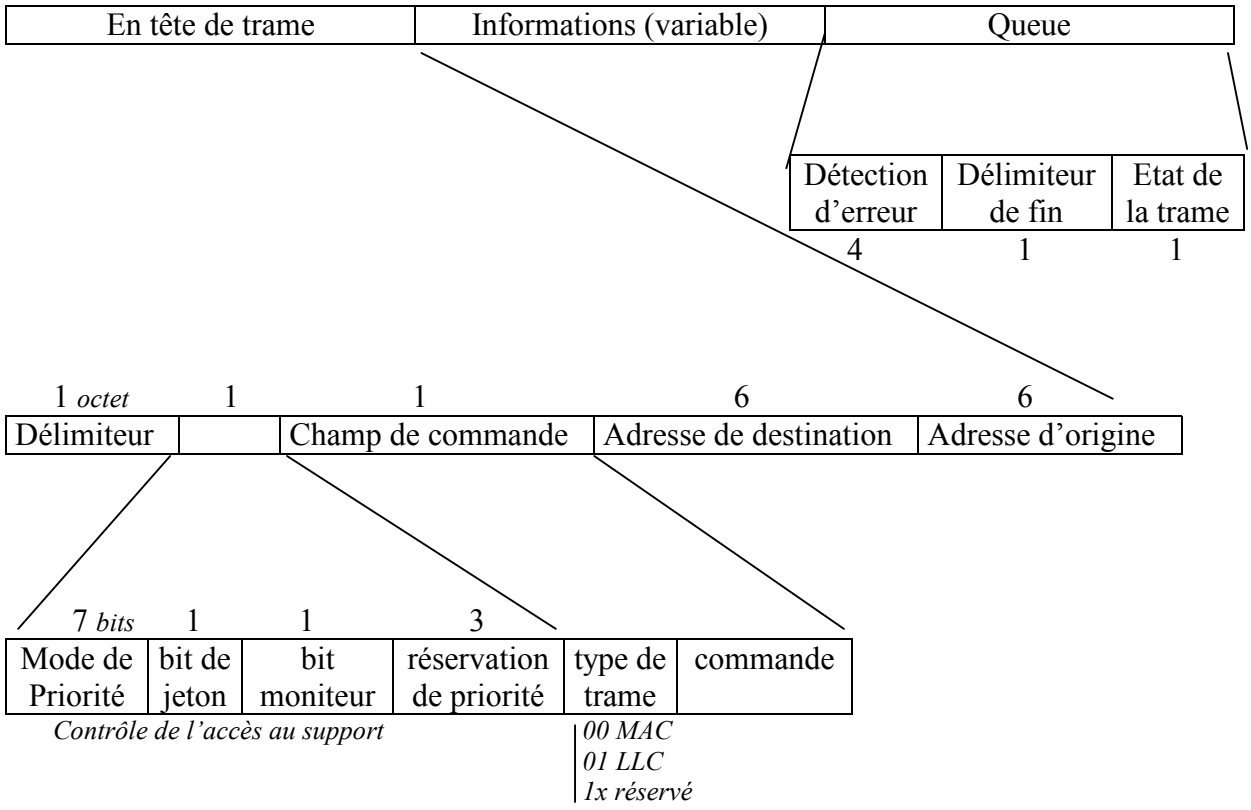
Principe du moniteur (superviseur du réseau) :

Il faut une station superviseur sur le réseau qui fasse la gestion du jeton. Elle veille au bon fonctionnement en faisant une régénération du jeton en cas de perte (time out). Si le moniteur est défaillant (panne ou autre), il n'envoie plus de trames signalant sa présence (trame : ACTIVE_MONITOR_PRESENT, MAC type AMP), alors une station qui peut devenir moniteur prend sa place (elle envoie une trame CLAIM_TOKEN). Ces stations qui peuvent

devenir moniteur (stations dormantes) se manifestent de temps en temps en envoyant des trames STANDBY_MONITOR_PRESENT (SMP). Tout ceci évidemment alourdit le protocole mais cela permet de savoir le nombre de stations sur le réseau et de savoir si elles sont actives.

Le moniteur a pour fonction la gestion du jeton mais aussi la surveillance des trames qui bouclent (*ex: mauvaise adresse destinataire*), et il gère le bon passage du jeton (trame de 24 bits) en fonction du temps de circulation sur l'anneau.

Format d'une trame MAC de Token Ring



Adressage Token Ring :

Chaque adresse physique des stations est connue (adresse MAC). Les adresses sont uniques (*sauf cartes bas de gamme type Taiwan*). Mais il existe également des adresses physiques de diffusion. D'ailleurs c'est beaucoup plus lourd que les diffusions d'Ethernet car il lui faut le temps de faire le tour (en Ethernet, tout le monde voit la trame en même temps).

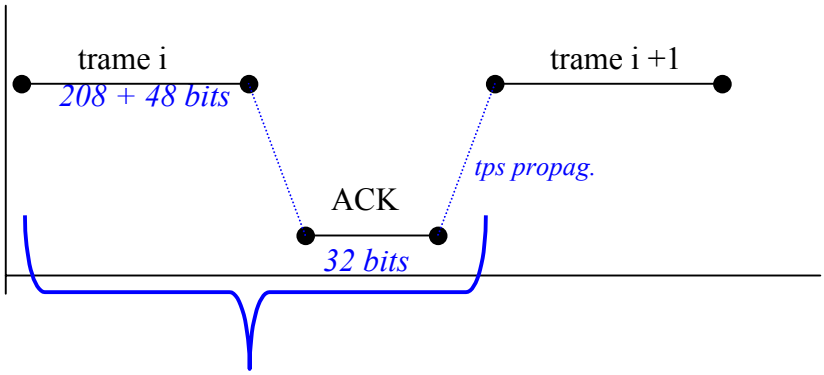
3.4) EXERCICE :Réseau 802.3 à 10Mb/s

Soit un réseau 802.3, avec un débit binaire de 10 Mb/s, de 800 m de longueur, la propagation géographique des signaux est de 200 m/µs. Les trames MAC contiennent 256 bits au total et l'intervalle d'après on a un accusé de réception de 32 bits (qu'on attend avant l'envoie de la trame suivante).

a) Quel est le nombre de bits en transit sur le bus à un instant donné ?

Sachant qu'un bit met 4 µs pour faire tout le réseau (800 m) et que nous avons un débit de 10Mb. Cela nous donne au total $4.10^{-6} * 10.10^{6} = 40$ bits sur le réseau (un tous les 20 m).

b) Quel est le débit efficace du réseau en supposant qu'il y a 48 bits de service (champs MAC+LLC) dans chaque trame ? *(En considérant que l'on a une taille de fenêtre de 1 bit pour l'accusé de réception).*



Temps de la trame : 256 bits sur 10^7 b/s
Temps de propagation : 4 µs
Temps de l'accusé : 32 bits sur 10^7 bits/s

On a donc la trame qui est validé au bout de :

Tps trame + tps propag. + Tps accusé + tps propag. = $\frac{256}{10^7}+\frac{4}{10^6}+\frac{32}{10^7}+\frac{4}{10^6}$ = 36,8 µs

Les données d'informations (pour le débit efficace) sont de 256 – 48 = 208 bits.

Donc le débit efficace est de : $\frac{(256-48)}{36{,}8.10^6} \approx 5{,}65$ Mb/s

3.5) EXERCICE : Réseau en Anneau

On considère un réseau local en anneau comprenant 50 stations. Le débit binaire est de 4Mb/s. Les trames MAC ont une longueur totale de 512 bits dont 32 sont utilisés par le protocole LLC.

a) Quel est le débit binaire maximum garanti à chaque station ?

Nombre de bits utilisés par trame : 512 – 32 = 480 bits *(de data)*

$\frac{4.10^6}{50}$ bits/s disponibles par station donc 80 Kb/s au total.

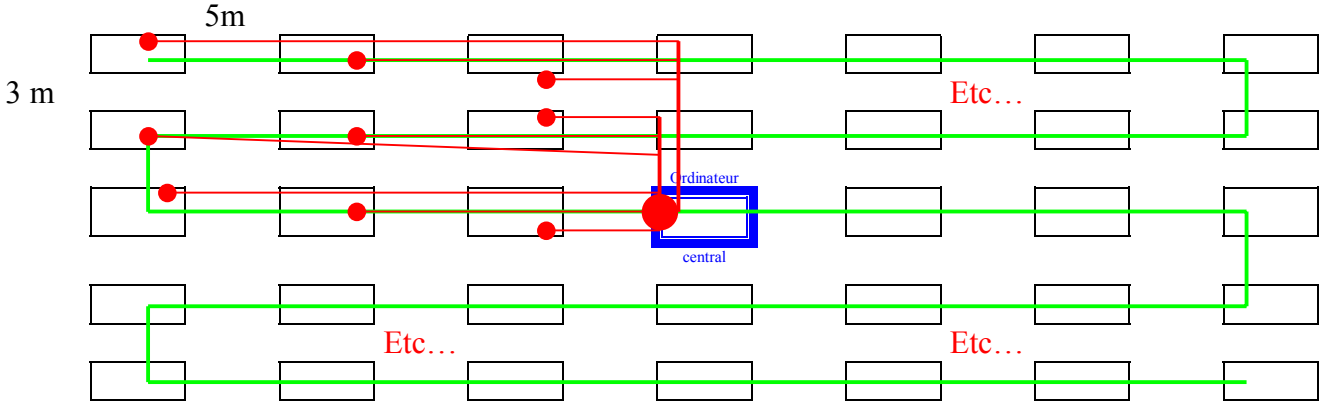
Le débit utile est donc de $\frac{80.10^3 * 480}{512}$ = 75 Kb/s

b) Peut-on transmettre de la parole numérisée sur le réseau ?

On sait que pour transmettre de la parole, il faut environ 64 Kb/s, ici c'est donc possible.

3.6) EXERCICE : Câblage d'un LAN

On cherche à câbler un réseau local dans un bâtiment de 5 étages avec 7 pièces/étage (5 m de largeur sur 3 m de hauteur). Sachant que la première station se trouve au 4ième bureau du 3ième étage et qu'il faut une prise dans chaque pièce.



1) Quelle est la quantité de câble nécessaire pour un réseau en bus, avec un câblage en bus ?

On a 6x5=30 m/étage de longueur. En comptant les 3 m de hauteur cela nous fait au total : (30 + 3) x 5 = 165 m mais il faut enlever la hauteur d'un étage puisque l'on considère que l'on place la prise à la même hauteur dans chaque bureau donc on a 165-3= 162 m

2) Quelle est la quantité de câble nécessaire pour un réseau en bus, avec câblage en étoile ?

D'après le schéma, il faut considérer chaque étage indépendamment. On a à chaque étage: (5+10+15)x2=60 m de longueur le long des bureaux. Sachant que l'on a 5 étage cela fait donc 5x60=300m en longueur.

Pour le 3ième étage : il n'y a pas de hauteur à considérer.
Pour les 2ième et 4ième étages : 7 stations par 3 m soit de hauteur.

Pour les 2ième et 4ième étages : 7 stations par 6 m de hauteur.
On en déduit donc le total : 300 + (21x2) + (42x2) = 426 m

3) Quelle est la quantité de câble nécessaire pour un réseau en anneau, avec câblage en bus ?

On reprend le câblage du 1) puis on rajoute le câble nécessaire pour faire la boucle. Soit 12m de hauteur et 15m de longueur pour partir du premier ou du dernier vers le centre.
Donc cela fait 162+2x(15+12) = 216 m.

4) Quelle est la quantité de câble nécessaire pour un réseau en anneau, avec câblage étoile ?

Il suffit en fait de centraliser l'anneau au centre de l'immeuble ce qui permet de reprendre la même quantité de câble qu'au 3) soit 426 m.

COUCHE RESEAU ET X.25

LE NIVEAU RESEAU DU MODELE OSI ERREUR ! SIGNET NON DEFINI.

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LE NIVEAU RESEAU DU MODELE OSI

I. INTRODUCTION

La couche réseau est chargée d'adresser les messages et de convertir les adresses et noms logiques en adresses physiques. Elle détermine aussi l'itinéraire à emprunter de la source à l'ordinateur de destination. Elle choisi le chemin que doivent suivre les données en fonction des contions du réseau, de la priorité du service et d'un certain nombre de facteurs. Elle gère aussi les problèmes de trafic comme la communication, l'acheminement et l'encombrement des paquets de données sur le réseau.

- Responsable de l'adressage, de la traduction des adresses en nom logique.
- Définie le routage des paquets.
- Gère les problème de trafique, commutation de paquets, encombrement.

La couche réseau gère les connexions entre les nœuds du réseau. Un service supplémentaire, fourni par la couche réseau, concerne la façon de router les paquets entre les nœuds d'un réseau.

La couche réseau sert à éliminer les congestions et à réguler le flot des données ; Cette couche permet aussi à deux réseaux différents d'être interconnectés en implémentant un mécanisme d'adressage uniforme. Token Ring et Ethernet possèdent, par exemple, différents types d'adresses. Pour interconnecter ces réseaux, vous avez besoin d'un mécanisme d'adressage compréhensible par les deux réseaux. Pour les réseaux TCP/IP, la couche réseau est implémentée en utilisant le protocole IP.

II. SERVICES fournis par la COUCHE RESEAU

- Adressage : permet de repérer chaque ETTD raccordé au réseau de manière unique.
- Contrôle de flux : permet de s'adapter aux vitesses (collisions en ethernet par exemple)
- Routage : permet de sélectionner le chemin entre deux adresses.
- Détection et correction d'erreur

Services facultatifs :

- Liaison séquentielle des paquets
- Données express (ex :CTRL + C)

III. TYPES de SERVICES utilisables

- Service avec connexion (CONS : Connection Oriented Network Services).

Notion de circuit virtuel utilisé dans X25 (circuits préalablement établis).

- Service sans connexion (CLNS : Connection Less Network Services).
  Paquets (datagrammes) est transmis de manière autonome. Possibilité d'éclatement et de concentration des paquets du message. Voir UDP

Notons que le protocole IP seul est un protocole non connecté.

IV. LA NORME X25

4.1) Introduction

Adopté en septembre 1976 par le CCITT, le protocole X25 définit l'interface entre un ETTD (Equipement Terminal de Traitement des Données) et un ETCD (Equipement Terminal de Circuit de Données). Il résulte de l'expérience accumulées sur différents réseaux à commutation de paquets.

On entend souvent par X25 l'ensemble des protocoles liés à X25 et qui couvre les couches 1 à 3 du modèle OSI de la norme ISO. Le niveau physique provient principalement de la norme X.21, la couche liaison est constitué par un sous-ensemble de la norme HDLC : le protocole LAP-B. Pourtant, le terme X25 désigne uniquement le niveau 3 ou niveau paquet transporté entre les champs d'information des trames LAP-B.

Le protocole X.25 est donc en premier lieu une interface locale entre un équipement informatique connecté au réseau et le réseau lui-même. Cependant, la définition de X25 a été étendue pour prendre en compte des transmissions entre des ETTD.

- 1er niveau : niveau physique : la norme préconise l'usage de X21.
- 2ème niveau : niveau liaison : la procédure choisie est LAP-B.
- 3ème niveau : niveau réseau : définit une interface locale avec transmission de paquets.

Pour résumer, nous avons un protocole en mode connecté avec un circuit virtuel. Suite à la sous utilisation des lignes allouées et pour réutiliser les temps de silence, il y a une optimisation des lignes. Ce qui a pour effet que la facturation est faite sur la durée et le volume de la communication (indépendant de la distance).

LAPB

Le protocole LAPB est le protocole de niveau 2 qui transporte les paquets X25. Le format standard d'une trame LAPB est le suivant:

Flag	Champ adresse	Champ de contrôle	Données	FCS	Flag

Flag: Toujours 0x7E
Address Field: Ce champ n'a aucune raison d'être quand on travaille de point à point. Cet octet est réservé à plusieurs utilisations. Il sert à séparer les commandes des réponses et peut

seulement prendre les valeur 0x01 et 0x03. 01 désigne une commande de l'ETTD à l'ETCD et 03 contient une réponse de l'ETCD à l'ETTD.
Champ de contrôle: Identifie le type de trame. En plus, il inclut la séquence de nombre, les fonctions de contrôles et le traquage des erreurs en fonction du type de trame.
FCS: Frame Check Sequence.

Types de trame:

Trames de supervision:

RR : Prêt à recevoir.
REJ : Demande de retransmission.
RNR : Pas prêt à recevoir.

Trames non séquentielles:

DISC : Demande de déconnexion.
UA : Trame d'acquittement.
DM : Réponse à DISC, mode déconnexion.
FRMR: Rejet de trame.
SABM: Mode asynchrone, pas de maître et d'esclave.

Trame d'information:

INFO

4.2) Format général d'un paquet

Notons que les paquets sont de petite taille puisque optimisés pour de longues distances.

Bits 8 7 6 5	4 3 2 1
Identificateur général de format	N° de groupe de Voie logique
N° de Voie Logique	
Identificateur de Type de paquet	

Trois octets d'en-tête qui servent de descripteur :
- **I**dentificateur **G**énéral de **F**ormat du paquet (modulo 8 ou 128 numéros des paquets).
- Identificateur de voie logique.
- Identificateur général de type paquet.

GFI: Identifiant de format général. Q indique un paquet X25 (0) ou X29 (1). D indique un acquittement local (0 : ETCD) ou distant (1 : ETTD). Les bits 01 indiquent que les numéros de trames vont de 0 à 7. Le format de trame où ils indiquent 10 montre que l'on numérote les trames de 0 à 127 (10). Cela permet d'envoyer beaucoup de trame avant d'acquitter ce qui est intéressant pour les réseaux lents tels que les réseaux satellites.

4.3) Différents type de paquets

La zone d'identificateur du type de paquet permet de déterminer la fonction du paquet; elle

ressemble à la zone de supervision de HDLC pour le contrôle de la connexion de réseau. Nous donnons dans le tableau suivant les différents types de paquets que l'on peut rencontrer dans le protocole X.25. Des paquets de diagnostic et d'enregistrement (pour demander l'enregistrement de services complémentaires) complètent la panoplie des paquets décrits précédemment.

Signification du type	Type de paquet
Demande d'ouverture d'un circuit logique	APPEL *(Call request)*
Appel d'ouverture	APPEL ENTRANT*(lncoming call)*
Appel accepté	COMMUNICATION ACCEPTE(*Call accepted)*
Circuit virtuel ouvert	COMMUNICATION ETABLIE*(Call connected)*
Demande de fermeture	DEMANDE DE LIBERATION (*Clear request)*
Confirmation de la fenêtre	Confirmation de Libération *(Clear confirmation)*
Indication de fermeture	INDICATION de Libération *(Clear indication)*
Paquets d'accusé de réception	RR *(Receive Ready)*
Paquets d'accusé de non-réception	RNR *(Receive Not Ready)*
Paquets de rejet	REJ *(Reject)*
Paquets de données	*(Data packet)*
Demande d'interruption	*(lntenupt request)*
Confirmation d'interruption	*(lnterrupt confirmation)*
Demande de réinitialisation	*(Reset request)*
Indication de réinitialisation	*(Reset indication)*
Confirmation de réinitialisation	*(Reset confirmation)*
Demande de reprise	*(Restart request)*
Indication de reprise	*(Restart indication)*
Confirmation de reprise	*(Restart confirmation)*

4.4) Transfert des paquets

a) Circuits Virtuels :

- CVP : Circuit Virtuel Permanent (facturation au volume uniquement) liaison point à point
- CVC : Circuit Virtuel Commuté.

A chaque instant, plusieurs circuits virtuels peuvent cohabiter sur une voie. Chaque circuit virtuel utilise une voie logique qui est repérée par un n° de groupe de voie logique (≤15) et un numéro de voie logique (≤255). Ces n° ont une signification locale à l'ETTD/ETCD.

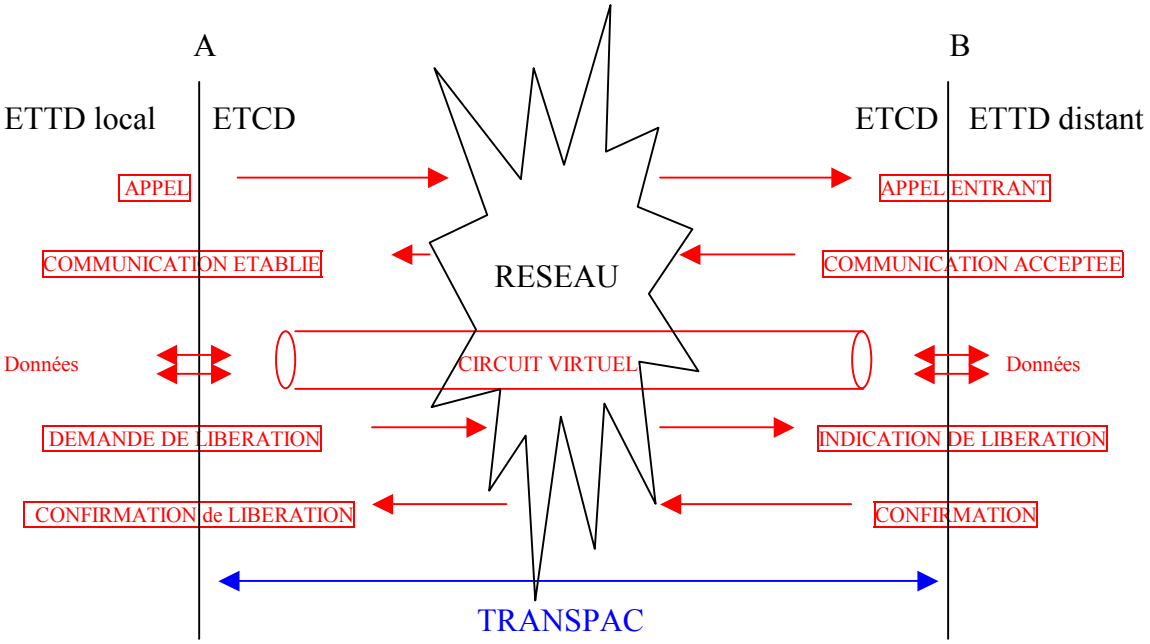
b) Ouverture et fermeture d'un circuit virtuel

Un utilisateur qui veut transmettre des paquets doit, au préalable, ouvrir un circuit virtuel. Pour ce faire, il émet une demande d'ouverture. Le paquet contient le numéro de la voie logique obtenu par l'utilisateur (le plus grand disponible) et l'adresse réseau des abonnés demandé et demandeur. Ces dernières sont inscrites dans un champ de longueur variable dont la longueur est donnée par un champ de quatre bits qui spécifie cette longueur en nombre de demi-octets. La recommandation X. 121 est utilisée; elle normalise l'adresse sur 14 demi-octets. Comme le champ est de 4 bits, il permet d'obtenir une longueur jusqu'à 16 demi-octets.

Le paquet contient un champ pour indiquer les options de contrôle du circuit virtuel ainsi qu'un champ de 64 octets maximum destiné à l'utilisateur. Il peut utiliser ce champ pour donner, entre autres, des adresses complémentaires, si l'adresse du récepteur est un réseau local ou un autocommutateur privé et des mots de passe.

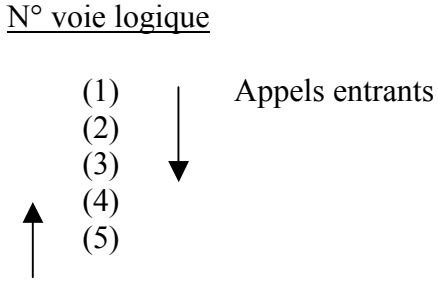
Le paquet d'appel, lorsqu'il arrive à l'ETCD destinataire, capte le plus petit numéro de voie logique pour éviter la collision avec une demande d'ouverture de circuit virtuel qui pourrait arriver sur l'ETCD après avoir réservé le même numéro de voie logique sur l'ETTD, la demande entrante étant alors prioritaire. Le récepteur retourne, s'il accepte la communication, un paquet "communication acceptée", sinon il envoie une demande de libération.

L'émetteur ou le récepteur peut mettre fin au circuit virtuel en envoyant une demande de fermeture qui est acquittée au niveau local. Le paquet de libération peut contenir la raison de la demande : numéro logique à l'ETTD récepteur occupé, émetteur absent ou occupé, paquet non dans l'ordre, erreur locale, congestion d'un nœud, etc. La zone d'identificateur du type de paquet permet de déterminer la fonction du paquet ; elle ressemble à la zone de supervision de HDLC pour le contrôle de la connexion de réseau.



c) Collision d'appel

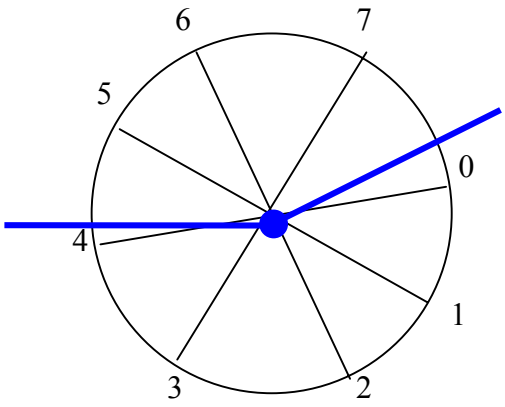
On a collision d'appel lorsqu'il y a une transmission simultanée, sur une voie logique, d'un paquet APPEL ENTRANT et d'un paquet APPEL (sortant). Transpac gère donc l'APPEL et ignore l'autre. Il donne en fait, la priorité à celui ayant le plus petit numéro de voie logique.



(6)
Appels sortants (7)

c) Contrôle de flux des paquets sur un circuit virtuel

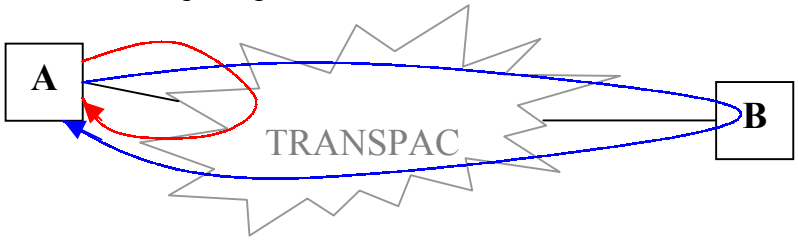
Le contrôle de flux est bidirectionnel avec un mécanisme de fenêtre. C'est une procédure de type HDLC avec deux compteurs de paquets : les paquets émis P(S) et les paquets reçus P(R). Les paquets sont numérotés modulo 8 et on a l'autorisation d'émettre si et seulement si dernier P(R) reçu ≤ P(S) courant ≤ dernier P(R) reçu + W. Ici W correspond à la taille de la fenêtre de niveau 3 (norme ISO modèle OSI).



d) Portée de l'acquittement des paquets de données

Depuis 1984, on a deux types d'acquittement. Cela dépend du choix de sécurité de la liaison.

- local : acquitté par l'ETCD local,
- de bout en bout : acquitté par l'ETCD distant.



e) Taille des paquets

La taille des paquets (champs de données plus en-tête) est fixé lors de la mise en place de l'abonnement à Transpac. On a le choix entre : 16 octets (très interactif), 32, 64 ou 128 octets (pour les gros fichiers).

Si taille du paquet ETTD émetteur > taille du paquet ETTD récepteur, Transpac assure la fragmentation. Dans le cas inverse, si taille du paquet ETTD émetteur < taille du paquet ETTD récepteur, Transpac peut regrouper les paquets (c'est à la demande).

f) Paramètres du circuit virtuel

Pour chaque sens de transmission, un circuit virtuel est caractérisé par :

- une classe de débit (bits/s) qui permet de ralentir le débit,
- les paramètres de contrôle de flux (taille de la fenêtre et taille du paquet de données). A savoir que les tailles de fenêtre de niveaux 2 et3 peuvent être différentes.

Et des services facultatifs suggérés dans la norme X.25 comme :

- des groupes fermés d'abonnées : dispositif du genre « succursales ». Des utilisateurs peuvent former un sous-réseau "prive" ils peuvent communiquer entre eux, mais non à l'extérieur et vice versa, les abonnés n'appartenant pas au groupe fermé ne peuvent pas communiquer avec ceux y appartenant;
- la prise en charge par le demandé. Des appels en PCV peuvent être effectués;
- la sélection des paramètres de contrôle de flux. Les utilisateurs peuvent sélectionner la longueur maximale des paquets, la largeur de la fenêtre ainsi que les classes de débit;
- la possibilité de prendre une fenêtre de 127. Dans ce cas, le champ p(r) et le bit M constituent un octet, le champ p(s) et le bit 0 en constituent un second.

Dans tous les cas, suivant la nature de l'abonnement, des options sont prises par défaut:

- Voie logique unidirectionnelle. Pour éviter des collisions de paquets d'appel, les voies logiques peuvent être unidirectionnelles;
- Temporisateurs pour envoyer les paquets d'interruption, de réinitialisation et de reprise ne sont pas définis. Ils seront déterminés par l'utilisateur, suivant les options prises, et en particulier Si la fenêtre de contrôle est locale ou de bout en bout.

4.5) Conclusion

La recommandation X.25 a été l'une des normes les plus utilisées depuis le début des années 1980. Elle est à la base des grands réseaux de transport de données dans les cinq continents. C'est une norme assez complexe dont le problème principal est la lourdeur. Elle devrait être remplacée petit à petit par les techniques de commutation par cellules. Cette transition sera très longue, d'autant plus que l'optimisation d'une norme demande de nombreuses années. On considère qu'il faut une dizaine d'années avant de parfaitement maîtriser une norme de la complexité de X.25. Les environnements à commutations de cellules ne seront donc pas vraiment opérationnels à un niveau national avant 2005, ce qui laisse encore à X.25 un laps de temps suffisant pour que des utilisateurs investissent dans cette direction, mais il est sûr que les questions de pérennité se posent.

X.25 est maintenant une norme parfaitement maîtrisée et les réseaux associés comme Transpac sont aujourd'hui techniquement irréprochables. Mais la dérégulation massive pousse les différents opérateurs à proposer de nouvelles solutions à base de relayage de trames, de réseaux métropolitains, etc. La suprématie de la norme X.25 pourrait disparaître petit à petit au profit de nouvelles techniques plus performantes.

V. LA NORME X.25 PLP

PLP signifie « Packet Level Protocol ». C'est une adaptation de X25 pour une utilisation dans les réseaux locaux. Standarisé par l'ISO référence 8208 avec le service ne mode connexion OSI.

5.1) Différences avec X.25

A la différence avec le protocole X.25, le routage des paquets qui se fait sans routeur. C'est à dire que le nombre de nœuds intermédiaires est limité. A noter aussi que les ETTD locaux sont appelés EXD (équipement d'échange de données).

a) Fonctions

- Etablissement des circuits virtuels.
- Transfert de paquets de données avec contrôle de flux.
- Gestion des interruptions pour le transfert en mode express (sans contrôle de flux). C'est la possibilité de transférer une petite quantité de données en dehors des procédures normales de contrôle de flux.
- Réinitialisation d'un circuit virtuel (donc au niveau paquet).
- Reprise de tous les circuits virtuels (restart).
- Libération de la connexion en fin de transfert.

b) En émission :

X.25 PLP fournit un service de fragmentation des données du niveau supérieur avant de les transmettre au correspondant. Il fournit aussi un service en mode connecté vers le niveau supérieur (TCP par exemple) . Les Circuits virtuels sont commutés ou permanents.

c) En réception :

X 25 PLP fournit un service de contrôle et de réassemblage des paquets reçus avant de les délivrer (ceci pour le transfert de données sur n'importe quel type de réseau local, quelques soient la taille des PDU véhiculés et le réassemblage des unités de données dans leur taille originelle).

5.2) Similitudes avec X25

X.25 PLP est assimilable à X.25 sur certains points :

- Le format des paquets,
- Le contrôle de flux,
- La fragmentation, et le réassemblage des paquets.

Ex : un numéro TRANSPAC :

1	33	661	889	3615
Pays	Département	Commutateur	Abonné	Service abonné

5.3) Services supplémentaires

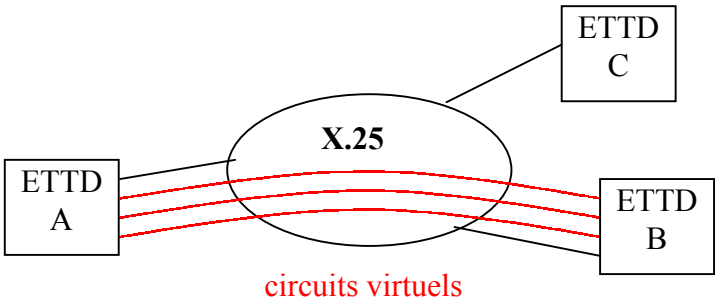
- Paramètres de contrôle de flux :
  - Taille des paquets,
  - Taille des fenêtres.
- Classe des débits,
- Choix et validations du délai de transit,
- Retransmission des paquets de données,
- Gestion de l'extension d'adresse,
- Sélection rapide.

VI. EXERCICE : Circuits Virtuels Multiples

a) Quelle est la différences entre un contrôle de flux de proche en proche et un contrôle de flux de bout en bout ?

La différence, c'est l'acquittement qui se fait en local ou a distance.

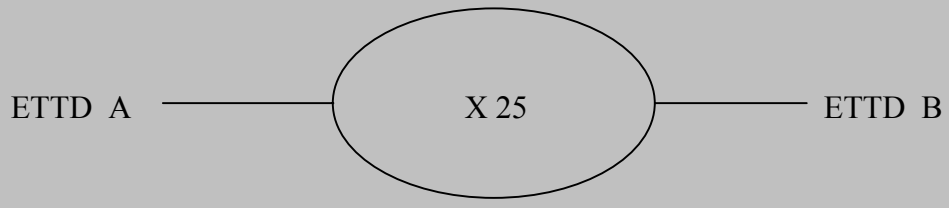
b) Un réseau X25 relie 3 ETTD entre eux, est-il possible d'avoir plusieurs circuits virtuels entre 2 ETTD données ? Si oui, à quoi cela sert-il ?



D'après la norme Transpac, on peut établir plusieurs circuits virtuels selon ce que l'on a définit dans le contrat d'abonnement. Sachant que les appels sortant sont numérotés dans l'ordre décroissant et les appels entrants dans l'ordre croissant, il n'y a pas de problème de circuits virtuels multiples tant qu'ils sont inférieur au nombre autorisé par l'abonnement.

L'intérêt est de pouvoir éclater les paquets sur plusieurs voies donc si l'on a une taille de fenêtre égale à 7 bits, on pourra avoir 7xn voies à la fois avant d'attendre l'accusé de réception ! Ceci entraîne donc l'augmentation de l'anticipation et du débit.

VII. EXERCICE : ECHANGE DE PAQUETS



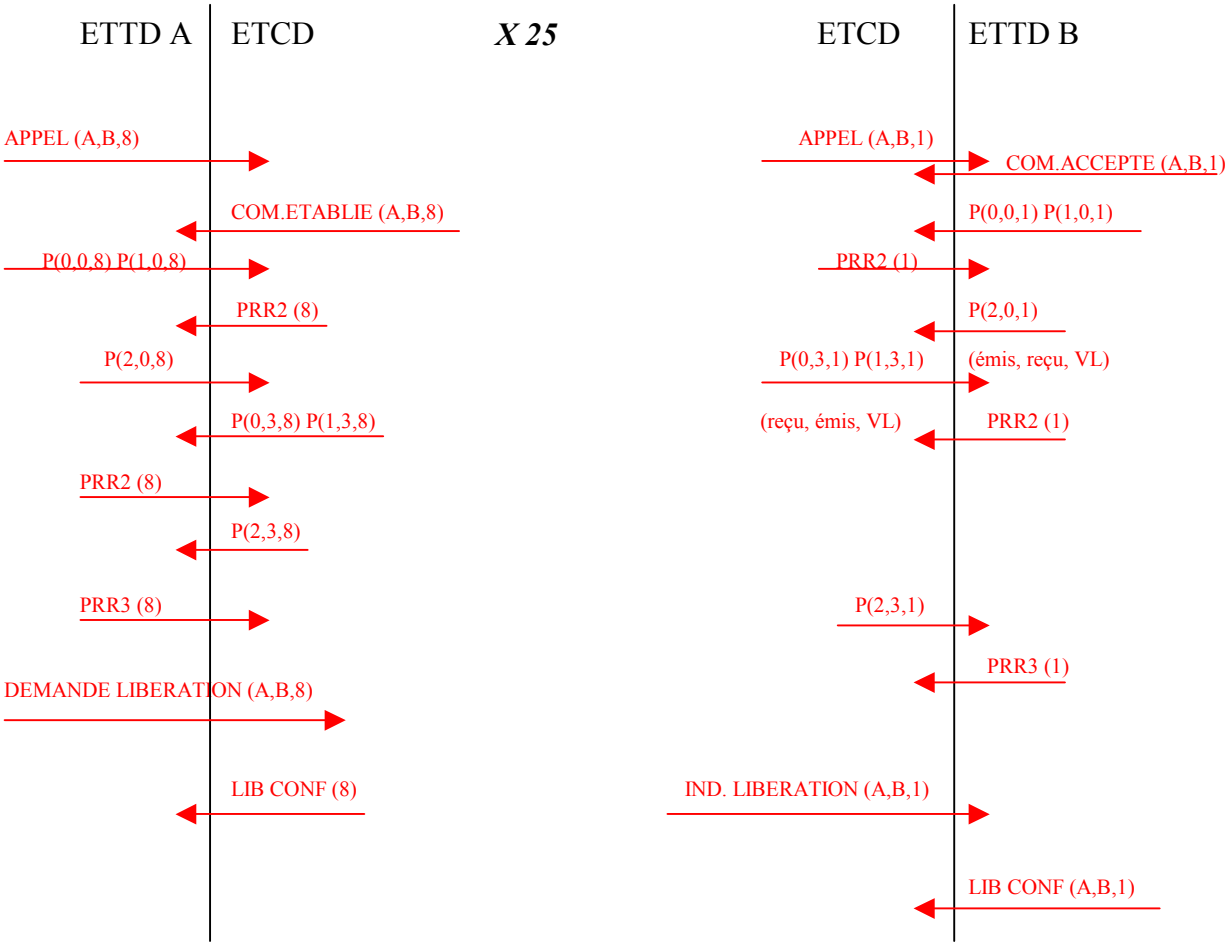
Indiquer quels sont les paquets échangés depuis l'établissement des CVC jusqu'à leur libération. Chaque ETTD veut envoyer 3 paquets de données vers l'autre. C'est A qui ouvre le circuit virtuel commuté. La taille de la fenêtre est de 2.

Paquet de données P(appelant, appelé, voie logique)

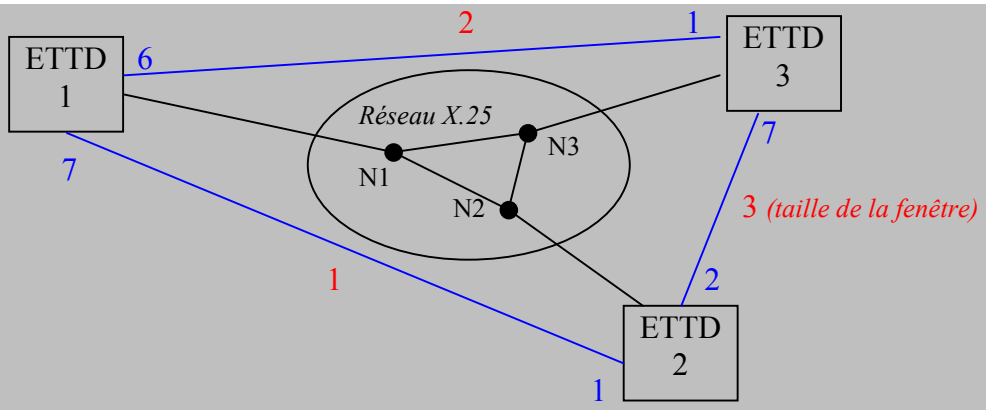
Paquet de supervision APPEL (A,B,8)

PR (VL)

LIB (A,B,VL)



VIII. EXERCICE : Echanges entre ETTD et ETCD



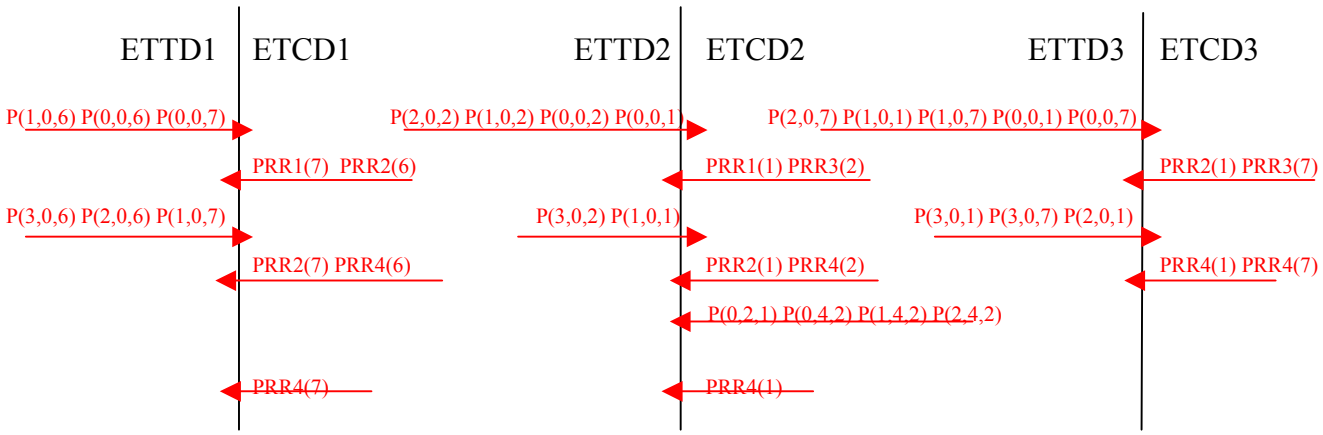
Chaque ETTD veut envoyer 4 paquets de données aux deux autres. Le plus grand numéro de voie logique est 7 pour tous les ETTD.. Les circuits virtuels sont permanents et déjà ouverts C'est 1 qui ouvre le circuit virtuel commuté avec 2 et 3 qui lui ouvre le circuit virtuel commuté avec 2. Les circuits virtuels sont des circuits permanents (CVP). L'acquittement des paquets est local. On entrelace les paquets des différents circuits virtuels. Les fenêtres entre CV sont les suivantes : entre ETTD 1 et ETTD 2 : 1 ; entre ETTD 1 et ETTD 3 : 2 ; entre ETTD 2 et ETTD 3 : 3.

Représenter les paquets échangés depuis l'établissement des CVC jusqu'à leur libération.

On a donc le tableau des circuits virtuels suivant :

Vers \ De	ETTD 1	ETTD 2	ETTD 3
ETTD 1		1	1
ETTD 2	7		7
ETTD 3	6	2	

Avec CV1, CV2, et CV3.



INTERCONNEXION DE RESEAUX

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

INTERCONNEXION DE RESEAUX

En général, des réseaux point à point sur de longues distances (le long des autoroutes, voies ferrées, canaux…) pour arriver à un réseau maillé (ou plutôt arborescent en France).

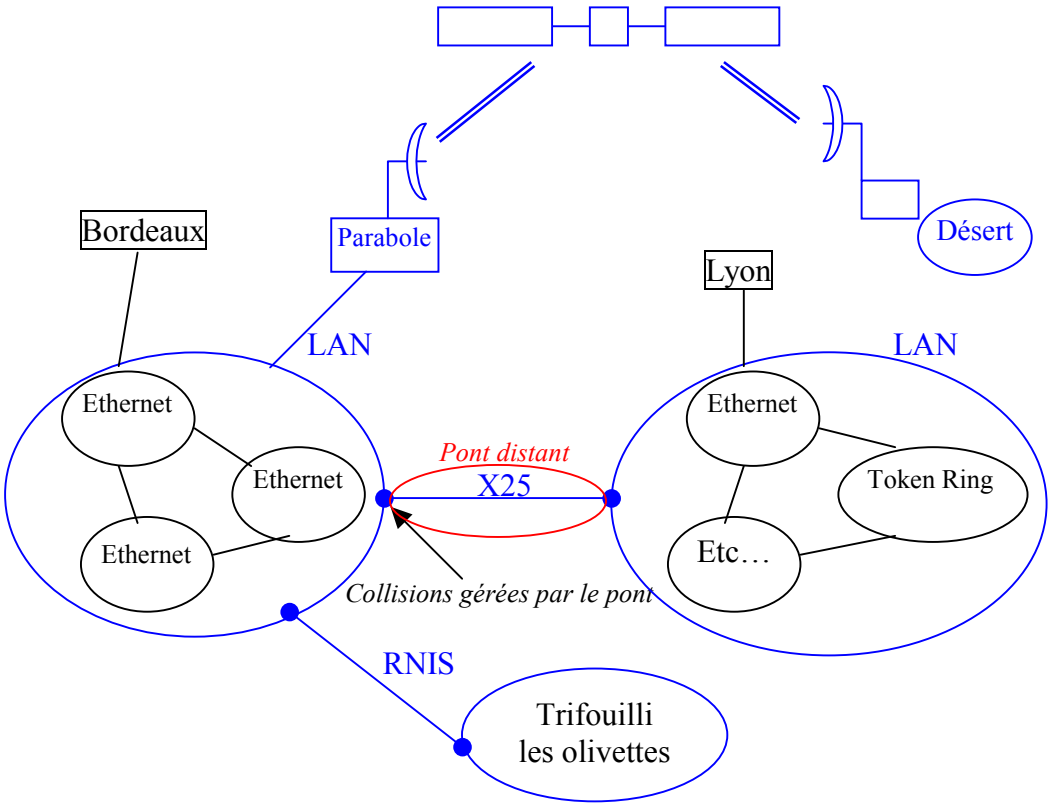
I. BESOINS D'INTERCONNEXION

Les besoins d'interconnexions viennent du besoin de communiquer :

- d'une entreprise sur le même site (LAN),
- dans une entreprise sur des sites différents par interconnexion (messagerie),
- dans une entreprise sur des sites différents avec X25,
- d'une autre entreprise sur un même site.

Intranet : réseau Internet protégé.
Extranet : intranet avec un contrôle précis de la communication vers l'extérieur.

Exemple :

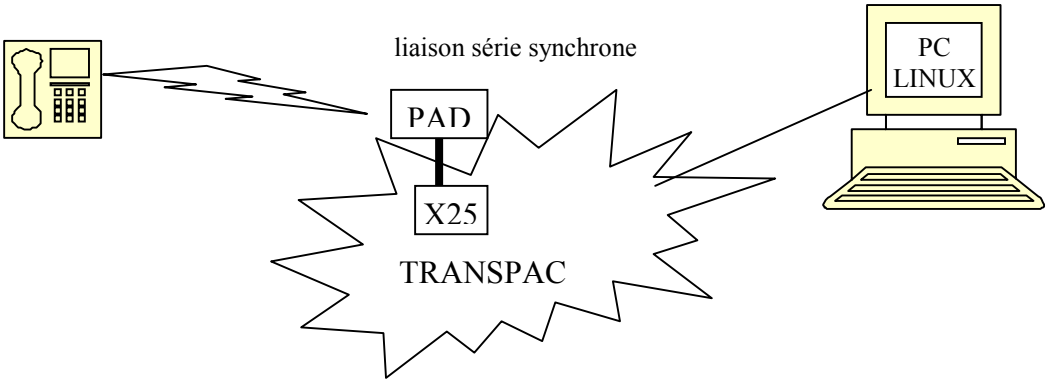


Pour des transmissions à la longue distance, il faut un bon rendement et des petits paquets (pour éviter de réémettre de gros paquets). La qualité de la ligne est exprimé en pourcentage de bits erronés (mauvaise : 10^{-3} : 1 bit/1000 erronés, bon : 10^{-5}).

II. PASSERELLE

Une passerelle est l'ensemble des ressources matérielles et logicielles nécessaires pour offrir aux équipements connectés à des réseaux différents, les moyens de communiquer entre eux.

Ex : Apple Talk vers TCP/IP, ou PAD vers Telnet…



Minitel	PAD	X25	Transpac	X25	Passerelle	Telnet	PC linux

Ici, la passerelle est un logiciel qui accepte un circuit virtuel d'un protocole X25 et qui établit une connexion du type Telnet.

2.1) Les différentes passerelles :

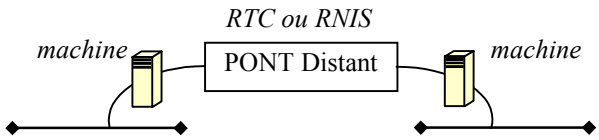
Niveau		
7	Application	Passerelles de niveau supérieur
6	Présentation	Ex : messagerie X400 ⇔ messagerie sendmail
5	Session	
4	Transport	Transcodage ou classes de transport
3	Réseau	Routeur (transmission de paquets)
2	Liaison	Pont (switch =multipont)
1	Physique	Répéteur (, ex : hub= multi répéteur)

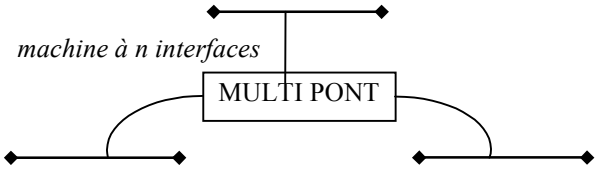
Physique :

remise en forme des signaux

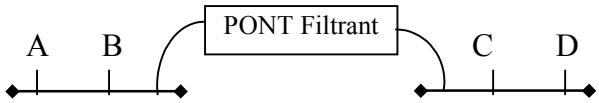
Liaison : *machine à 2 interfaces*

recopie les trames mais ne transmet pas les collisions.

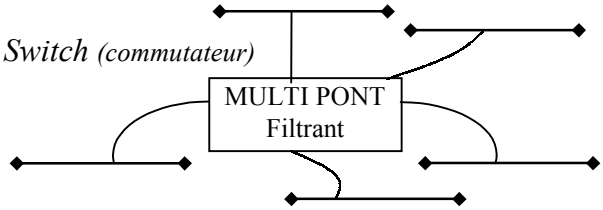




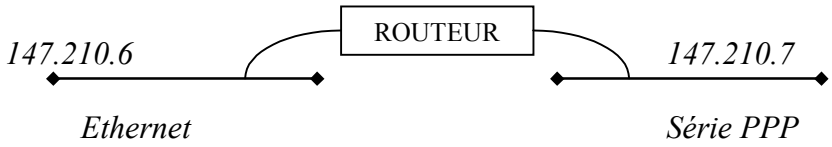
Module le trafic (si a parle à B, C ne le sait pas)



Switch (commutateur)



Réseau : ROUTEUR



Le routeur fait une transmission de paquets en effectuant les conversions nécessaires. On utilise plusieurs protocoles de routage statiques ou dynamiques.

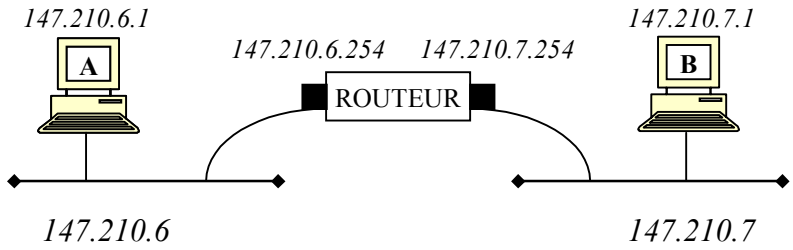
Routage statique : Les routeurs sont des machines qui communiquent entre elles .

RIP : *Routing Information Protocol* (protocole ancien basé sur les fichiers d'adresse de chaque nœud). Ex : daemon Routed d'Unix.

IGRP : *Interior Gateway Routing Protocol* (amélioration de RIP avec délais, envois de paramètres des liaisons, …).

OSPF: *Open Shortest Path First* (permet un routage distribué).

Routage statique :



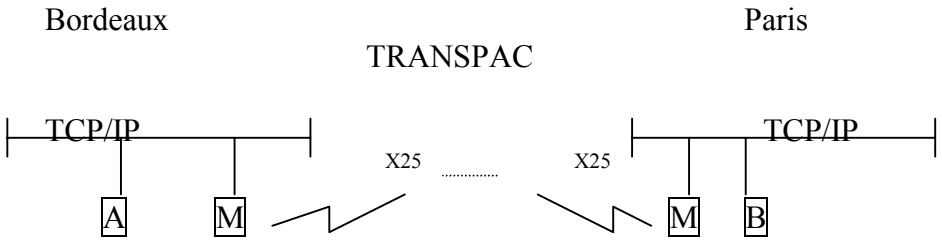
Notons que pour relier 2 réseaux X25 entre eux, il existe la norme X75. INTERNET est en fait une simple interconnexion de réseaux IP.

Transport : transcodage ou adaptation de classes de transport.

Niveaux supérieurs (5, 6, 7) : transfert de fichiers (exemples : PAD - Telnet, X400 - SMTP, Appletalk - Imprimantes TCP/IP…)

2.2) Techniques d'interconnexion

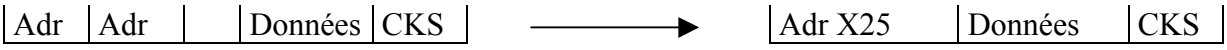
Encapsulation



Cette technique universelle est plus facile et plus simple à mettre en place. Par contre, l'inconvénient c'est que c'est plus lourd et beaucoup moins efficace. En fait, cela consiste à envelopper une trame de R1 comme donnée d'une trame de R2.

Translation ou conversion

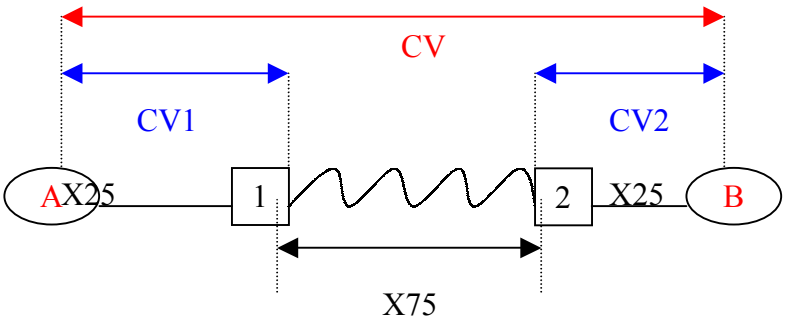
Cette solution est plus efficace mais elle n'est pas universelle.



Transformation en trame X.25

Interconnexion X75

Elle permet l'interconnexion de réseaux X25 (ex : entre la France et l'Espagne).



Principe de fonctionnement :

1) Demande de connexion de la station A et génération du paquet d'appel contenant l'adresse internationale de la station B. Le réseau 1 crée un Circuit Virtuel entre la station et le réseau de sortie (nœud dédié).
2) Etablissement de la liaison entre 1 et 2 (passerelle).

3) Etablissement du Circuit Virtuel entre le nœud 2 et B (il est possible d'avoir 4096 interconnexions différentes et pas de CVP) et transfert des données.

2.3) EXERCICE : Ethernet interconnecté avec X.25

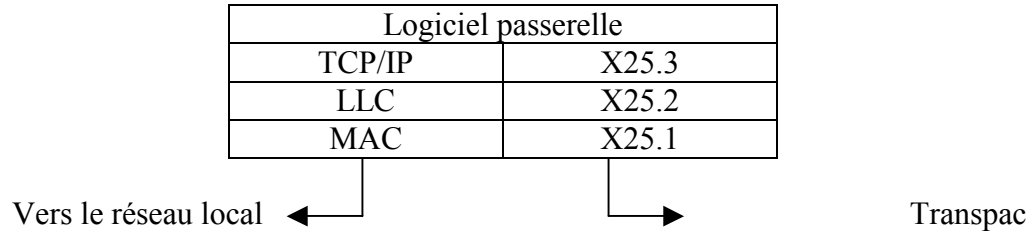
Soit un réseau local d'entreprise de type Ethernet. Toutes les stations veulent communiquer avec le monde extérieur en X.25 (Transpac).

a) Proposer des solutions optimales sans considération de prix

On installe une ligne X.25 à chaque station avec le logiciel correspondant et l'interface (carte). Ceci doit couter dans les 25 000 Frs par an et par poste !!!

b) On veut faire moins cher. Quelle est le rôle de la passerelle ?

On rajoute un routeur vers X25 (passerelle spécialisée) afin de se partager une seule ligne Transpac. Une machine fait office de passerelle (machine générique). Elle établit une passerelle (aussi appelé proxy) vers le Circuit Virtuel avec le X.25.



c) Comparer les échelles de coût.

Il faut prévoir le coût du matériel, le coût des communications, le coût des contrats de maintenance, le coût des ressources humaines…

TCP / IP

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

TCP/IP

Transmission Control Protocol / Internet Protocol
Couche 4 du modèle OSI couche 3

Inclus services tel : telnet, ftp.
TCP défini en fait une famille de protocoles.

I. HISTORIQUE

- 1969 : début d'ARPANET + DOD (Departement Of Defense) → 4 nœuds
  (D) ARPANET : Defense Advanced Research Project Agency Network du departement of defense
- 1972 : 50 sites + 20 commutateurs
  basé sur IMP (ancêtre de X25 et NCP (Network Control Program)).
  Début des spécifications de TCP/IP pour ARPANET
- 1980 : UNIX BSD 4.1 incluant TCP/IP et NCP
  (Berkeley Software Distribution)
- 1983 : TCP remplace NCP dans ARPANET
  ARPANET éclate en MILNET (pour les militaires) et NSFnet (pour les autres)

II. DOCUMENTATION

RFC (Request For Comment)
RFC 793 décrit TCP/IP

RFC est numéroté chronologiquement à partir de 1.
Les documents font entre 1 et 200 pages (moyenne = 20).
Cela peut être la spécification d'un protocole, des informations pour l'utilisateur, des recommandations pour l'organisation des réseaux, des informations pour l'Internet, la spécification de lui même (RFC 1111).
Ce sont des documents publics et gratuits.

2.1) vie d'un RFC

- Sous le contrôle de l'IAB (Internet Activity Board).
- On propose un RFC en envoyant un mail à rfc-editor@isi.edu.
- 5 états dans la vie d'un RFC : niveaux de standardisation

- Proposed Standard Protocol (4 mois minimum à attendre l'avis d'experts).
- Draft Standard Protocol (attente de tests : 6 mois).
- Standard Protocol (devient vraiment un standard).
- Experimental Protocol.
- Information Protocol.

- Documenté dans la RFC 1360.

<table>
<tr><td>7
6
5</td><td colspan="3">Applications
telnet, ftp, smtp</td></tr>
<tr><td>4</td><td>TCP</td><td colspan="2">UDP</td></tr>
<tr><td rowspan="2">3</td><td rowspan="2"></td><td>ICMP</td><td></td></tr>
<tr><td>ARP</td><td>RARP</td></tr>
<tr><td></td><td colspan="3">interface</td></tr>
<tr><td>2</td><td colspan="3">Liaison (ethernet, X25, série…)</td></tr>
<tr><td>1</td><td colspan="3">Physique</td></tr>
</table>

Internet Protocol Suite

III. FONCTIONS DES COUCHES

IP : - Internet Protocol avec RFC 791.
- Elément :datagramme.
- Chaque datagramme comprend l'adresse IP de l'émetteur, l'adresse IP du destinataire et l'indication sur le protocole supérieur (notion d'encapsulation).
- L'interface de niveau 3 :
  - RFC 894 pour ethernet.
  - RFC 877 pour X25.
  - RFC 1188 pour le passage d'IP sur FDDI (FO à 100 Mbits).
  - RFC 1171-1172 pour liaison PPT (Point To Point).

ARP :
- Décrit dans RFC 826.
- Adress Resolution Protocol.
- A une adresse IP, on associe une adresse ethernet (pour les réseaux en bus uniquement).

RARP :
- Décrit dans RFC 903.
- Reverse Adress Resolution Protocol.
- A une adresse ethernet, on associe une adresse IP (BOOTP, RARPD, DHCP).

ICMP :
- Internet Control Message Protocol
- Décrit dans RFC 792.
- Envoi des paquets d'indication pour alerter et contrôler, renvoi un paquet vers l'émetteur.

TCP :
- Transmission Control Protocol
- Décrit dans RFC 793.
- Sur machine uniquement.
- Transport fiable de bout en bout.
- Mode connecté : n° port, origine + destination

UDP :
- User Datagram Protocol
- Décrit dans RFC 768.
- TCP allégé (sans contrôle).
- Mode non connecté, avec un transport non fiable.

IV. ADRESSAGE IP (V4)

- Adresse IP formée de 4 octets
  A.B.C.D
  $0 \leq A,B,C \text{ ou } D \leq 255$ ex : 130.190.5.1

- Chaque adresse devrait être unique au monde car elle est significative d'un lieu géographique.

- Composée d'une adresse réseau + une adresse locale (adresse de machine ou adresse de sous-réseau + adresse machine).

Classe A :
A = adresse de réseau

A < 127 → on utilise 126 réseaux possibles et on garde 254^3 adresses locales possibles (≈ 16 millions).
N'existe pas en France mais aux Etats-Unis (16.0.0.0 : DEC, 18.0.0.0 : MIT)

Classe B :
127 < A < 192

64 * 254 réseaux possibles, 254^2 adresses locales.
Ex : 129.88 : IMAG, 134.157 : Jussieux, 147.210 : Bordeaux I

Classe C :
191 < A < 223

31 * 2542 réseaux possibles (≈ 2 millions), 254 adresses locales possibles

Classe D :
222 < A < 239

Adresses multicast

Classe E :
Classe RFU

Autre méthode : transformer le premier chiffre en binaire sur 8 bits.
Ex :130 → 10000010.
Classe A si commence par 0, B si commence par 1, C si 11, D si 111 et E si 1111.

Loopback : 127.0.0.1
Défaut :0.0.0.0
130.190.0.0 → réseau de classe B 130.190
130.190.255.255 → adresse broadcast (toutes les machines du réseau 130.190)

on associe généralement une adresse IP à un masque de sous réseau (subnetmask).

147.210.9.3
255.255.0.0 netmask

147.210.8.0 8 1000
93.D2.08.00 9 1001
1001 0011 1101 0010 0000 1000 0000 0000

23 bits 9 bits

FF.FF.FE.00
255.255.254.0

Paquet broadcast :

- Envoyé à toutes les machines
- Paramétrable (dans l'interface).
- On met à 1 tous les bits de la partie adresse de la machine.

Adresse broadcast où on réuni 147.210.8.2 et 147.210.9.5 → 147.210.9.255
Le netmask serait 255.255.254.0
L'adresse du réseau serait 147.210.8.0

Exercice

Proposer un mécanisme d'adressage pour faire 4 sous réseaux dans un réseau de classe C
192.33.148.
8 bits 00
01
10
11 11 00 00 00

		netmask	broadcast
4 sous réseaux →	192.33.148.0	255.255.255.192	192.33.148.63
	192.33.148.64		192.33.148.127
	192.33.148.128		
	192.33.148.191		

192.33.148.192 192.33.148.255

4.1) Sigles des organismes qui contrôlent IP

IAB : Internet Activity (Architecture) Board
Groupe de personnes qui gère sur papier Internet. Dirige IETF et IRTF. Se réunissent aux Etats-Unis.

IETF : Internet Engineering Task Force
Etude des problèmes d'exploitation, nombreuses Technical Areas. Composé de volontaires.

IRTF : Internet Research Task Force
Promouvoir la recherche en réseau.

NIC : Network Information Center
Fournir les informations aux utilisateurs (INRIA en France). Chaque NIC à une adresse. Ex de l'Europe : NIC.RIPE.NET.

NOC : Network Operation Center
Exploitation du réseau (terme générique).

FNET : association loi 1901
Fournit des services de messagerie, d'accès à IP à l'Europe, interEUnet (news).

INRIA : Institut National de Recherche en Informatique et Automatisme
Fédérateur Internet. Devenu NIC France.
Attribue de numéros IP et des adresses symboliques. Gestion du DNS.

RENATER : REseau NAtional de la TEchnologie et de la Recherche
Interconnecte des réseaux régionaux. Liaisons internationales (ATM).

RIPE : coopération de réseaux IP européens.

E BONE : Backbone IP Européen.

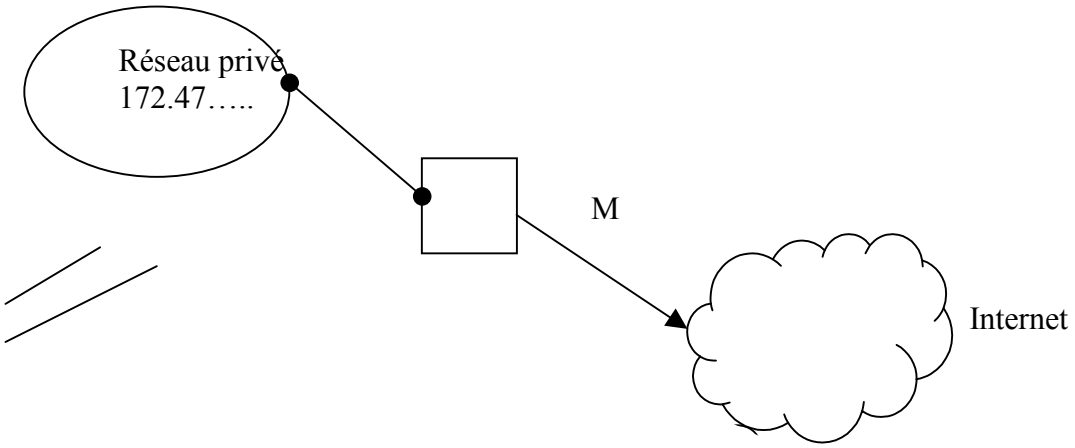
NSF net : National Science Foundation
Le plus gros réseau de l'Internet.
Il y a un backbone à 16 nœuds reliés entre eux à 45 Mb/s.

ISOC : Internet Society
Promouvoir l'utilisation d'Internet et le contrôle.

Beaucoup de documentation sur WWW.UREC.FR (serveur CNRS) ou FTP.UREC.FR

Pour obtenir les RFC : ftp.inria.fr
ftp.ripe.net
nic.ddm.mil (dépositaire officiel) 192.112.36.5

Si on a une adresse privée sur une seule machine



Si les machines du réseau privé veulent discuter avec l'extérieur, elles demandent à la machine ayant l'adresse de sous traiter le trafic. La machine M contient une table de correspondance. Cette machine s'appelle un proxie (lien entre réseau privé et réseau public).

4.2) Entête d'un paquet IP

8 bits		1 octet	2 octets
Version	HIL	TOS	TL
Id (2 bits)		FLG (3 bits)	FO
TTL	Protocol	Header checksum	
Adresse source			
Adresse destination			
Options		Padding	

L'entête ne fait pas moins de 20 octets.
Actuellement nous sommes à IP V4.

HIL : Internal Header Length (unité = 4). Vaut souvent 5.
TOS : Type Of Service (qualité de service).
TL : Total Length (longueur totale du datagramme).
ID : Identification faite par l'émetteur. Sert uniquement pour la fragmentation.
FLG : Flags pour la fragmentation
001 : il y a d'autres fragments
000 : dernier fragment
01- : pas de fragmentation
FO : Fragment Offset (unité = 8 octets)
Le premier fragment = 0
TTL : Time To Live (de 1 à 255)
Décrémenté de 1 à chaque franchissement de routeur. Quand il est à zéro, le paquet est détruit.
Protocol : Identifie le protocol de la couche 4
6 → TCP
17 → UDP
1 → ICMP
Header Checksum : code de redondance dépendant uniquement de l'entête.

@ source
@ destination
} adresse IP

Options : variable en taille, pour contrôle de flux par exemple.

V. TCP

Transmission Protocol (RFC793)

IP

TCP = données d'un paquet IP

5.1) Service de transport

- De bout en bout, c'est à dire de l'émetteur au destinataire
- Il est en mode connecté avec ouverture + fermeture.
- Système d'acquittement.
- Pas de pertes : numérotation des paquets + retransmission si pas tout reçu.
- Ordonné.
- Sans erreur (checksum).
- Contrôle de flux (avec une taille de fenêtre).
- Full duplex.
- Indication de service (n° port).
- Taille minimale de l'entête : 20 octets.

5.2) Entête de TCP

<table>
<tr><td colspan="3">Port source (16 bits)</td><td colspan="3">Port destination (16 bits)</td></tr>
<tr><td colspan="6">Sequence number (4 octets)</td></tr>
<tr><td colspan="6">Ack number</td></tr>
<tr><td>Data offset
(4bits)</td><td>Réservé
(6 bits)</td><td colspan="3">U A P R S F
R C S S Y I
G K H T N N
(6 bits)</td><td>Window
(16 bits)</td></tr>
<tr><td colspan="3">Checksum</td><td colspan="3">Pointeur urgent</td></tr>
<tr><td colspan="6">Options</td></tr>
</table>

Source port
Destination port
} Numéro de service

Sequence number : n° du 1er octet de données dans le segment. Modulo 2^{32}.

Acknowledgement number : numéro du prochain octet que l'on souhaite recevoir. Paquets précédents reçus sans erreurs ni pertes.

Data offset : Longueur de l'entête (unité = 4 octets). Valeur habituelle = 5.

URG : Données urgentes. Le champ « pointeur vers urgence » indique alors la donnée à transmettre immédiatement.

ACK : indique qu'il faut tenir compte de l'acknowledgement number.

PSH : Push. Délivrer les données immédiatement à l'application (vider le buffer, return…).

RST : Reset. Reprendre la connexion au départ.

SYN : Demande d'ouverture.
Ex : demande d'ouverture d'une session TCP par A vers B.
A→B sequence number = 100 ; SYN
B→A sequence number = 300 ; Ack number = 101 ; Ack ; SYN
A→B sequence number = 101 ; Ack number = 301 ; Ack
A→B sequence number = 101 ; Ack number = 301 ; données

FIN : Termine une connexion.

Window : Taille de la fenêtre. Nombre d'octets que l'on peut recevoir par rapport à l'acknowledgement number. (fonction du contrôle de flux).

Cheksum : Fonction des entêtes TCP, IP + données. Assure un transport sans erreurs.

Options : - Fin de la liste d'options (type = 0).
- Pas d'opération (type = 1).
- Taille maximum des segments acceptés.
  Type = 2. Taille = 4 – taille maxi du segment.

5.3) Ajustement des délais de transmission (contrôle de flux).

- Se fait automatiquement et dynamiquement au cours de la connexion.
- Fonction des délais d'acquittement
  ROUND TRIP TIME (RTT) : temps entre l'envoi d'un segment et la réception de l'acquittement.
  Lissage des RTT → SRTT (Smoothed Round Trip Time).
  Retransmission Time Out (RTO).
- Tout cela permet à TCP de s'adapter à la vitesse du réseau et des délais de retransmission.

VI. UDP

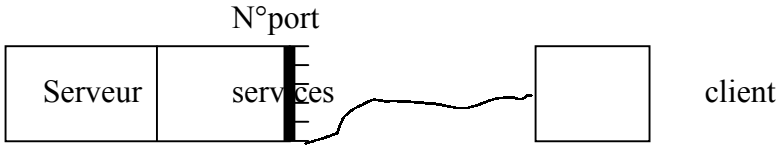
Port source	Port destination
Longueur	Checksum

VII. NUMEROS DE PORT

20 : FTP data
21 : FTP control
23 : Telnet
25 : SMTP (Simple Mail Transfert Protocol)
110 : POP3
111 : SUNRPC (Remote Procedure Call)
520 : UDP Router (routage)

7.1) Sockets

Combinaison adresse IP + numéro de port.



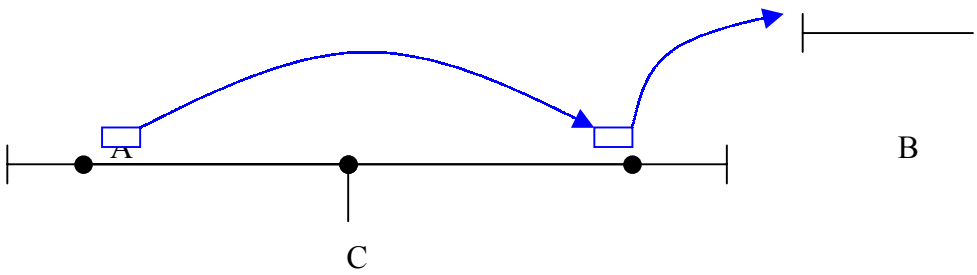
7.2) Routage

Un routeur reçoit un paquet, que doit-il en faire ? ?
Dans chaque paquet : @ IP de destination

- Idée :

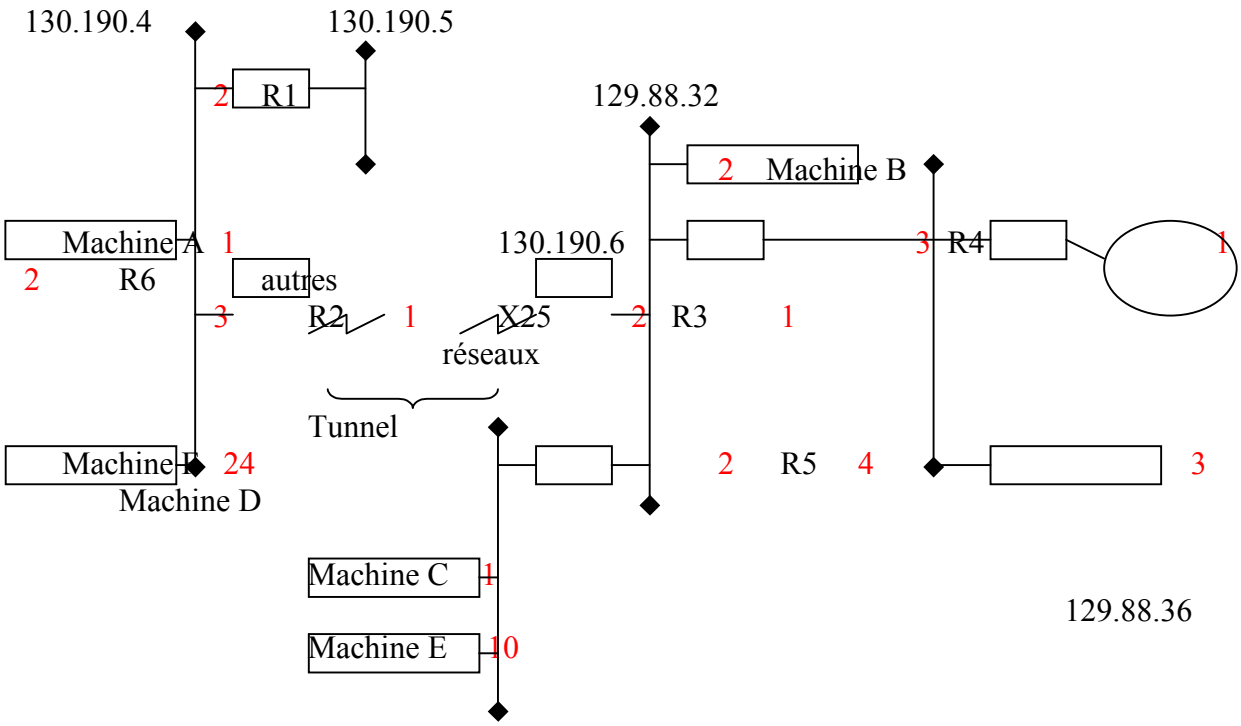
- Chaque routeur connaît uniquement son voisin.
- La carte complète du réseau n'existe pas.
- Table de routage sur chaque routeur.
  La mise à jour de la table de routage peut être :
  - statique : manuelle, il faut remplir à la main les lignes de configuration.
  - Dynamique : des démons échangent des infos de routage.
  - Mixte

Routage avec Ethernet (table ARP)



Un routeur peut recevoir sur son interface ETHERNET des trames ethernet qui lui sont destinées mais avec un paquet IP qui n'est pas pour lui. Il le renvoi alors vers le routeur le plus proche ou vers un autre tronçon.

Exemple de routage



B : if config eth0 129.88.32.2
Netmask 255.255.255.0 } config de B

Pour aller sur A { Route add net 130.190.0.0 129.88.32.1 1

pour aller ici — il faut passer par là — nb de routeurs à franchir

Pour aller sur E { Route add net 193.33.64.0 129.88.32.4 1

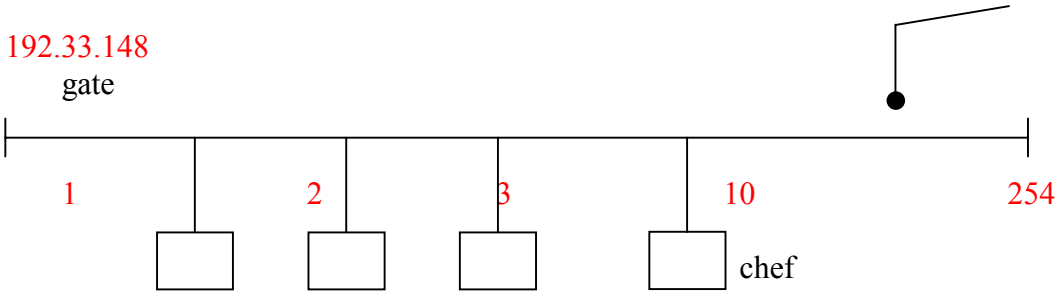
Pour aller sur D { Route add net 129.88.36.0 129.88.32.3 1

Route add default
Remplacé par

Exercice

L'administrateur d'un parc de machine LINUX voudrait contrôler les accès réseau vers l'extérieur.

1) Couper le réseau automatiquement pendant certaines plages horaires ou en fonction de certains critères. Proposer une solution.
   a) sans rajouter de matériel
   b) écrire les table de routage sur les machines



1.a) le chef devient routeur des postes

1.b) sur les postes : route add default 192.33.148.10
sur le chef : route add default 192.33.148.254

Pour empêcher le routage, le chef refuse les requêtes qui ne sont pas pour lui (ou bien il peut aussi déconnecter son poste ! !).

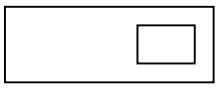
Routage des paquets IP

Problématique : Dans chaque paquet IP indépendamment les uns des autres, on a les informations, y figurent deux adresses « mondiales » : émetteur et récepteur.

Fonctionnement d'un routeur

1) Le routeur déballe le paquet (ethernet ou autre, X25...) pour récupérer l'adresse IP ⇔ ouvrir « l'enveloppe » pour récupérer le paquet IP.
2) Lire les adresses.
3) Refaire une « enveloppe » pour réinjecter le paquet sur un autre réseau.

1) Ether

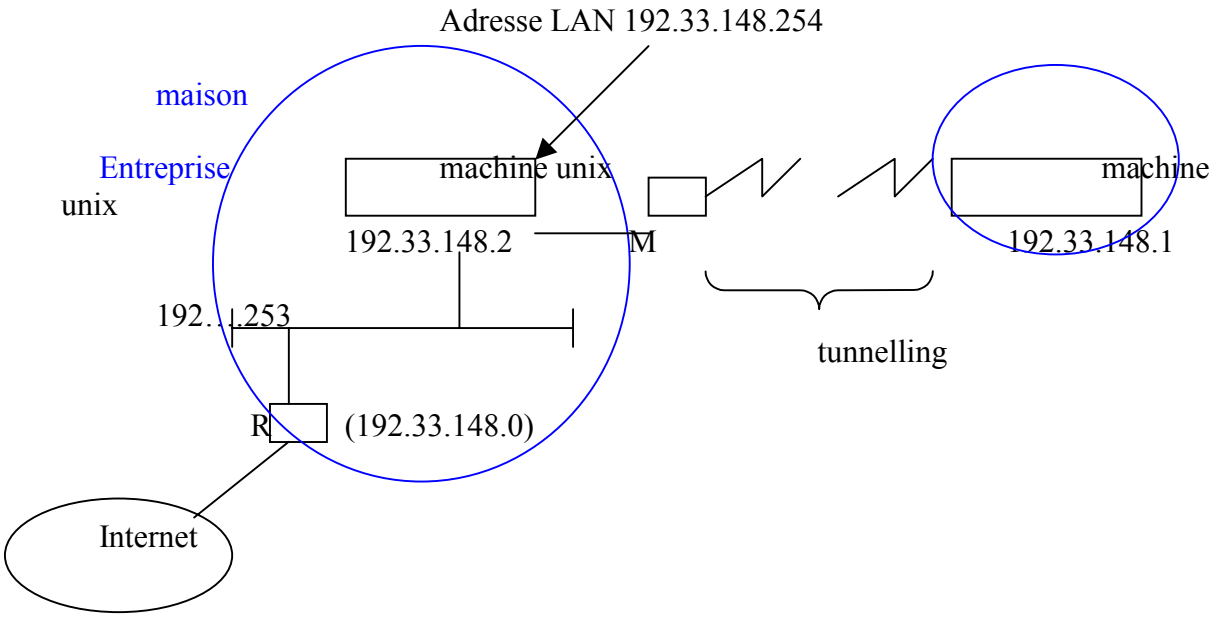


IP

2) Regarde entête IP du paquet IP

3) (routage host ou net)
   a) Trouver l'adresse MAC du destinataire ou bien l'intermédiaire le connaissant. Si l'adresse n'est pas dans la table ARP (locale à la machine et construite dynamiquement) il faut alors envoyer un paquet broadcast à toutes les machines, et celle qui se reconnaît dans le message répond en envoyant son adresse ethernet (MAC).
   (commande rup sur Unix pour toutes les machine distantes sur le réseau).
   b) Refaire une trame ethernet et l'injecter sur le support

Exercice



Dans la table ARP de 192.33.148.2, on met l'adresse de la machine maison 192.33.148.1

Table ARP

192.33.148.1	0008.7b54.3b0c	P
192.33.148.2	0008.7b54.3b0c	
192.33.148.3	0008.7ba2.3b7f	

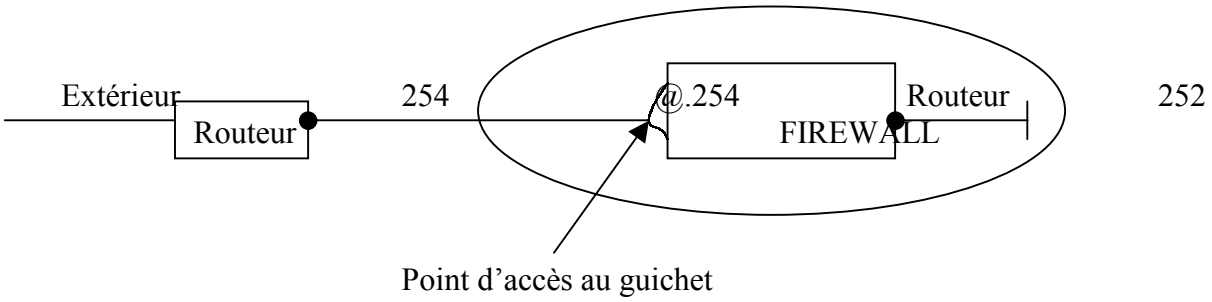
Seul 2 connaît 1 et peut lui faire parvenir une info que lui enverrait Internet via un routeur.

P : manuellement, il faut préciser que cette liaison est permanente.

→ if config PPP∅ 192.33.148.254
→ route add 192.33.148.1 192.33.148.254 1

PPP∅ : point d'attache entre machine Unix 2 et son modem. Saut

Routeur FIREWALL (pare feu, garde barrière)



Le firewall filtre les paquets pour éviter ce qui pourrait polluer son réseau.

Point d'accès : seul point par lequel l'extérieur peut entrer dans le réseau. En fait, on travaille sur les entêtes des paquets et les numéros de services.

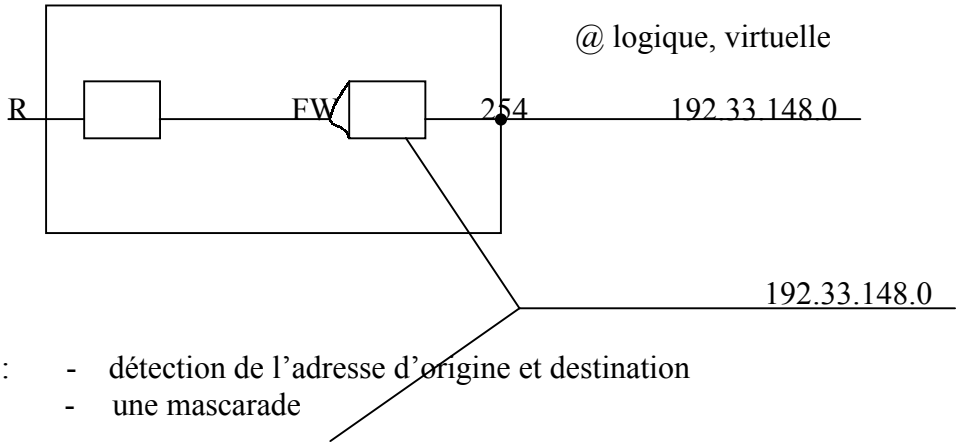
Il faut en plus protéger le firewall par un routeur.
Ce n'est qu'une parade car un routeur est facile à faire tomber, son buffer est de 64 Ko, donc facilement débordable et les paquets passent. Normalement, le paquet est stocké dans le buffer avant de le renvoyer mais quand le buffer est over, le routeur distribue le paquet sans le contrôler.

Si un firewall est en panne, on l'enlève, tout le monde va vers 252, si on l'enlève : problème, personne ne saura aller vers le routeur 254. (car seul le firewall avait cette adresse).

EXERCICE

1) Etudier un mécanisme de FW transparent. C'est à dire qui peut être ajouté ou enlevé de façon transparente.
   a) Peut-on le faire avec des mécanismes ARP + routage ?
   b) Faut-il modifier ou adapter les protocoles.

   NB : Le logiciel de FW le plus connu est le TIS.



manip : - détection de l'adresse d'origine et destination
- une mascarade

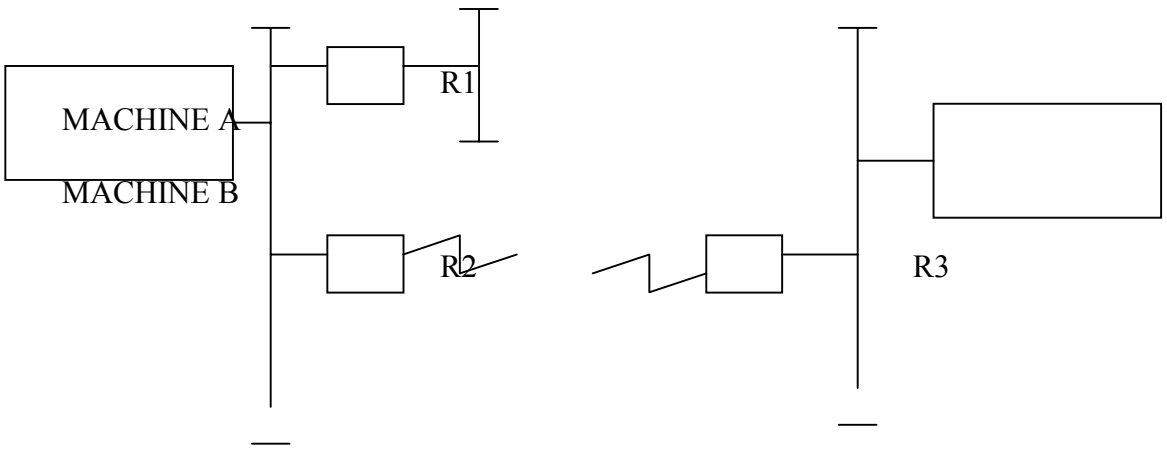
→ restart – 1 → table de routage

Fw peut être : un routeur ou une machine LINUX (redhat + samba ⇒ le + fiable pour PME). Le FW est transparent quand il a l'adresse de l'expéditeur.

Routage statique

ICMP redirect

Indication d'un routeur qui indique qu'il y a eu une erreur et qui permet de renvoyer le paquet. Permet de corriger dynamiquement une table de routage.



Si A est mal configurée, elle envoie un paquet à destination de B vers R1 (au lieu de R2). R1 l'envoie à R2 et ensuite R2 envoie à R3 qui l'envoie vers B.

R1 envoie un paquet ICMP redirect à A en disant que pour atteindre B, il faut passer par R2.

A ajoute cette info dans sa table de routage s'il supporte ICMP redirect.

Problème du routage statique :
- MAJ manuelle.
- Sert surtout pour des postes finaux.

Routage dynamique

Principe : régulièrement, les routeurs diffusent des informations pour indiquer quels réseaux on peut atteindre par eux. Les machines-routeurs enregistrent ces informations et font la MAJ des tables de routage.

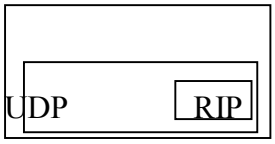
Il existe plusieurs protocoles pour échanger les informations de routage. On met alors les machines en domaines.
Autonomous System (AS) : ensemble des machines qui ont la même politique de routage. (n° à 16 bits, ex : CEA : 777, RNI renater : 1717)

Dans un AS, 2 types de protocoles :
- interne (dans un AS) : RIP, OSPF, IGRP (propriétaire CISCO).
- Externe (entre les AS) : EGP, BGP.

RIP : Routing Information Protocol (RFC 1058)
(à l'intérieur d'un AS)
mis en œuvre par daemon routed ou gated (sous Unix)

ethernet

IP



Le routeur diffuse la liste des réseaux qu'il peut atteindre avec une certaine distance (nb de sauts). Les machines écoutent sur le port UDP 520. Les routages peuvent être actifs (émet, reçoit) ou passifs (écoute seulement).

R2 envoi un paquet broadcast sur le réseau 130.190.4 qui dit

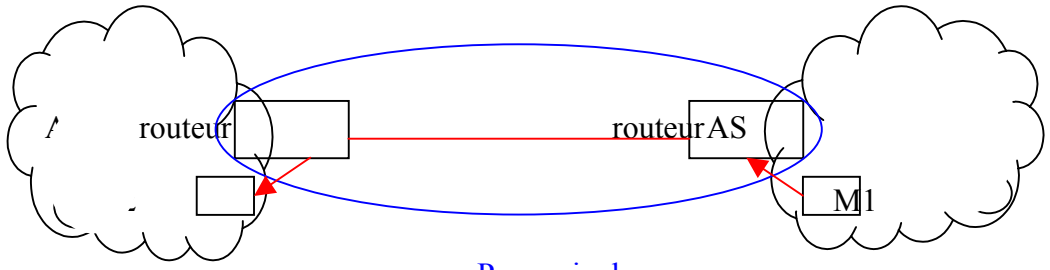
« je peux atteindre 130.190.6 avec une distance d = 1
129.198.2 avec d = 2
193.33.64 avec d = 3
tout le reste avec d = 4 »

Avantages du RIP :
- Très commun et très utilise (surtout au états unis).
- Permet de s'adapter automatiquement en cas de panne.

Inconvénients du RIP :
- Distance donnée comme seule info (pas le coût, ni la vitesse, la charge, le débit, ..)
- Distance max = 15. (16 = inaccessible).
- Pas de garantie sur l'origine des informations.

EGP :

Protocole phare d'AS en AS
Exterior Gateway Protocol (RFC 904).
Protocole d'échange d'information de routage entre 2 AS.
Dans IP, protocol = 8.
Utilisé à l'origine dans ARPANET.
Utilisé entre 2 routeurs aux frontières des AS.

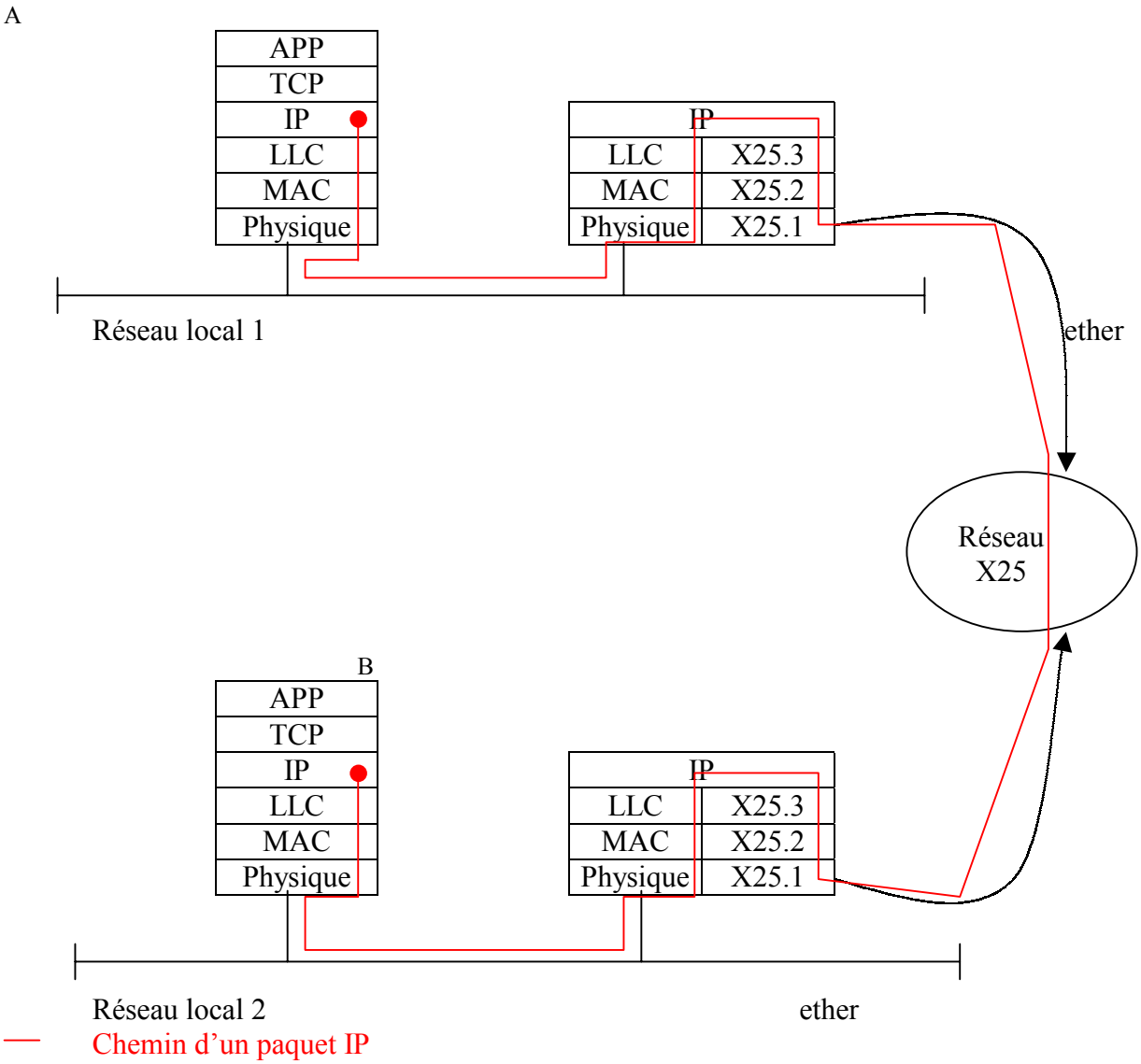


Pear neiggbour
(liaison point à point)

Messages échangés :
- « es tu là ? »
- « oui je suis là »
- « envoies moi ta table de routage »
- « voici ma table de routage »

Accessible en lecture seule.
Pas d'information sur les sous réseaux.
Si il y une boucle → catastrophe.
Seule protocole correctement implémenté sur les routeurs actuellement (utilisé par RE NATER).

EXERCICE



Le datagramme émis par A contient 1300 octets de données. X25 n'accepte que des paquets de 64 octets de données.

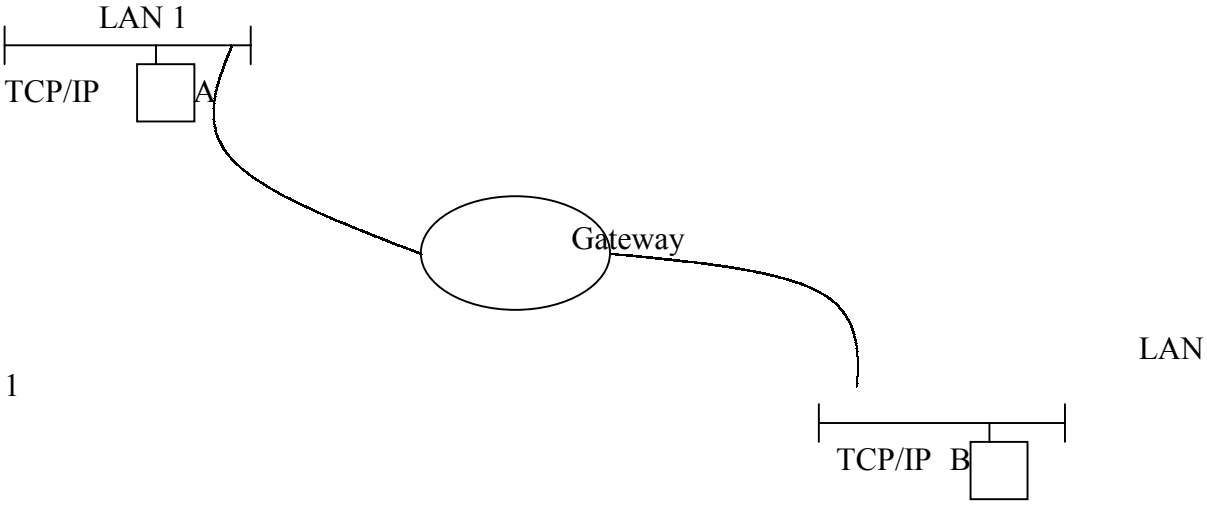
1) Décrire la suite des paquets transmis.
2) Calculer et comparer le volume.

- ouvrir le circuit virtuel : 2 paquets (appels + confirmation appel)
- 1300 / 64 ≈ 20,3 → 21 paquets Transpac
- fermer le circuit virtuel avec libération et confirmation

⇒ 25 paquets de (64 octets + 3 octets d'en-tête)
⇒ 25*67 = 1675

paquet IP → 1300 + 20 octets d'en-tête = 1320
1675 – 1320 = 355
Erreur != 26,9% d' "overhead "

EXERCICE



1) expliquer ce qui se passe au niveau des échanges entre STA et STB
2) proposer des passerelles pour : a) les 2 réseaux sur le même site
   b) les 2 réseaux sont dans une même ville
   c) les 2 réseaux sont dans 2 villes différentes
   d) les 2 réseaux sont sur 2 continents différents

2a) même pièce : répéteur
2b) liaison spécialisée
2c) réseau transport du pays (ex :Transpac)
2d) passerelle X75 qui interconnecte 2 réseaux X25

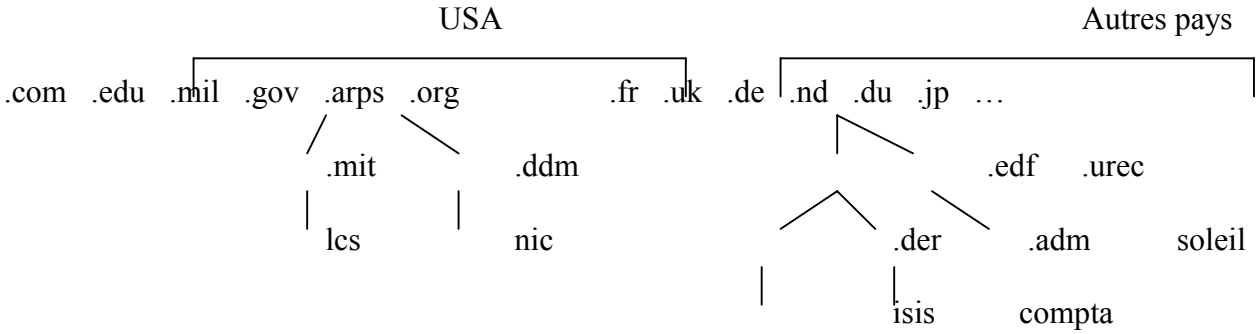
7.3) Le Nommage

- Service qui établit la correspondance entre un nom symbolique et une adresse IP
  - Les machines communiquent avec des à IP
  - Les applications et les utilisateurs utilisent les noms
  - Un nom doit être unique
  - Une adresse peut avoir plusieurs noms

- RFC 1032 et RFC 1033

- L'espace des noms est domainisé.
  Ex : der.edf.fr (direction étude recherche appartient à EDF en France). Isis.der.edf.fr (machine isis de der de EDF de France)

- Le système est arborescent. Le monde est découpé en domaines (ex :fr) au-dessus, il y a ce qu'on appelle la racine(virtuelle).



- L'administration des noms est hiérarchisée.
- Un seul organisme (actuellement Nic France) est responsable des noms x.fr (il enregistre tous les noms de domaines sous .fr avec un gérant pour chaque domaine. Pour obtenir un nom de domaine sous .fr, il faut contacter Annie Renard à l'inria : Annie.Renard@inria.fr)
- Le gérant du domaine x.fr est responsable des noms de la forme y.x.fr (ex :der.edf.fr)
- Si y est un domaine, on réitère.
- Cette gestion inclut les noms de machine.
- Tous les noms sont ainsi uniques.

Différences entre @ IP et nom de domaine

- Il n'y a pas de correspondance systématique entre un nom de domaine et une adresse de réseau IP (théoriquement aucun lien)
- Le nom est une notion administrative (les domaines ont tendance à reproduire l'organigramme d'un organisme ou d'une société).

- L'adresse IP doit tenir compte de la structure du réseau donc de la géographie.

Qui fait la conversion entre @ et nom ?

- etc/hosts (sous Unix)
  table ASCII, mise à jour manuellement, limitée
  → gestion difficile
  → tend à disparaître
- DNS
  Système de serveur de noms : global, utilisé dans l'Internet, hiérarchique, mise à jour facile et décentralisée.
  → gestion facile
  → remplace la table hosts

DNS (Domain Name System) RFC 1034 et 1035

- Basé sur le mode client serveur
- Connexion UDP ou TCP (port 53 pour le serveur)

(sous Unix, etc/resolv.conf)
Très simple à mettre en œuvre, pas de mise à jour à faire sur cette machine.

Le serveur : le démon « named » répond aux requêtes

Le serveur d'un domaine connaît :
- les @ et Nom de domaine local.
- les @ des serveurs de domaines inférieurs.
- les @ des serveurs racines.

Les serveurs racines connaissent les @ des serveurs de tous les top-level domain.

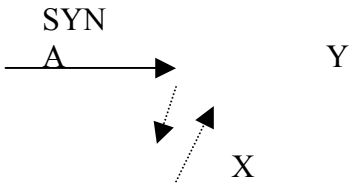
Redondance : plusieurs serveurs possèdent dans leur mémoire ou dans des fichiers les même informations.
Un serveur primaire contient la base d'information d'un domaine.
Cette base est mise à jour manuellement.
Des serveurs secondaires ont une copie avec MAJ automatique de la base d'information du serveur primaire.
→ il faut choisir correctement son serveur primaire et ses serveurs secondaires.

La commande NSLOOKUP sous Unix permet d'interroger un DNS.

7.4) Sécurité des réseaux

- Sécurité matérielle contre :

- Les vols de connexions par « sniffeur » (de paquets hors protocoles).
- Les refus de services



A envoi une demande de connexion (SYN) à Y en se faisant passer pour X; comme X ne répond jamais à Y, Y attend jusqu'au time out pour se libérer et permettre une autre connexion. Alors A recommence.

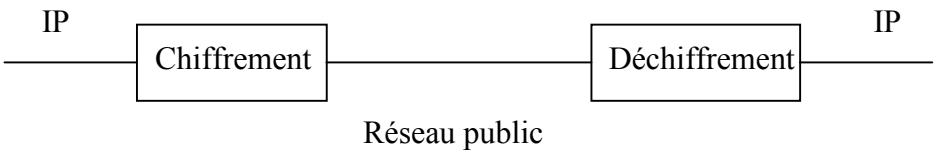
- Les scans systématiques de tous les ports d'une machine pour savoir lequel est associé à un service (peut se détecter avec un journaliseur d'événement sur la machine : log).

- Sécurité logicielle :

- Les antivirus. (site WWW.UREC.FR pour savoir comment faire un virus).

- Sécurité au niveau des couches 2 et 3 :

Pour éviter que l'on puisse se connecter aux câbles, on chiffre les données (cryptage). Le cryptage se fait par l'intermédiaire de boites de chiffrement.



Les boites à chiffrement sont soumises à autorisation.

Le problème qui se pose le plus souvent c'est l'authentification de l'utilisateur (par calculette (synchronisation avec un matériel), fond de l'œil, empreintes digitales, empreinte vocale).

COUCHE TRANSPORT

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LE NIVEAU TRANSPORT DU MODELE OSI

I. DEFINITIONS

Rôle de la couche Transport :

Son rôle peut être défini assez formellement sous forme des quatre propriétés suivantes :

- Transport de bout en bout,
- Sélection d'une qualité de service,
- Transparence,
- Adressage.

Cette couche garantie la bonne livraison des messages, sans erreurs, dans l'ordre et sans pertes ni doublons. Cette couche reconditionne les messages pour en assurer une transmission efficace sur le réseau. Coté réception, la couche transport désencapsule les messages, rassemble les messages d'origine et émet un accusé de réception.

- S'assure que les paquets sont reçus sans erreurs, dans l'ordre, sans perte ni duplication,
- Découpe en paquet et réassemble.
- Envoi d'un accuse de réception,
- Contrôle le flux et gestion des erreurs.

La couche transport offre des services supplémentaires par rapport à la couche réseau. Cette couche garantie que les données reçues sont telles qu'elles ont été envoyées. Pour vérifier l'intégrité des données, cette couche se sert des mécanismes de contrôle des couches inférieurs. Cette couche transport est aussi responsable de la création de plusieurs connexions logiques par multiplexage sur la même connexion réseau. Le multiplexage se produit quand plusieurs connexions logiques partagent la même connexion physique. La couche transport se trouve au milieu du modèle OSI. Les trois couches inférieures forment le sous-réseau, les trois couches supérieures sont implémentées par les logiciels réseau.

La couche transport est aussi implémentée sur les nœuds. Son travail consiste à relier un sous-réseau non fiable à un réseau plus fiable. Dans les réseaux TCP/IP, la fonction de la couche transport est assurée par le protocole TCP et par le protocole UDP. La couche transport implémente le multiplexage dans lequel plusieurs éléments logiciels partagent la même adresse de la couche réseau. Pour identifier sans erreur l'élément logiciel dans la couche transport, un forme plus spécifique d'adresse est nécessaire. Ces adresses, appelées adresses de transport, sont fournies par une combinaison de l'adresse de la couche réseau et d'un numéro TSAP (Transport Service Access Point). Dans les réseaux TCP/IP, l'adresse de transport porte le nom de numéro de port.

II. SERVICE DE RESEAU

*La fonction de la couche transport est de rendre invisible les couches inférieures du modèle OSI. Elle est définie par le type de liaison (connecté ou non), le QoS (*Quality of Service*) et des primitives.*

Services	***Mode connecté***	***Non connecté***
Délai d'établissement + probabilité d'échec	*X*	
Délai de libération + probabilité d'échec	*X*	
Débit + temps de transit	*X*	*X*
*Taux d'erreur résiduelles (*de 10^{-10} à 10^{-12}*)*	*X*	*X*
Probabilité d'incidents de transfert	*X*	
Probabilité de rupture de connexion réseau	*X*	
Protection des connexion de réseau	*X*	*X*
*Coût maximal acceptable (*transpac : accès international)	*X*	*X*
Priorités de routage		*X*

III. SERVICE DE TRANSPORT

D'une manière générale ce sont les mêmes services que la couche réseau (Mode connecté ou non, primitives, QoS…) mais avec des taux d'erreurs beaucoup plus faible.

IV. FONCTIONS de la COUCHE TRANSPORT

Que ce soit pour TCP/IP comme pour la couche du modèle OSI.

- *Transfert de données normales et express.*
- *Segmentation, concaténation et groupage.*
- *Libération inconditionnelle : évite de devoir débrancher la prise !*
- *Etablissement de connexion de transport et traduction adresse de transport par rapport à l'adresse de réseaux plus différent contrôle au niveau de l'établissement…*
- *Multiplexage et éclatement. Le multiplexage permet de minimiser les coûts en utilisant plusieurs connexions transport pour une connexion réseau (ex : http + ftp + mail en même temps). Pour l'éclatement, on utilise plusieurs liaisons réseaux pour une liaison transport pour accélérer le débit total (plutôt au niveau de France Telecom : ex : numéris sur deux canaux).*
- *Contrôle de séquence : vérifie que le numéro de séquence de chaque paquet est OK.*
- *Détection et reprise sur erreurs, contrôle de flux (ex : taille de fenêtre à uniformiser des deux côtés). TCP/IP contient aussi des indicateurs d'erreurs et s'il est mal paramètré, le débit baisse et finit par s'écrouler.*

Notons qu'il y a cinq classes différentes dans le modèle OSI.

V. EXERCICE : PROBA. de PAQUETS ERRONES

TCP

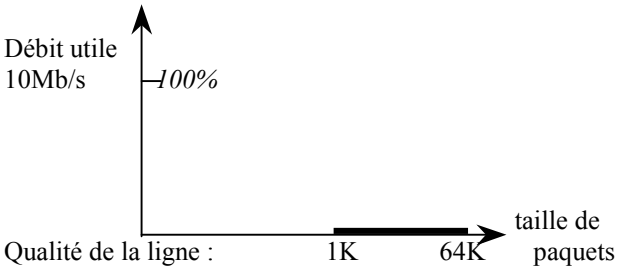
Port source	Port destination
Numéro de séquence	
Acquittement	
Lg | 1 2 3 4 5 6	taille de fenêtre
Champs de contrôle	Pointeur urgent
Options éventuelles	

4 octets

1. URG Pointeur urgent
2. ACK N° acquittement significatif
3. EOM Fin de message
4. RST Réinitialiser la connexion
5. SYN Synchroniser les n° de séquence
6. FIN Libération connexion

Quand j'envoie un paquet d'une certaine longueur, quelle est la probabilité qu'il soit erroné. Sachant que la qualité de la ligne est de 10^{-3} (donc un caractère HS tous les milles).

On peut le faire de manière expérimentale (par simulation) pour voir le débit utile.



T= taille de paquet $\in$ [32, ….. 2^{16}]

$$\text{Débit utile} = \frac{\text{Nbre paquets sans erreur}}{\text{Nbre paquets total}}$$

(nbre paquet sans erreur + nbre paquets proba erreur x3)

P= probabilité d'erreur : t x qualité de la ligne
Nb paquet avec probabilité d'erreur : taille du paquet x P

Voir ADSL pour les lignes téléphoniques avec ATM (petits paquets).

COUCHES HAUTES DU MODELE OSI

LES COUCHES HAUTES DU MODELE OSI.....120

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LES COUCHES HAUTES DU MODELE OSI

I. LA COUCHE SESSION

C'est la première des couches supérieures. On va donc travailler uniquement sur des problèmes logiciels (protocoles…). Cette couche permet à 2 applications tournant sur différents ordinateurs d'établir, d'utiliser et d'interrompre une connexion appelée session. Cette couche procède à l'identification et assure des fonctions, telles que la sécurité, nécessaire à l'établissement de la communication de deux applications sur le réseau. Cette couche assure la synchronisation des taches utilisateurs. Elle permet également de contrôler le dialogue entre deux processus de communication, de savoir d'ou vient la transmission, à quel moment elle se produit, combien de temps elle dure.

- Permet de créer, utiliser et achever une connexion entre 2 ordinateurs.
- Place des points de contrôle dans le flux de données.
- Contrôle le dialogue entre processus communiquant.

La couche session gère les connexions entre les application coopérantes. Avec cette couche, un utilisateur peut se connecter à un hôte, à travers un réseau où une session est établie pour transférer les fichiers. La couche session offre les fonctions suivantes :

- Contrôle du dialogue,
- Gestion des jetons *(le jeton dont il est question ici n'a rien à voir avec le jeton des réseaux Token Ring. La gestion du jeton dans les réseaux Token Ring relève des couches 1 et 2 du modèle OSI, alors que celui dont il est question ici relève du niveau 5)*
- Gestion de l'activité

En général, une session permet des communications full duplex, bien que certaines applications se contentent d'une communication half duplex. La couche session peut fournir une ou deux voies de communication (contrôle du dialogue).

Pour certains protocoles, il est essentiel qu'un seul côté lance une opération critique. Pour éviter que les deux côtés lancent la même opération, un mécanisme de contrôle, comme l'utilisation des jetons (cf. note plus haut), doit être implémenté. Avec la méthode du jeton, seul le côté qui possède le jeton peut lancer une opération. La détermination du côté qui doit posséder le jeton et son mode de transfert s'appellent la gestion du jeton.

Les réseaux TCP/IP ne possèdent pas de couche session, car certaines caractéristiques de cette couche sont fournies par le protocole TCP. Les applications TCP/IP fournissent elles-mêmes certains services. Par exemple, le service *NFS* comporte son propre service de la couche session : le protocole RPC

But :

Le but est de fournir à la couche supérieure les moyens pour gérer le dialogue (et les échanges de données). Sachant que les couches supérieures offrent des services orientés « utilisateur », la couche session fournira des services à la couche « présentation ». Notons que TCP/IP mélange les trois couches supérieures sans respecter vraiment le modèle OSI.

Particularités :

On utilise quatre jetons différents pour représenter les droits exclusifs qui ont été acquis sur demande. Ceci afin d'utiliser des services critiques :

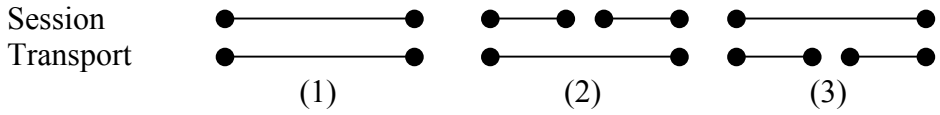
- Données
- Terminaison
- Synchronisation
- Gestion d'activité

On utilise des point de synchronisation qui permettent de structurer un échange de données en dialogue, ils peuvent être majeurs (acquittés) ou mineurs (non acquittés).

Principale fonction :

Etablir des connexions qu'on appelle des sessions et on y transfère les données dans le bon ordre. Notons que TCP/IP est un peu le mélange de la couche transport et de la couche session.

Correspondance entre session et transport.



(1) : correspondance bi-univoque (ex : TCP/IP)
(2) : 2 connexions de session sur une connexion de transport (ex : plusieurs sessions telnet)
(3) : une session sur plusieurs connexions de transport successives.

1.1) Transfert de données

On voit ici la différence avec la couche transport. En cas de libération de la couche session, on vérifie que tous les paquets sont bien arrivés avant de couper la liaison.

S.CONNECT Request (S → session)
T.CONNECT Request (T → transport)

Demandes entre les deux stations pour : - qualité de service
- données expresses

ensuite : S.RELEASE Request

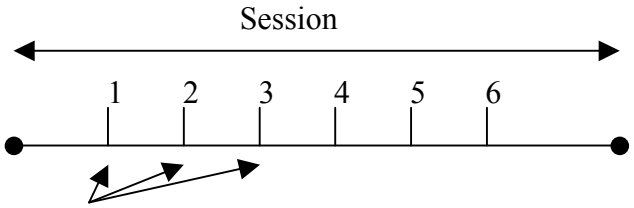
types de données : 4 types (normales, expresses, typées (propriétés de l'objet), capacité (informations de contrôle de la couche session, gestion du protocole)).

1.2) Gestion du dialogue

Une fois la connexion établie, il y a une possibilité de transmission à l'alternat (ex : session telnet). C'est un dispositif mis en œuvre avec la gestion du droit à la parole à l'aide d'un jeton de données. Seul le possesseur du jeton transmet des données. Quand il a fini, il transmet le jeton à son correspondant.

S-TOKEN-GIVE.Request → passer le jeton
S-TOKEN-PLEASE.Request → demande le jeton

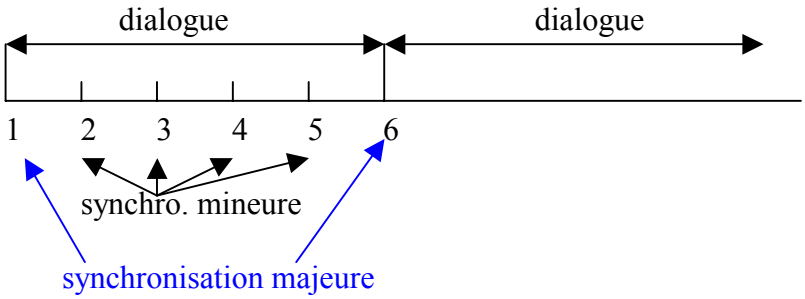
Synchronisation :



points de synchronisation

Si vous transférez un fichier pendant une heure entre deux machines, et qu'une panne réseau intervienne au bout de trente minutes, vous ne pourrez reprendre le transfert là où il s'était arrêté. Il vous faudra toujours reprendre le transfert à son début. Pour éviter cela, vous pouvez traiter tout le fichier comme une seule activité et insérer des points de vérification dans le flot de données. Ainsi, si une coupure survient, la couche session synchronisera à nouveau le transfert, à partir du dernier point de vérification transmis. Ces points de vérification s'appellent « points de synchronisation ». Il en existe deux types : majeurs et mineurs.

Un point de synchronisation majeur est inséré par un des côtés doit recevoir un accusé de réception de la part de l'autre côté, alors qu'un point de synchronisation mineur n'a pas besoin d'être vérifié par un accusé de réception. La session comprise entre deux points majeurs s'appelle une *unité de dialogue*. La gestion de toute l'activité s'appelle *une gestion d'activité*. Une activité consiste en une ou plusieurs unités de dialogue. Chaque point de synchronisation possède un numéro de série d'où reprend la session en cas de rupture. Les dialogues correspondent à des activités…



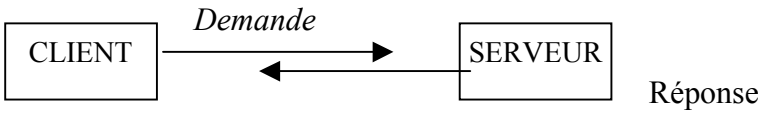
1.3) éléments de protocoles de la couche session

SPDU → unité de données du protocole de session

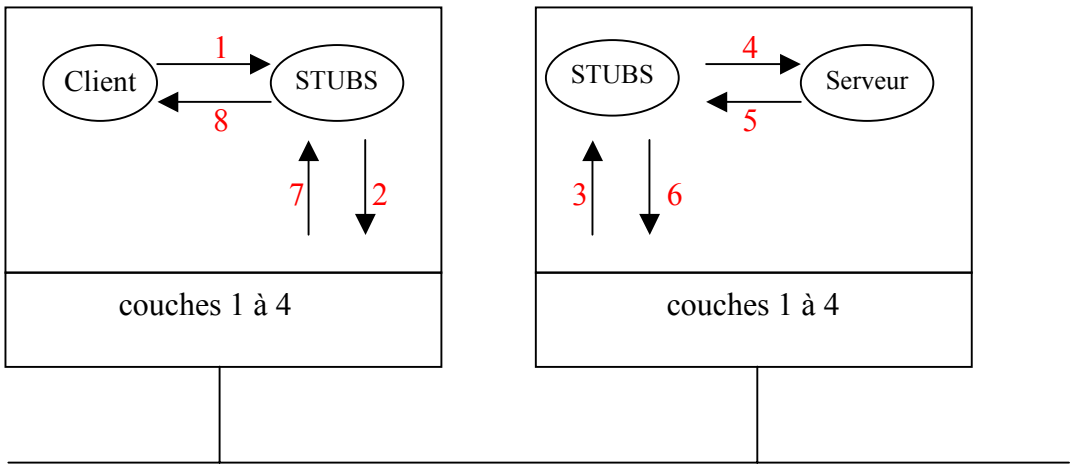
Le noyau contient plusieurs SPDU obligatoires (établissement de connexion, transfert, libération), ainsi que 11 fonctionnalités optionnelles :

- terminaison négociée
- transmission half duplex
- transmission full duplex
- transfert de données expresses
- transfert de données typées
- qualité de service
- synchronisation mineure
- synchronisation majeure
- re-synchronisation
- signalisation d'anomalies
- gestion d'activité

Le modèle client-serveur et appels de procédures à distance



Dans l'appel de procédure à distance (RPC :Remote Procedure Call), un message envoyé est considéré comme un appel de sous programme. On peut inclure le RPC dans le langage lui même (langage développement). Les détails de fonctionnement du réseau sont cachés, ils sont inclus dans des procédures locales (STUBS).



- passage des paramètres : par valeur, par adresse locale non significative ou par URL (Unique Ressource Location).
- pannes du client ou du serveur : nombreux cas particuliers d'erreurs.
- protocole utilisé : exactly once (panne), at most once, at least once.

Grande difficulté à obtenir une transparence totale.

1.4) EXERCICES : Questions - Réponses

1) Pourquoi les SPDU full duplex sont elles vides ?

Elles ne servent à rien, car il n'y a rien à faire (pas de contrôle à faire = pas de jeton à gérer).

2) La couche session à des données express, que deviennent elles dans les couches inférieures ?

Cela dépend. Soit elles sont véhiculées par les services transport, soit elles sont transformées en données normales (si la couche transport n'est pas disponible).

3) Une application d'interrogation d'une base de données est programmée avec le protocole question-réponse. On est obligé d'attendre la réponse pour poser une nouvelle question.
A quelles unités fonctionnelles de session peut-on faire appel ?

Transmission half duplex.

4) Un processus client envoi via un RPC une requête qui se traduit au niveau du serveur par la création d'un processus fils pour réaliser le traitement et renvoyer le résultat au processus client qui est son père. Que se passe-t-il si la machine client tombe en panne ?
Donner une méthode sure pour exterminer les orphelins au niveau du serveur .

Présence d'orphelins sur la machine : le client re-fonctionne au bout d'un temps, comme sur le serveur les requêtes ont un numéro d'identifiant, le client interroge le serveur pour savoir si il y a des requêtes. Si oui, il demande de les supprimer. On peut aussi prévoir des time-out pour les requêtes.

II. LA COUCHE PRESENTATION

Pour que deux systèmes puissent se comprendre, ils doivent utiliser le même système de représentation des données. La couche présentation gère cette représentation universelle des données échangées par des systèmes ouverts. Il existe plusieurs façons de représenter des données, par exemple, l'ASCII et l'EBCDI pour les fichiers texte. La couche présentation utilise un langage commun compréhensible par tous les nœuds du réseau. Cette couche détermine la forme sous laquelle s'échangent les données entre les ordinateurs du réseau; coté émission, elle converti les données du format transmis par la couche application en un format intermédiaire, admis de tous.

Coté réception, elle traduit le format intermédiaire en un format que peut lire la couche application de cet ordinateur. Cette couche gère aussi tous les problèmes de sécurité du réseau en offrant des services tels que le cryptage des données. Elle établit aussi des règles en matière de transfert des données et permet la compression des données de façon à réduire le nombre de bits à transmettre. Elle transforme les données dans un format reconnu par les

applications (traducteur). Elle redirige les données par le redirecteur. Elle est responsable de la conversion des protocoles, l'encodage des données et la compression.

Il faut décrire les structures de données (grâce à ASN1 ISO 8824, 8825 : codage). Elle assure aussi la manipulation de données notamment pour des raisons de sécurité (chiffrement), ou de gain de rentabilité (compression).

2.1) SERVICES ET PROTOCOLES DE PRESENTATION

Cette couche n'est qu'un filtre. Les demandes de connexion (par exemple) passent telles qu'elles (sans être traitées à ce niveau). Le contexte est le structure de données nécessaires pour assurer le fonctionnement de la couche présentation.

P-CONNECT.Request : exécute le « S-CONNECT.Request » de la couche session,

P-ALTERCONTEXT : correspond à un service confirmé.

2.2) SYNTAXE ABSTRAITE (ASN.1)

C'est la description des données dans le langage ASN-1 (Abstract Syntax Notation). On ne fait pas d'hypothèse sur la représentation des données.

Exemple de description ASN.1

```
type tome-2 = record
        titre : array[1..20] of characters ;
        nbauteur : entier ;
        disponible : booleen ;
        nbpages : entier ;
        nbexercices : entier ;
end
```

pseudo Pascal

commentaire

```
type tome-2 : := SEQUENCE {
        titre OCTET STRING(SIZE(20)), -- 20 car
        nbauteur INTEGER ,
        disponible BOOLEAN ,
        nbpages INTEGER ,
        nbexercices INTEGER ,
        pot BOOLEAN Defaut True }
```

(on définit la structure tome-2)
(taille maximum 20 caractères)

syntaxe ASN-1

Les différents types primitifs possibles d'ASN.1

INTEGER	entier	≥0
REAL	réel	
BOOLEAN	vrai ou faux	TRUE/FALSE
BIT STRING	suite de bits	
OCTET STRING	suite de caractères ordonnés	Pas de longueur maxi
ANY	n'importe quel type	Réunion de tous les types
NULL	sans type	complément de ANY
OBJECT IDENTIFIER	nom d'objet	

Entier : pas de limite à la taille maximum, valeur énumérées (janvier = 1, février = 2, …).
Suite de bits : '01001101' B au lieu de '4D' H
ANY : on reporte à plus tard le type de données (considéré comme la réunion de tous les types).
NULL : complément de ANY. Une seule valeur possible (abstraite) : NULL

Quand une connexion est établie, la couche présentation est responsable d'une négociation qui permet de « synchroniser » les représentations pour se mettre d'accord sur la syntaxe abstraite utilisée. Tous les éléments sont des objets identifiés par leur OBJECT IDENTIFIER.

Les types complexes (construits) d'ASN.1

- SEQUENCE est une collection ordonnée d'éléments de divers types (structure).
- SEQUENCE OF est une collection ordonnée d'éléments de même types (tableau).
- SET est un ensemble d'éléments de divers types (sac).
- SET OF est un ensemble d'éléments de même types (ensemble).
- CHOICE est un choix d'un type dans une liste (union).

Exemple de description ASN-1

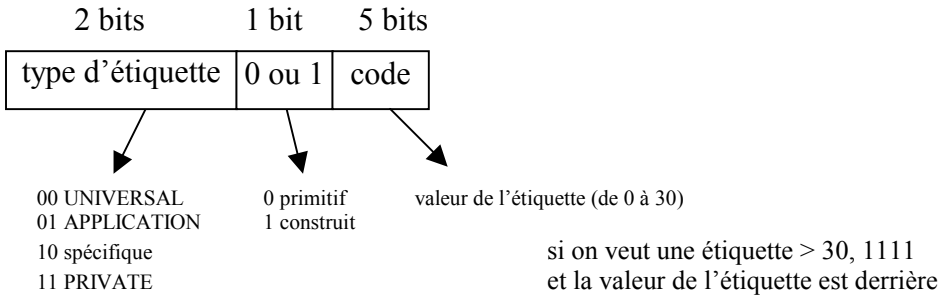
```
Données utilisateur : := CHOICE {
        [APPLICATION 0] IMPLICIT Données-codage-simple
        [APPLICATION 1] IMPLICIT Données-codage-integral }

      Données-codage-simple : := OCTET STRING
      Données-codage-integral : := SEQUENCE OF Liste-PDU

      Liste-PDU: := SEQUENCE {
        Nom-syntaxe-transfert OPTIONAL,
        Identificateur-contexte-presentation : := CHOICE {
              Type-ASN1-unique [0] ANY,
              Aligné –octet [1] IMPLICIT OCTET STRING,
              Arbitraire [2] IMPLICIT BIT STRING }
                                    }          (séquence)

      Nom-syntaxe-transfert : := OBJECT IDENTIFIER
      Identificateur-contexte-presentation : :=INTEGER
```

Syntaxe de transfert ASN-1

Chaque objet possède quatre champs :

1) identificateur (type ou étiquette)
2) longueur du champ de données
3) le champ de données
4) fanion de fin si la longueur est inconnue

1) identificateur :



2) longueur du champ de données : 1 octet si ≤ 254 sinon sur 2 octets

3) le champ de données

Pour les entiers : bit de poids fort en tête
Pour les booléens : 0 FALSE, ≠0 TRUE
Pour les chaînes de bits : pas de codage (transmis tel quel) la longueur est exprimée en caractères, on indique dans le 1er octet le nombre de bits utiles dans le dernier.

Ex :

4	F	8	0
0100	1111	1000	0000

données utiles

Ce qui donne 03 01 4F 80 (en hexa).

Longueur des données — contrôle (combien de bits utiles dans le dernier octets)

Pour les chaînes de caractères : transmis tel quel avec poids fort en tête. La valeur NULL est indiquée par un champ de données de longueur nulle.

EXERCICE : DESCRIPTION ASN-1

Donner la syntaxe de transfert avec :

	Code de l'objet (n° de label)
titre = TOME 2	INTEGER = 2

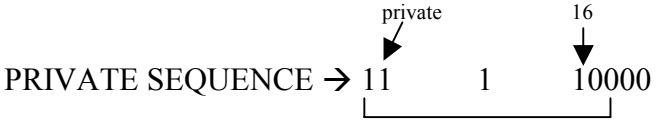
nbauteur = 4
disponible = true *(n'importe quoi sauf 0)*
nbpages = 280
nbexercice = 75
pot = true

BOOLEAN = 1
OCTET STRING = 4
SEQUENCE = 16

On suppose qu'il s'agit d'étiquettes UNIVERSAL. Code ASCII avec parité impaire :

T	O	M	E		2
54	4F	4D	45	20	32
01010100	01001111	01001101	01000101	00100000	00110010

TOME 2 → 04 06 54 4F CD 45 A0 32
4 → 02 01 04
true → 01 01 01
280 → 02 02 01 18
75 → 02 01 4B



Longueur = 24 =0x18 = FO 18

EXERCICE : Meilleurs scores « jeux ordinateur » de la famille TERES.

La famille Teres est composée de 6 personnes, toutes passionnées de jeux sur ordinateur individuel. On souhaite créer un fichier pour stocker les meilleurs scores sous forme d'enregistrements :

- Prénom
- Jeu (libellé alphanumérique + code hexadécimal)
- Niveau (de 1 à 4)
- Dernier score obtenu (de 1 à 99 999).

1) Donner un exemple de représentation en ASN.1 pour spécifier la structure de chaque membre puis la liste entière de ses membres.

```
Joueur_TERES : : SEQUENCE  {
        Prenom OCTET STRING,
        Jeu_prefere jeu,
        Niveau INTEGER (1..4),
        Dernier_score INTEGER (1..99999),
        Date GENERALIZEDTIME  }

Jeu : : SEQUENCE  {
        Libelle OCTET STRING,
        Code OCTET STRING,  }

Famille_TERES : : = SEQUENCE OF Joueur_TERES
```

2) « Initialiser » un membre.

Pere : = = {« jean-luc », {« echec », « 1B7 » H }, 4, 99997, 19 94 04 01 23 31 00.0 }

3) Spécifier à nouveau la liste sous forme d'ensemble.

Il suffit de remplacer SEQUENCE OF par SET OF

4) Ajouter un chat à la famille et donner la nouvelle structure.

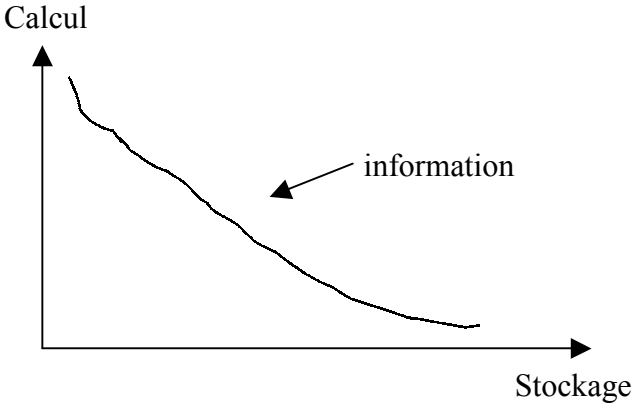
Il suffit de créer une structure chat et on mettra dans « Famille_TERES » un CHOICE entre Joueur_TERES et chat.

2.3) COMPRESSION DE DONNEES (non destructive)

Information

↓

(calcul 1 + stockage 1) = (calcul 2 + stockage 2)



La compression de données, consiste à présenter l'information sous une forme mieux adaptée à la transmission. On veut minimiser les coûts, la durée, les risques (confidentialité). Si on veut évaluer ou fabriquer des algorithmes de compression, il faut calculer la quantité d'information, c'est l'entropie d'un système d'information (le minimum d'information sans pertes).

Si on connaît un alphabet avec des symboles Si avec des occurrences de probabilité Pi.

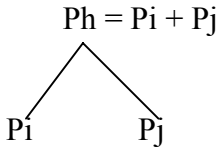
$$\underbrace{\sum_{i=1}^{N} Pi \log_2 Pi}_{\text{entropie par symbole}}$$

avec N = nb de symboles
Pi = probabilité d'occurrence du symbole i

Code de Huffman

Ce codage fonctionne bien s'il y a une équiprobabilité d'occurrence des symboles. C'est un codage spécifique au jeu de données. Déroulement :

1. Ecrire la liste de tous les symboles et leur probabilité d'occurrence.
2. Construire un arbre binaire dont les symboles sont les nœuds terminaux.
3. Trouver les deux plus petits nœuds (plus faible probabilité d'occurrence) et les marquer.
4. Construire ensuite un nouveau nœud avec 2 axes qui permettent d'atteindre les nœuds. La probabilité de ce nœud est la somme des probabilités des nœuds connectés.
5. Répéter les étapes n°3 et 4 jusqu'à ce que tous les nœuds soient marqués sauf un.
6. Le nœud non marqué est la racine de l'arbre, et sa probabilité est égale à 1.
7. Le codage de chaque symbole correspond au chemin à parcourir depuis la racine jusqu'à ce symbole en mettant à chaque fois les embranchements (droite = 1 et gauche =0). Le code d'un mot est donc le chemin obtenu.

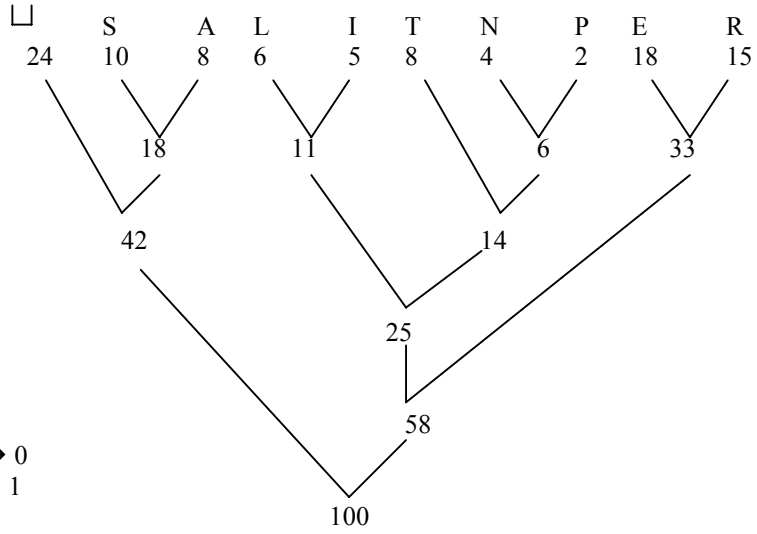


EXERCICE : Codage de texte selon le pourcentage d'occurrence.

Pourcentage d'occurrence :

␣	E	R	S	A	T	L	I	N	P
24%	18%	15%	10%	8%	8%	6%	5%	4%	2%
00	01	100	1010	1011	1100	1101	1110	11110	11111

1) Construire l'arbre et coder les différentes lettres. Quelle est la longueur moyenne du code ? Comparer avec un code de longueur fixe. Comparer avec la longueur minimale théorique.



␣	00	longueur moyenne du code	0,24 * 2	(0,24 → 24%)
E	110		0,51 * 3	(0,18+0,15+0,10+0,08)
R	111		0,19 * 4	(0,08+0,06+0,05)
S	010		0,06 * 5	(0,04+0,02)
A	011			
T	1010			
L	1000		0,48	
I	1001		1,53	
N	10110		0,76	
P	10111		0,30	
			3,07	

Longueur théorique minimum = 3,02 bits/caractère
Longueur moyenne de codage = 3,07 bits/caractère
Ici, on peut donc utiliser un codage fixe sur 4 bits.

Codage par plage

Ce codage est plus particulièrement utilisé pour coder des chaînes qui contiennent des longues plages de 0 ou bien des répétitions de plages. Chaque symbole de k bits indique combien de « 0 » séparent deux « 1 » consécutif. Ex : 111111 AAAAA BBBBB. Dessins N&B au trait…

Le principe est de détecter des répétitions d'occurrence et de factoriser.

000100000010010000000000000001000001000100000011010000101
3 6 2 15 5 3 6 0 1 4 1 (plage de zéros)

Soit avec un codage par plage sur 3 bits :

011 110 010 111111001 101 011 110 000 001 100 001
3 6 2 15 5 3 6 0 1 4 1

Notons que le chiffre 7 serait représenté par 111 00 car 111 indique que la plage suivante fait partie de la même suite.

EXERCICE : Codage de suite de bits par plage.

Soit une suite de bits avec les plage suivantes :
16x0 ; 1x1 ; 15x0 ; 1x1 ; 14x0 ; 1x1 ; 1x0 ; 1x1 ; 0x0 ; 1x1 ; 0x0.

1) Utiliser un codage par plage avec 4 bits et coder la suite. Quel est le gain ?

16	15	14	1	0	0
1111 0001	1111 0000	1110	0001	0000	0000

Avant nous avions : 16+1+15+1+14+4 = 51 bits. Et après codage, nous avons : 6x4=28 octets.

On a donc 2800/51 ce qui donne environ 55% de gain.

2) Appliquer une seconde fois l'algorithme et conclure.

Cette fois, nous avons les plages suivantes : 4x1.3x0.5x1.4x0.3x1.4x0.1x1.
Ce qui donne en codage par plage de zéro : 00003000040044 à coder ce qui ne favorise pas vraiment le gain ! Il faut vraiment étudier les occurrences pour optimiser le codage et il ne sert pas à grand chose d'appliquer plusieurs fois de suite le même algorithme.

EXERCICE : Transferts pour une application bancaire.

Une application bancaire transfère 100 000 transactions chaque jour de la succursale vers le siège. Chaque transaction comporte :

- L'entête : 4 octets
- Le type : 1 octet
- Le numéro : 10 octets (entier positif)
- Le montant : 10 octets (entier positif)
- Le code banque : 8 octets (entier positif)
- La date & heure : 5 octets
- Les infos bancaire : 400 octets (texte de a-z, A-Z et espaces)
- Et la clef : 4 octets

Le codage est ASCII et se fait sur 8 bits. L'enveloppe globale X.25 – LAP.B est de 20 octets. Le débit binaire est de 9600 b/s et chaque paquet contient 512 octets au maximum.

1) Calculer le temps de transmission total.

Une transaction est donc composé de 442 octets. On peut faire des paquets de 512-20=492 octets. Si l'on prend une transaction par paquet, on doit donc faire des paquets de 442+20=462 octets minimum. Le codage étant sur 8 bits et le débit étant de 9600 b/s cela donne : 462x8/9600=385 ms par paquet. Soit 38 500 s = 10,7 heures au total pour tout transmettre.

2) Chaque minute revient à 50 cts. Quel est donc le coût total de la transmission ?

Le coût total de la transmission sera donc de 50x60x10,7=320 Frs environ.

3) Proposer une méthode pour payer moins cher.

On peut compresser les données « infos bancaire » (pour ne pas devoir décompresser toute la transaction à chaque fois, on connaît « l'identité » de la transaction). Mais cela dépendra beaucoup de type des données de ce champ.

III. LA COUCHE APPLICATION

C'est la couche la plus haute du modèle de référence. Elle offre des services à l'utilisateur final. Cette couche, appelée ALS : Application Layer Structure, est la fenêtre par laquelle les processus d'application accèdent aux services du réseau. Normalisée sous ISO 9545 (définit la composante communication).Elle représente les services qui prennent en charge les applications utilisateur, par exemple:

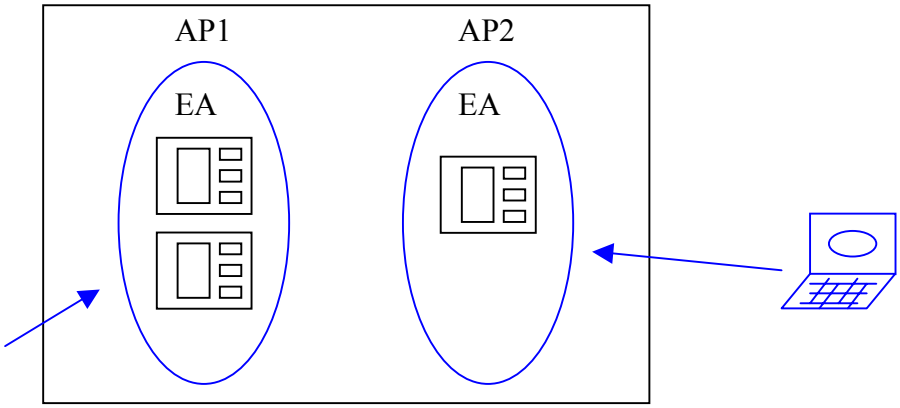
- Les protocoles pour les services de fichiers distants tels que l'ouverture, la fermeture, la lecture, l'écriture et le partage de fichiers
- Les services de transfert de fichiers et d'accès aux bases de données distantes.
- Les services des répertoires pour localiser les ressources d'un réseau.
- La gestion des périphériques
- L'exécution de travaux distants.

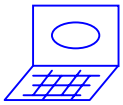
ALS a pour but d'assurer les qualités suivantes :

- L'atomicité : un ensemble d'opération doit être entièrement réalisé ou alors pas du tout .
- La cohérence : un ensemble d'opérations doit être termine correctement et les informations doivent être laissées dans un état cohérent.
- L'isolation : un ensemble d'opérations doit s'exécuter sans interférence avec les opérations extérieures.
- La durabilité : les effets des opérations ne doivent pas être altérées par une défaillance applicative ou de communication.

3.1) Le modèle générique

- Le processus d'application (AP) est la représentation abstraite des éléments d'un système (programme écrit par exemple).
- L'activité d'un processus (un programme qui s'exécute) (API : Application Process Invocation) est représenté par une invocation de processus d'application.
  Un processus d'application peut être fabriqué à partir d'autres processus communicants.
- Un processus d'application peut être coupé en 2 parties :
  - locale (calcul) et spécifique d'un système.
  - Partie communicante représentée par une EA (Entité d'Application).
- Une EA n'est attachée qu'à un seul processus d'application (AP) mais un processus d'application peut compter plusieurs EA.





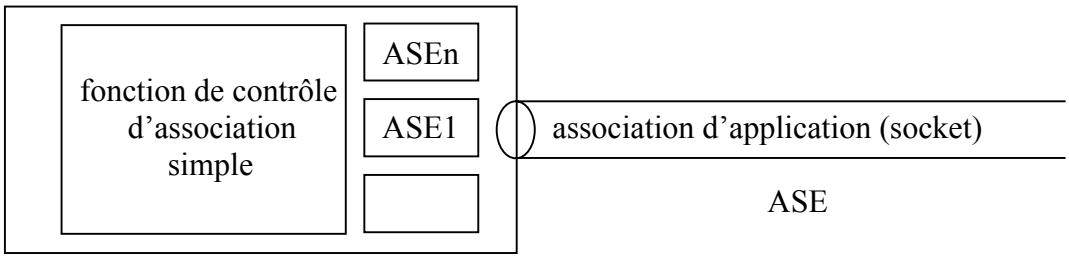
Couche présentation

- L'activité d'une EA est représentée par une invocation d'entité d'application (AEI : Application Entity Invocation) qui modélise les fonctions de communication particulières d'une API.
- Les EA comportent un ensemble d'éléments de service d'application (ASE : Application Service Element).

3.2) Association d'application (AA)

Connexion établie entre deux AEI. Une AA ne peut être créée que après l'activation des deux AE correspondants.

Entité d'application



AAI (Application Association Identifier) : identificateur d'association d'application (n° circuit virtuel, nom de socket…) (c'est le nom de la connexion).

SAO (Single Association Object) : ensemble des fonctions et des informations de contrôle relatives à une association.
Un SAO (objet d'association simple) contient des informations sur la combinaison d'ASE qui est utilisée dans le cadre d'une association. La collaboration entre ASE à l'intérieur d'un SAO est régie par des règles internes (SACF : fonction de contrôle des associations simples).

3.3) Les ASE de base

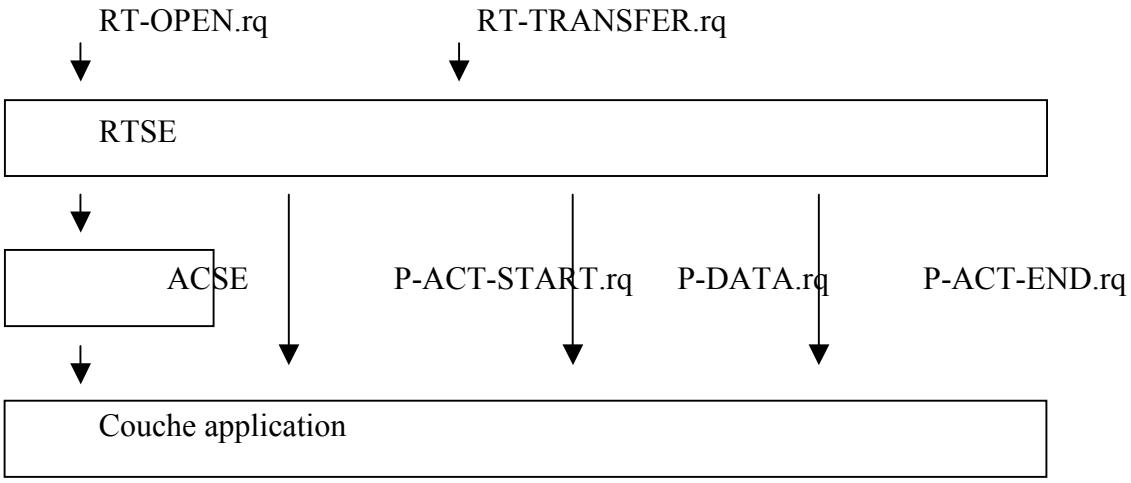
Elément de service (protocole) :

- RTSE (Reliable Transfert Service Element) : élément de service de transfert fiable.
- ROSE (Remote Operation Service Element) : utilitaire pour mettre en œuvre des applications interactives (telnet).
- CCRSE (Commitment, Concurrency and Recovery Service Element) : élément de service d'engagement, de concurrence et de reprise qui permet aux entités d'une application repartie de réaliser des actions atomiques avec une coordination garantie.

- CMISE (Common Management Information Service Element) : échange d'informations de gestion.
- ACSE (Association Control Service Element) : iso 8649 et CCITT X217 et iso 8650 et CCITT X227.

Primitives :
- A-ASSOCIATE : service confirmé comprenant une requête, une indication, une réponse et une confirmation.
- A-RELEASE : service confirmé avec fermeture de connexion sans perte.
- A-U-ABORT : (U : user) service non confirmé, libération brutale avec perte.
- A-P-ABORT : (P : Provider) service non confirmé, rupture de liaison brutale avec perte

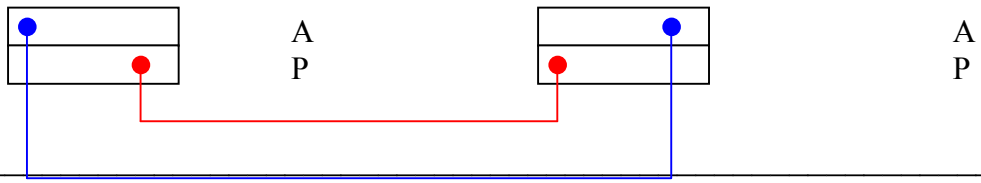
3.4) RTSE



EXERCICE

1) Quelle correspondance existe t'il entre association d'application et connexion de présentation ? Quels sont les services de présentation requis par ACSE ?

association d'application
connexion de présentation



a) une association d'application = une connexion de présentation
b) P-CONNECT
   P-ABORT

EXERCICE

A-ASSOCIATE
APDU

1) Sélecteur de mode (normal par défaut).
2) Nom du contexte.
3) Paramètre d'adressage
4) Informations de l'utilisateur.
5) Adresse de présentation de l'entité appelante.
6) Adresse de présentation de l'entité appelée.
7) Reste de définition des contextes de présentation.
8) Nom de contexte par défaut.
9) Informations de qualité de service.
10) Propositions de l'utilisateur de service de présentation.
11) Propositions de l'utilisateur pour la session.
12) Numéro de service du point de synchronisation initial.
13) L'attribution initiale de jeton.
14) L'identificateur de connexion de session.

Ces informations sont replacées dans la requête P-CONNECT puis vers S-CONNECT

Q : justifier la présence des paramètres 11,12,13 et 14 dès la couche 7.

R :dans le cas half duplex, il faut savoir qui va commencer le dialogue. Il faut donc gérer le jeton dès le niveau application.

3.5) LES APPLICATIONS

Application principale : transfert de fichiers

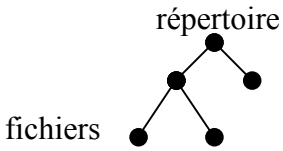
- système de fichiers virtuels → abstraits (cas d'un serveur de fichier NFS).
- Protocoles normalisés.

(il y a 4 ans, 100 Go en ligne → 1 MF
— 2 ——————— → 250 KF
— 3 ——————— → 50 KF)

a) un fichier n'est qu'une suite de données sans structures particulières
   - lecture
   - écriture
   - positionnement

b) les structures à accès directe se trouvent au niveau application

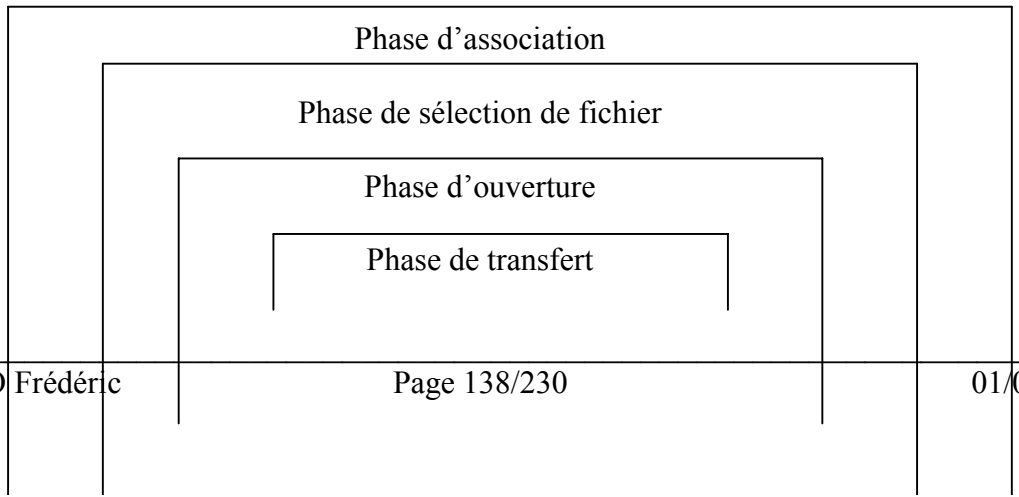
c) modèle hiérarchique



Les fichiers sont des objets conservés :
- nom
- taille (implémentation)
- type (ce qu'on peut en faire)
- date (création, modif, accès)

* FTAM : File Transfert Access and Management

- correspondance entre fichiers virtuels et réels.
- le traitement du fichier virtuel est orienté connexion.



F-read F-transfert
F-write end

f-open f-close

f-select f-deselect
ou f-create ou f-delete

f-initialize f-terminate

* Primitives de FTAM

- F-initialize : établissement d'une connexion
- F-terminate : libération d'un connexion

- F-U-Abort : interruption à l'initiative de l'utilisateur
- F-P-Abort : interruption à l'initiative du fournisseur

- F-Select : sélection d'un fichier
- F-Deselect : fin de sélection

- F-Create : création de fichier
- F-Delete : destruction de fichier

- F-Read-Attrib : lecture des attributs d'un fichier
- F-Change-Attrib : changement des attributs d'un fichier

- F-open : ouverture d'un fichier
- F-close : fermeture d'un fichier

- F-Begin-Group : début d'une action atomique (indivisible)
- F-End-Group : fin d'une action atomique

- R-Recover : restauration après incident

- F-locate : déplacement d'un pointeur FADU (File Access Data Unit = répertoire)

- F-Erase : effacement d'une FADU

- F-Read : lecture d'une FADU
- F-Write : écriture d'une FADU
- F-Data : arrivée des donnée
- F-Data-End : fin de FADU
- F-Transfert-End : fin de transfert

- F-Cancel : abandon de transfert

- F-Check : pose d'un point de contrôle
- F-Restart : retour vers un point de contrôle

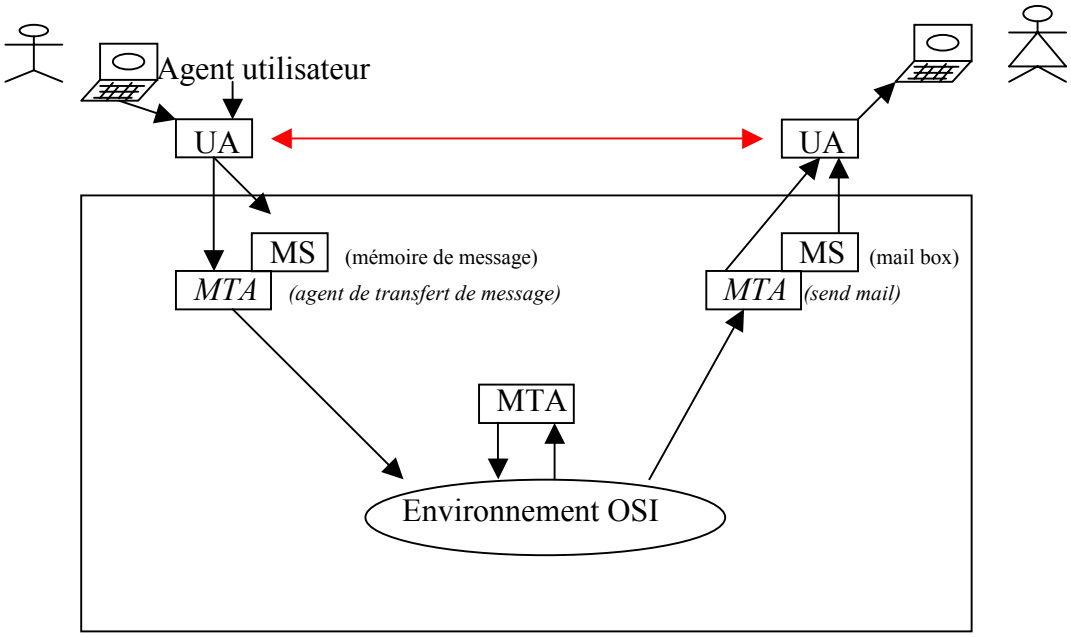
Messagerie électronique X400

Décrite par la norme X400 et ISO 10021

- Composition : fabrication de messages
- Transfert : acheminement vers le destinataire. ACSE de la couche présentation.
- Information : dire à l'émetteur ce qu'est devenu son message
- Conservation : on stocke temporairement l'information (mbox)
- Remise : que fait on des messages ?
- Facilités : faire suivre, répondeur automatique, copie carbone, accusé de réception, accusé de réception, blind copie carbone, liste de diffusion, recommandé (pour savoir si le message est arrivé).
- SPAM : redirection du courrier en copie carbone.
- Messageries : relais-passerelles.

Idée : encapsulation d'un message et envoi à un MX (Mail eXchange)

SMTP → X400 on enveloppe le message X400 dans une enveloppe SMTP

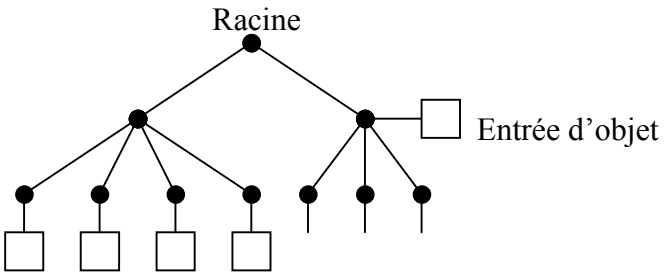


MTS : Système de transfert de messages

Serveur d'annuaire électronique X500

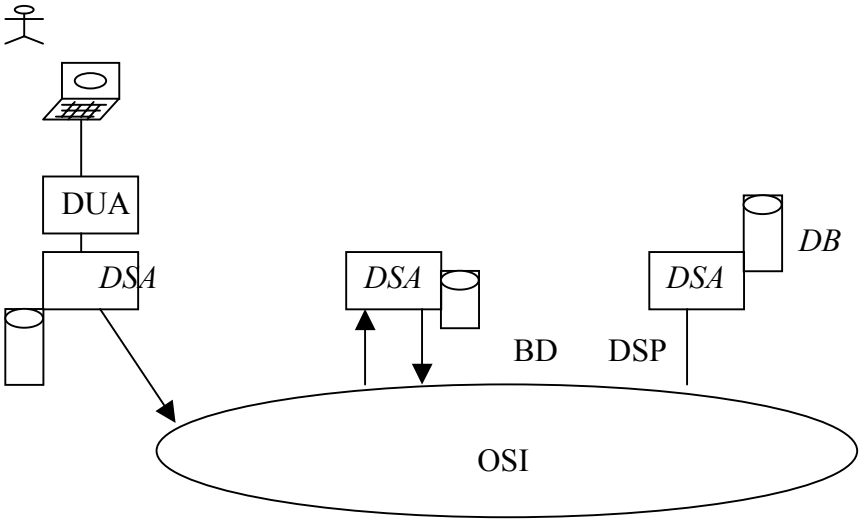
- L'ensemble des informations contenues dans l'annuaire constituent la base d'information de l'annuaire (DIB : Directory Information Base).
- C'est un arbre d'information de l'annuaire (DIT : Directory Information Tree).
- L'information élémentaire

  RDN (Relative Distinguished Name)
  DN (nom spécifique)



- Accès à l'annuaire

  DUA : Directory User Agent



Autres applications

- Transfert et gestion de travaux à distance : JTM
- Architecture de documents de bureau : ODA (Office Document Architecture)

  → Utilise ASN-1 (format)

  → Utilise ODIF (Office Document Interchange Format)
- Messagerie commerciale EDI : Electronique Data Interchange

Une autre norme a été définie autour de EDI : EDIFACT → ISO 9735
(Electronic Data Interchange For Administration Commerce & Trade)

- Traitement réparti ouvert : ODP (Open Distributed Processing)
  Pour faire du // sur un réseau
- Modèle d'applications Bureautique distribuées
  DOAM (Distributed Office Application Modele)
- Traitement transactionnel : TP (Transaction Porcessing)
  Norme OSI TP (ISO 10026)
- Terminal virtuel (ex : telnet) : VT
- Tests de conformité de protocole
  ISO 9646
  TTCN (Tree Tabular Combined Notation)
  Décrit une suite de tests pour vérifier la conformité d'un produit aux normes ISO.

PROCESSUS ET APPLICATIONS REPARTIES

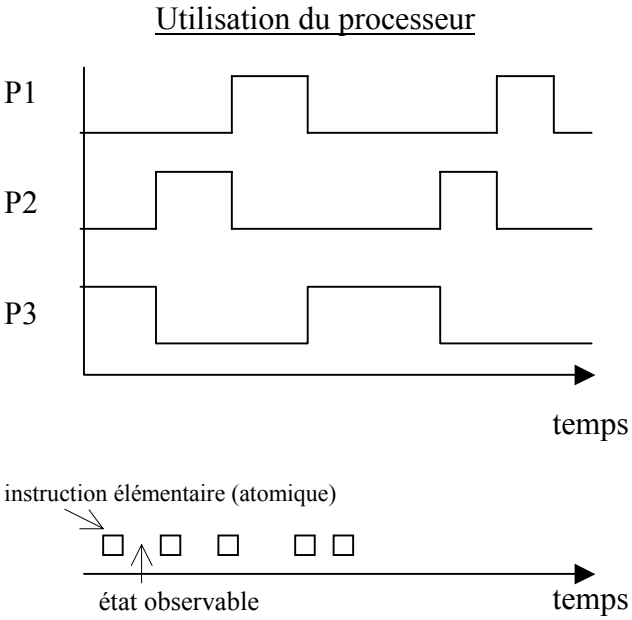
PROCESSUS & APPLICATIONS REPARTIES.....145

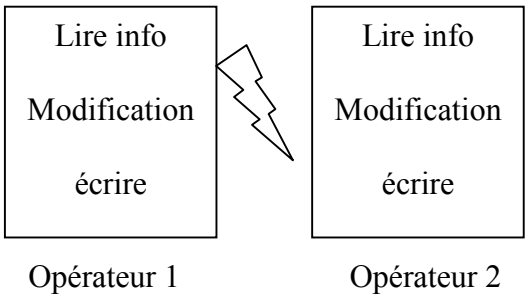
RESEAUX – Cours du CNAM BORDEAUX 1999-2000

PROCESSUS & APPLICATIONS REPARTIES

I. INTRODUCTION

Que ce soit dans une machine, un système ou un réseau, on peut distinguer 3 composants : un serveurs, des clients et des ressources. On introduit alors les processus (activité indépendante). Un processus est l'exécution d'un programme ou plus précisément la suite temporelle d'exécution (interprétation) d'instructions. Un processeur est un dispositif qui permet à un programme de s'exécuter.





1.1) Quelques définitions :

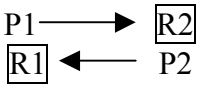
- Vecteur d'état d'un processus : ensemble des variables et procédures qu'il peut utiliser.
- Point observable : instant où l'on peut observer l'état d'un processus.
- Ressource : ce qui est nécessaire à un processus pour fonctionner (Ex : mémoire, un processeur, une information, un périphérique…).
- Les différents états d'un processus :
  - Actif : le programme à tout ce qu'il faut pour fonctionner.
  - Bloqué : en attente de ressources pour continuer (Ex : un processus P fait un calcul et range le résultat dans un buffer. Q ne peut continuer que si le buffer est rempli).

- Accès au ressources :
  - Locale : n'est utilisé que par ce processus.
  - Commune : partageable avec n point d'accès.
- Pouvoir d'un processus : ensemble d'infos qui lui définissent ses ressources accessibles.
- Identité d'un processus : information qui permet de le repérer.
- Synchronisation : - Interactions entre processus (Ex : libération des ressources).
  - Notification par les processus eux même.
- Communication : - Echange de résultats intermédiaires.
  - Echange d'états.

ex : CU sous Unix (émulation terminal).

1.2) Dangers d'une application répartie

- Incohérence des données (d'où l'idée d'exclusion mutuelle),
- Mort subite…
- Blocage ou inter-blocage (*dead lock*),



Ex Pour planter un machine Unix :
cat > vers
vers &
vers &
ctrl D
./vers &

pour arrêter cela :
> vers

1.3) Outils de gestion de partage de ressources

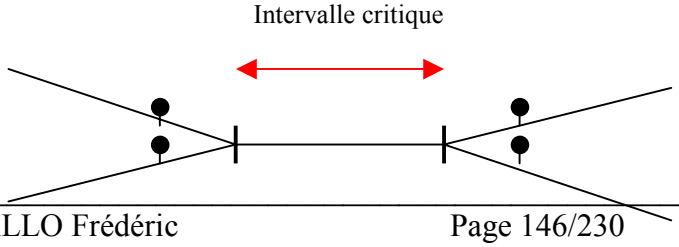
- Sémaphores (introduits par Dijkstra vers 1967-68),
- Rendez-vous ADA (J. David Ichbiah),
- Tubes de communication + socket (chaussette),
- Mémoire partagée…

II. EXCLUSION MUTUELLE

Exemples :

- Une imprimante est disponible et plusieurs personnes veulent l'utiliser. Le premier doit finir d'imprimer avant que le second ne puisse s'en servir.

- Un train, deux voies et un pont à une voie.



Un Intervalle critique est un intervalle de temps pendant lequel un processus utilise la ressource critique (qui ne peut être utilisée que par un seul processus à la fois).

2.1) Contraintes à respecter

Pour gérer une exclusion mutuelle, il faut respecter ces quatre règles élémentaires :

a) A tout instant, un processus et un seul (au plus) peut se trouver en section critique.
b) Si plusieurs processus sont bloqués en attente de la ressource critique, alors qu'aucun processus ne se trouve en section critique, l'un d'eux doit pouvoir y entrer au bout d'un temps fini (ceci pour éviter les éventuels blocages).
c) Si un processus est bloqué en dehors d'une section critique, il ne doit pas empêcher l'entrée d'un autre processus dans cette section critique.
d) La solution doit être la même pour tous le processus (aucun processus ne doit jouer de rôle privilégié).

S'il existait une fonction d'exclusion mutuelle (section critique), la suite d'instructions serait :

```
début
        entrée
        section critique
        sortie
fin
```

2.2) Attente active (par implémentation)

Pour cela, il faut déclarer une variable P commune aux processus ayant pour valeur 0 ou 1 (booléen) suivant que la ressource est libre ou occupé. Un processus doit consulter P et le remettre à zéro après usage. Il faut savoir qu'il existe (depuis 1980) une instruction en code machine indivisible : **Test And Set (TAS)** qui recolle 2 instructions atomiques pour en faire une indivisible. On appelle aussi cela une solution universelle sur machine monoprocesseur.

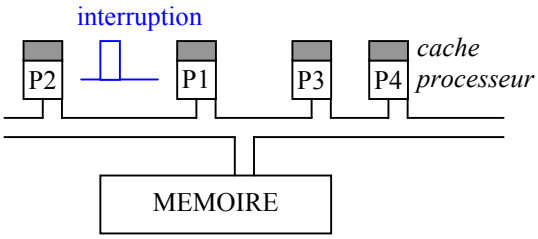
Machines monoprocesseurs (préhistoire) :

Masquer les interruptions + déroutements

Code section critique (à partager entre tous les processeurs) | TAS (P) *(test + positionnement)*

Valider les interruptions + déroutements.

Machines multiprocesseurs :



Bi-processeur : pendant que P1 traite les instructions, P2 regarde en mémoire. C'est donc facile à réaliser.

Cache : les processeurs copient la mémoire dans leur cache. Donc quand un processeur ne fait rien, il met à jour le cache.

Machines en réseau :

Les ressources (processeurs) sont banalisées. Actuellement, les calculs sont répartis sur une centaines de machines pour avoir une puissance de 70-80% de la puissance totale. Il vaut mieux avoir un protocole avec des transmissions de messages plutôt que de la mémoire partagée etc… On execute un sous programme ou un morceau du programme en autonomie sur une des machines avec seulement le passage des paramètres et des résultats par le réseau.

2.3) Les verrous

Un verrou est un mécanisme qui permet de contrôler l'accès à une ressource critique. Les verrous sont constitués d'une variable P et d'une file d'attente f(P). On accède à cette variable par deux primitives :

Au départ : P=0.

Verrouiller (P) : si P=0 alors P=1
sinon mettre P1 dans la file d'attente f(P).
(ce qui le fait passer à l'état bloqué)

Déverrouiller (P) : si f(P) n'est pas vide
alors sortir un processus de f(P). *(ce qui le rend actif)*
sinon P=0 ;

Exemple :

```
Calcul 1 ;
Verrouiller (P) ;
        Calcul 2 section critique ;
Déverrouiller (P)
Calcul 3 ;
```

2.4) Solution réseau

On a une machine dédiée qui n'accepte qu'une communication à la fois, ce qui permet d'enchaîner les processus. *(Avec des moniteurs par exemple…)*

Paquets réseau :

P3		P2	P7	P5	

III. LES SEMAPHORES

En 1967-1968, Mr Dijkstra invente le principe des sémaphores. Ce concept permet de synchroniser des activités parallèles. Il sera très vite adopté pour des applications informatiques très « gourmandes » en activités parallèles.

3.1) Définitions

Un sémaphore « s » est constitué d'une variable entière e (s) et d'une file d'attente f (s) avec e(s) ∈ Z (positif ou négatif), c'est la valeur du sémaphore. On ne fait pas d'hypothèse sur la gestion des files d'attente. Un sémaphore est créé par une déclaration qui précise la valeur initiale $e_0(s)$ (nécessairement une entier non négatif). Il correspond aussi au nombre de ressources à gérer. f(s) est toujours vide à la création. On a deux opérations (primitives, méthodes, appels systèmes) indivisibles de base :

P(s) : pour demander des ressources au système

```
e (s) = e (s) – 1                                   Puis-je disposer de la ressource ?
si e (s) < 0    alors    état (R) = bloqué          Si le processus R exécute P(s)
                         mettre processus R dans la file du sémaphore f(s)
```

V(s) : pour libérer des ressources du système

```
e (s) = e (s) + 1                                   Vas-y c'est libre !
si e (s) ≤ 0    alors    sortir un processus Q de la file f(s)
                         état (Q) = actif
```

3.2) Propriétés des sémaphores

L'exécution des primitives P(s) et V(s) laisse invariante la relation :

nf(s) = min (np(s), e0(s) + nv(s))

avec :

nf : nombre de processus qui ont franchi P(s)
np : nombre d'instructions P exécutées sur le sémaphore s
nv : nombre d'instructions V exécutées sur le sémaphore s
e0 : valeur initiale du sémaphore

3.3) Sémaphore d'exclusion mutuelle

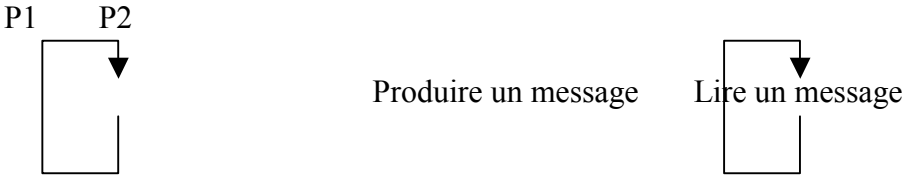
Soit le sémaphore d'exclusion mutuelle mutex = 1. Pour demander un accès exclusif, il va falloir avoir les instructions suivantes:

```
...instructions...
    P(mutex)
        section critique
    V(mutex)
```

…instructions…

Exemple d'utilisation :

On a de la place pour n messages. Au début : on a n cases vides et 0 cases pleines dans le buffer. Si tout est plein : on a 0 cases vides, n cases pleines.



Sémaphore plein = 0, vide = n ;

P1 : *(producteur)*

Produire un message
P(vide) *(demande si il y a une case vide de libre)*
Déposer le message dans la case ← *(section critique)*
V(plein) *(on notifie qu'une case est remplie)*

P2 : *(lire des messages pour les imprimer)*

P(plein) *(demande si une case est pleine)*
Lire le message ← *(section critique)*
V(plein) *(on libère la case)*
Imprimer le message

A l'intérieur d'une section critique *(si l'on a P(mutex) au début et V(mutex) à la fin)*, il ne faut jamais mettre un autre P(…) sous risque d'interblocage avec un autre procesus. Dans un réseau, on sort d'un interblocage au bout d'une temporisation (reste à la configurer correctement selon les temps des processus).

Exemple de « Dead Lock » (interblocage) :

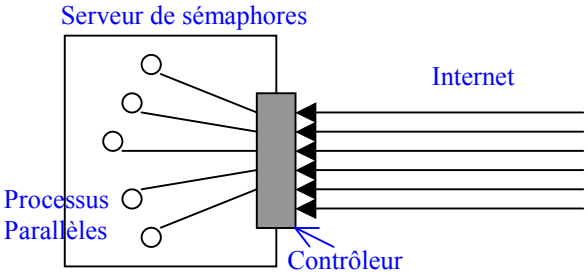
P1 :	P2 :	
P(mutex)	P(mutex)	
P(RC)	P(RC)	*(blocage car si RC est déjà prise,*
V(mutex)	V(mutex)	*il ne fera jamais le V(mutex))*
V(RC)	V(RC)	

3.4) Utilisation des sémaphores (à travers le réseau)

L'implémentation des mécanismes de type sémaphore sur un réseau permet de limiter une quantité de ressource. En fait, le P(s) est exécutable par le propriétaire mais le V(s) est exécutable par tout le monde. Pour gérer l'exclusion mutuelle, il faut dédier une machine (le notaire) pour gérer des sémaphores. Cela garantit l'unicité (une machine) et l'atomicité de P et V (mécanisme de socket).

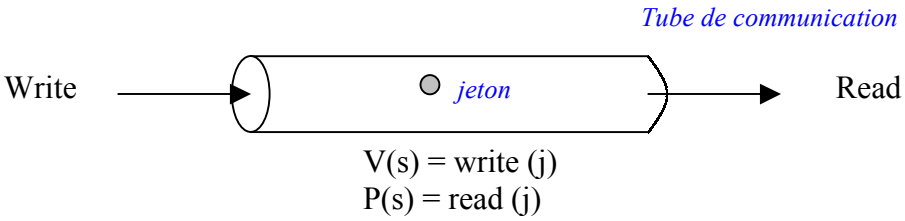
Chaque opération sur sémaphore correspond à l'établissement d'un socket, puis l'échange de message et enfin la fermeture du socket (tube de communication):

- Création, initialisation (requête sur le réseau),
- P(…)
- V(…)
- Destruction.

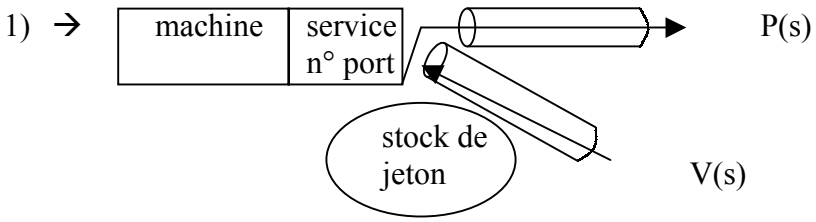


Il faut se débrouiller pour que sur un sémaphore donné, il n'y ait exécution que d'une seule primitive à la fois. Il faut donc mettre en œuvre, sur la machine locale, des exclusions mutuelles pour avoir des processus en parallèle. Dès que l'on a un socket, cela correspond à une exclusion mutuelle.

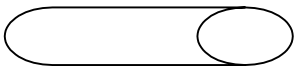
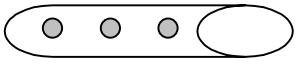
Tubes de communication et/ou socket :

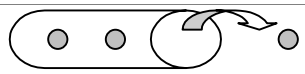
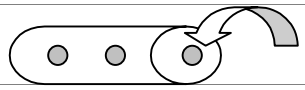


C'est une sorte de producteur / consommateur avec contrôle de flux. La longueur du tube dépend de la taille du buffer. On y fait transiter un jeton. Les lectures sont bloquantes s'il n'y a rien dans le tube, du coup la file d'attente est celle du système ou de réseau.



On peut résumer l'implémentation d'un sémaphore à l'aide d'un socket dans ce tableau :

Création d'un sémaphore	Création d'un tube de longueur inconnue	
Initialisation = n	Ecrire n « jetons » dans le tube	

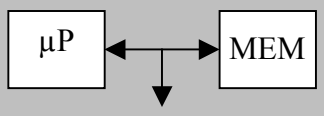
P(s)	Lire un jeton	
V(s)	Ecrire un jeton	
Destruction	Détruire le tube (fin du programme)	

Rendez-vous ADA :

Plus principalement dans les systèmes de programmation plutôt que dans les réseaux.

3.5) EXERCICE : exclusion mutuelle à variables.

On souhaite réaliser une exclusion mutuelle à l'aide de variables communes (mémoire partagée). On utilise donc des variables partagées (en lecture - écriture).



M 1- aller chercher en mémoire une instruction.
2- décodage de l'instruction
M 3- exécution de l'instruction
4- MAJ du registre

Seules opérations indivisibles : affectation d'une valeur à une variable, et test d'une valeur.

A) On se limite à 2 processus P1 et P2.

1) On essaye une solution qui utilise une variable unique (booléen) c = vrai si l'un des processus Pi se trouve en section critique, c = faux sinon. Vérifier que l'exclusion mutuelle ne peut pas être programmée (obtenir une section critique).

```
Pi :        i = 1 ou 2        j = 3-i
Contexte commun : booléen : c = faux ;          (ressource libre)
    début
    Ai :    si c alors aller à Ai          (ne marche pas si Pi est interrompu à ce moment là)
              c = vrai                     (car i et j rentrent en section critique en même temps)
                  section critique         (ce n'est pas une instruction atomique).
            c = faux
                  reste du programme
            aller à Ai
    fin
```

CELA NE MARCHE PAS

2) On essaie maintenant de construire une solution avec une variable unique t. t = i si et seulement si Pi est autorisé à s'engager en section critique. Ecrire Pi et vérifier que a) b) et d) sont satisfaites mais pas c) (voir contraintes en 2.1)

```
Entier t = 1                          (la mise à jour de t se fait en section critique)
Pi :    Contexte commun : i = 1 ou 2        j = 3-i
        Ai :    si t=j alors aller à Ai
                |  section critique i
                |  t=j
```

reste du programme i
aller à Ai

a) On a un seul processus à rentrer car la condition est modifiée en section critique.
b) Solution commune à tous les processus i.
c) Si un Pi reste bloqué en dehors de la section critique, l'autre Pi va lui donner la main, qu'il ne pourra redonner.
d) Il n'y a forcement un Pi indiqué par t.

CELA NE MARCHE PAS

3) On introduit une variable booléenne par processus.
c[i] est la variable attachée à Pi
c[i] = vrai si Pi est en section critique ou demande à y entrer
c[i] = faux si Pi est hors de sa section critique
Pi peut modifier (lire et écrire) c[i] mais seulement lire c[j]. (i≠j)

Ecrire le programme du processus Pi et vérifier qu'on peut avoir a) c) et d) ou bien b) c) et d) mais pas les 4 en même temps.

2 variables communes : j = 3-i
Contexte commun (tableau booléen c[1..2])
c[1] = c[2] = faux

Pi : Ai : si c[j] alors aller à Ai *(solution b c et d)*
c[i] = vrai *(Pb : si les 2 Pi testent en même temps, ils*
section critique i *passent tous les 2 dans la section critique).*
c[i] = faux
reste du programme
aller à Ai

Pi : Ai : c[i] = vrai *(solution a c et d)*
si c[j] alors aller à Ai
section critique
c[i] = faux
reste du programme
aller à Ai

Amélioration : Ai : c[i] = vrai
si c[j] alors
c[i] = faux
aller à Ai
section critique
c[i] = faux
reste du programme
aller à Ai

4) On complète la solution 3 en introduisant une variable supplémentaire commune t qui sert à régler les conflits à l'entrée de la section critique. En fait, c'est un indicateur de

priorité (c'est mon tour ?). En cas de conflit (si c[1] = c[2] = vrai) les 2 processus s'engagent dans une séquence d'attente où t garde une valeur constante. Pi annule sa demande si i≠t en faisant c[i] = faux. Pi attend ensuite que t passe à i avant de refaire sa demande.

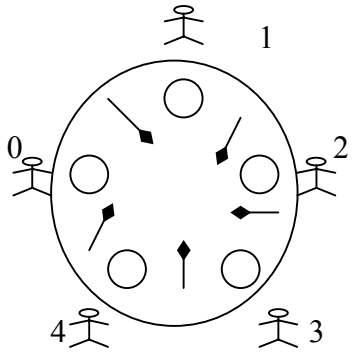
Solution de Dekker :

Contexte : on pose j = 3-i
entier t=1
tableau booléen c[1..2] avec c[1] = c[2] = faux

```
Pi :    Ai :   c[i] = vrai
                                        Li :   si c[j] alors
                si t=i      alors aller à Li
                            sinon c[i] = faux
        Bi :    si t=j  alors aller à Bi
                aller à Ai
           section critique
             t=j ;
             c[i] = faux
           reste du programme
           Aller à Ai
```

3.6) EXERCICE : Les philosophes et les spaghettis

Problème de DIJKSTRA en 1971 :



- Chaque philosophe n'a qu'une fourchette
- On ne peut pas manger avec une seule fourchette
- Action possibles du philosophe :
  1) pense
  2) faim → voudrait manger (quand il veut)
  3) mange (avec les 2 fourchettes)

1. *Définir un algorithme qui soit le même pour chaque philosophe et qui respecte ces contraintes (on repère les philosophes par i modulo 5) :*
   - *Les philosophes mangent à tour de rôle.*
   - *Il faut 2 fourchettes pour manger.*
   - *Quand on ne mange pas, on laisse la fourchette sur la table.*

2. *Ecrire l'activité* φi

3. *Que faire pour passer de l'état 1 à 3 ?*

L'état des philosophes sera représenté par un tableau de 5 variables c[0..4] avec :

- c[i]=0 lorsqu'il pense (il n'utilise pas de fourchette),
- c[i]=1 lorsqu'il voudrait bien manger mais ne peut pas par manque de fourchette,
- c[i]=2 lorsqu'il mange (il a donc les deux fourchettes).

Le passage de l'état 0 à l'état 2 n'est possible que si {c[i+1]≠2} et si {c[i-1]≠2}. *(condition « section critique » donc protégé par un sémaphore mutex).* Si cette condition n'est pas réalisée c[i]=1 et le φ se bloque. En fait, pour bloquer les philosophes, le tableau sempriv[i] sera initialisé à O donc sempriv[0..4] = 0. *(Seul le propriétaire peut se bloquer mais n'importe qui peut le débloquer).* Le sémaphore d'exclusion mutex sera initialisé à 1.

Action exécutée par le philosophe i pour demander les fourchettes :

```
B1 :  P(mutex)
      si { (c[i+1] ≠ 2) et (c[i-1] ≠ 2) }
section     alors | c[i] = 2
critique          | V(sempriv[i])        (Je me débloque de façon préventive)
          sinon c[i] = 1
      V(mutex)
      P(sempriv[i])        (Si on a pas fait le V(sempriv), on se bloque juste en attendant notre tour)
```

Action exécutée par le philosophe i qui a fini de manger (provisoire) et rend les fourchettes :

Le passage de l'état c[i]=2 à 0 entraîne le réveil des philosophes i+1 et i-1 à deux conditions :

- ils ont l'intention de manger donc c[i-1] = 1 ou/et c[i+1] = 1,
- ils ont l'autre fourchette donc c[i+2] ≠ 2 ou/et c[i-2] ≠ 2.

```
B2 :  P(mutex)
      c[i] = 0
      si { (c[i+1] = 1 ) et (c[i+2] ≠ 2) }
section     alors |c[i+1] = 2
critique          |V(sempriv[i+1])
      si { (c[i-1] = 1 ) et (c[i-2] ≠ 2) } ← (Pour réduire la longueur de la section critique, on peut
          alors |c[i-1] = 2                   mettre V(mutex) et P(mutex) avant cette ligne)
                |V(sempriv[i-1])
      V(mutex)
```

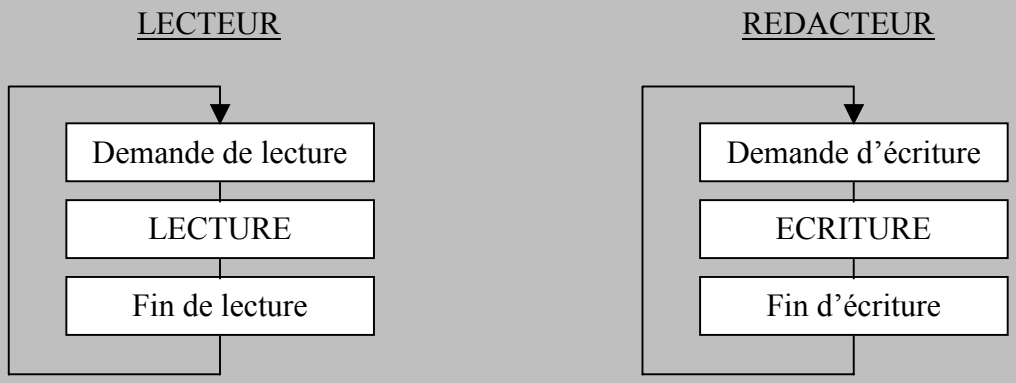
Remarque :

La suite (0,3) ; (0,2) ; (4,2) ; (0,2) ; (0,3) ; … est possible (le 1 ne mangera donc jamais) car on ne garantit pas que tous les philosophes mangeront. Pour cela, il faudrait introduire des sémaphores pour la gestion de la file d'attente. Par exemple, en FTP, on essaye de répartir la charge de la bande passante du réseau.

3.7) EXERCICE : Les lecteurs et les rédacteurs

Problème de COURTAS en 1971 : « modèle pour des systèmes partageables »

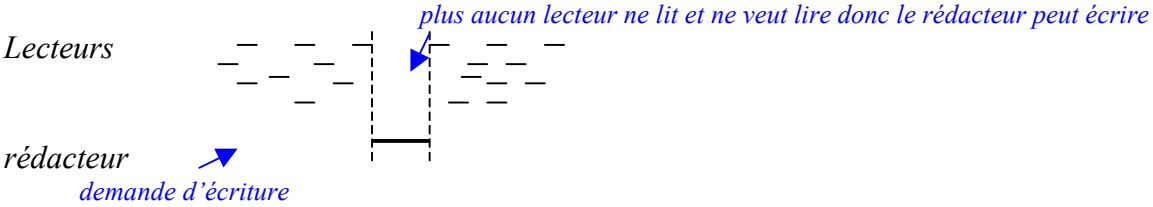
On considère n processus (n entier >2) parallèles indépendants, divisés en deux : lecteurs et rédacteurs. Ces processus peuvent se partager une ressource unique : le fichier dans lequel on lit ou on écrit). Ce fichier peut être lu simultanément par plusieurs lecteurs mais écrit par un seul rédacteur (exclusion mutuelle → aucun autre rédacteur et pas de lecteur).



Ecrire ces procédés à l'aide de sémaphores dans 4 cas particuliers :

1er cas) Priorité des lecteurs sur les rédacteurs, sans réquisition (variable d'état).

C'est uniquement lorsqu'il n'y a plus de lecteur en train de lire que le rédacteur peut écrire. Soulignons que les lecteurs ont donc toujours la priorité sur les rédacteurs.



Règles qui régissent les lecteurs et rédacteurs :

Il va falloir mettre en place des systèmes de contrôle. Le principe serait de protéger la ressource critique de la même façon en écriture et en lecture. Mais cela n'autorise qu'un seul lecteur à la fois. ! On décide donc d'utiliser :

- entier nl = 0 *(nombre de lecteurs qui « occupent le fichier »),*
- sémaphore Rmutex = 1 *(file d'attente rédacteur),*
- sémaphore w = 1 *(contrôle d'accès au fichier),*
- sémaphore lmutex = 1 *(protège la variable nl).*

LECTEUR

P(lmutex) *(on protège la variable nl)*

```
    Si nl = 0 alors P(w)      (s'il n'y avait pas de lecteur présent : on protège l'accès rédacteur fichier)
    nl = nl + 1               (on incrémente le nombre de lecteurs qui occupent le fichier)
V(lmutex) (on libère la variable nl)
Lecture                       (on fait la lecture dans le fichier)
P(lmutex)                     (on protège la variable nl)
    nl =nl -1                 (on décrémente le nombre de lecteurs qui occupent le fichier)
    Si nl = 0 alors V(w)      (s'il n'y a plus de lecteur présent : on libère l'accès rédacteur au fichier)
V(lmutex)                     (on libère la variable nl)
```

REDACTEUR

```
P(Rmutex)                     (on protège la file d'attente rédacteur)
    P(w)                      (on protège l'accès au fichier)
        Ecriture              (on fait l'écriture dans le fichier)
    V(w)                      (on libère l'accès au fichier)
V(Rmutex)                     (on libère la file d'attente rédacteur)
```

Remarque : Dans le cas présent, la priorité des lecteurs n'est évidemment pas respectée.

2ième cas) Priorité des lecteurs sur les rédacteurs (sans sémaphore privé).

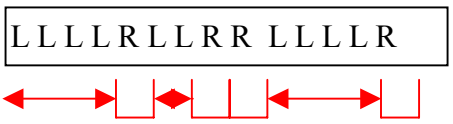
C'est à dire une priorité des lecteurs sur les rédacteurs si et seulement si un lecteur occupe déjà le fichier. Il suffit donc de faire la même chose que dans le premier cas mais sans sémaphore privée (pas de variable rmutex). Ce qui donne pour le processus du rédacteur :

REDACTEUR

```
P(w)                          (on protège l'accès au fichier)
    Ecriture                  (on fait l'écriture dans le fichier)
V(w)                          (on libère l'accès au fichier)
```

3ième cas) Priorité égale pour les lecteurs et les rédacteurs.

On a toujours n > 1 lecteurs simultanés et un seul rédacteur à la fois. Si un lecteur a le fichier, les lecteurs suivants y accèdent en même temps jusqu'à l'arrivée d'un rédacteur, les lecteurs suivants sont mis en attente. Il faut donc une file d'attente commune (Rmutex par ex).



LECTEUR

```
P(Rmutex)                     (on protège la file d'attente commune rédacteur - lecteurs)
    P(lmutex)                 (on protège la variable nl)
```

```
    Si nl = 0 alors P(w)      (s'il n'y avait pas de lecteur présent : on protège l'accès rédacteur fichier)
    nl = nl + 1               (on incrémente le nombre de lecteurs qui occupent le fichier)
  V(lmutex) (on libère la variable nl)
V(Rmutex)                     (on libère la file d'attente commune rédacteur - lecteurs)
Lecture                       (on fait la lecture dans le fichier)
P(lmutex)                     (on protège la variable nl)
  nl =nl -1                   (on décrémente le nombre de lecteurs qui occupent le fichier)
  Si nl = 0 alors V(w)        (s'il n'y a plus de lecteur présent : on libère l'accès rédacteur au fichier)
V(lmutex)                     (on libère la variable nl)
```

REDACTEUR

```
  P(Rmutex)                   (on protège la file d'attente commune rédacteur - lecteurs)
    P(w)                      (on protège l'accès au fichier)
      Ecriture                (on fait l'écriture dans le fichier)
    V(w)                      (on libère l'accès au fichier)
  V(Rmutex)                   (on libère la file d'attente commune rédacteur - lecteurs)
```

4ième cas) Priorité des rédacteurs sur les lecteurs.

Dès qu'un rédacteur demande l'accès au fichier, il doit l'obtenir dès que possible. S'il arrive après les demandes de lecteurs, il est prioritaire (en fait, la file d'attente des rédacteurs est prioritaire sur celle des lecteurs). C'est à dire que l'on bloque toutes les nouvelles demandes mais on laisse les processus actifs se terminer.

Les lecteurs L passent leur tour !

On va avoir les variables supplémentaires suivantes :

- entier nr = 0 *(nombre des rédacteurs),*
- entier nl = 0 *(nombre de lecteurs qui « occupent le fichier »),*
- sémaphore w = 1 *(contrôle d'accès au fichier),*
- sémaphore mutex1 = 1 *(protège la variable nl)*
- sémaphore mutex2 = 1 *(protège la variable nr)*
- sémaphore r = 1 *(bloque les lecteurs : contrôle la priorité des rédacteurs),*
- sémaphore mutex3 = 1 *(garantie que l'on a qu'un lecteur à la fois pour priorité rédacteur c'est à dire qu'il empêche les lecteurs de réserver leur place dans r).*

LECTEUR

On aura un seul lecteur à la fois qui réservera le fichier pour garantir la file des rédacteurs.

```
P(mutex3)                  (empêche les lecteurs de réserver leur place dans r)
   P(r)                    (on bloque pour avoir un seul processus à la fois)
      P(mutex1)            (on protège la variable nl)
         nl = nl + 1       (on incrémente le nombre de lecteurs qui occupent le fichier)
         Si nl = 1 alors P(w)  (s'il n'y avait pas de lecteur présent : on protège l'accès rédacteur fichier)
      V(mutex1)            (on libère la variable nl)
   V(r)                    (on libère la file d'attente commune rédacteur - lecteurs)
V(mutex3)                  (on libère la réservation de place dans r des lecteurs)
Lecture                    (on fait la lecture dans le fichier)
P(mutex1)                  (on protège la variable nl)
   nl =nl -1               (on décrémente le nombre de lecteurs qui occupent le fichier)
   Si nl = 0 alors V(w)    (s'il n'y a plus de lecteur présent : on libère l'accès rédacteur au fichier)
V(mutex1)                  (on libère la variable nl)
```

REDACTEUR

```
   P(mutex2)               (on protège la variable nr)
      nr = nr + 1          (on incrémente le nombre de rédacteur dans la file d'attente)
   si nr = 1 alors P(r)    (on bloque les lecteurs)
   V(mutex2)               (on libère la variable nr)
   P(w)                    (on protège l'accès au fichier)
      Ecriture             (on fait l'écriture dans le fichier)
   V(w)                    (on libère l'accès au fichier)
   P(mutex2)               (on protège la variable nr)
      nr = nr - 1          (on décrémente le nombre de rédacteur dans la file d'attente)
   si nr = 0 alors V(r)    (on libère les lecteurs)
   V(mutex2)               (on libère la variable nr).
```

3.8) EXERCICE : Les feux de circulation

Soit un carrefour à quatre directions sans changement de direction : c'est à dire qu'aucune voiture ne tourne (elles traversent le carrefour tout droit).

a) Toute voiture qui se présente à un carrefour doit le franchir (pas de panne au milieu).

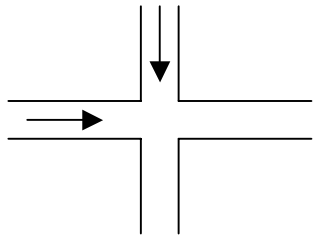
b) Chaque feu passe du vert au rouge alternativement, chaque couleur est maintenue pendant un temps fini. Rouge dans un sens, vert dans l'autre

c) A un instant donné le carrefour ne contient des voitures que d'une seule voie.

d) Modélisation : processus parallèle ou coopérants.

Il y a un processus « Changement » qui gère la commande des feux et un processus par voiture. La traversée du carrefour par une voiture de la voie i ∈{1,2} est représentée par la procédure « traversée (i) ». Il faut donc écrire les processus changement, traversée 1 et traversée 2 selon les cas décrit. On considérera que P = prendre, et V = libérer.

1er Cas : le carrefour ne peut contenir qu'une voiture au plus à la fois.



Attention ! Il faut absolument que la voiture soit sortie du carrefour avant de donner le feu vert aux autres.

On a donc deux files d'attentes à programmer. Il y a une file d'attente par feu où la voiture de la file attend que la précédente voiture soit passée (carrefour libre).

Sémaphores feu1 = 0, feu2 = 1 *(autorisations de s'engager dans le carrefour 1 ou 2*
mutex 1, mutex 2

TRAVERSEE 1

```
P(mutex1)        (on protège la variable)
  P(feu1)        (on protège l'accès au feu 1)
    Traversée 1  (on fait la traversée direction 1)
  V(feu1)        (on libère l'accès au feu 1)
V(mutex1)        (on libère la variable).
```

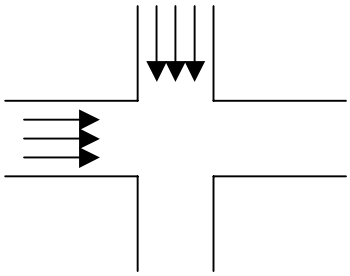
TRAVERSEE 2

```
P(mutex2)
  P(feu2)
    Traverse 2
  V(feu2)
V(mutex2)
```

CHANGEMENT

```
Attendre (30)          (commuter les feux)
  P(feu1)
    Temporisation de sécurité
  V(feu2)
Attendre (30)          (commuter les feux)
  P(feu2)
    Temporisation de sécurité
  V(feu1)
```

2ième Cas : le carrefour peut contenir plusieurs voitures à la fois.



Donc le carrefour n'est plus une ressource critique puisqu'il peut être utilisé par **k** voitures en même temps.

Sémaphores : feu1 = 1 *(autorisation de s'engager dans le carrefour 1)*
feu2 = 0 *(autorisation de s'engager dans le carrefour 2)*

```
                mutex1, mutex2 = k    (autoriser k voitures max. dans le carrefour)
                mutex = 1             (protection de la variable n)
                w = 1                 (w = wait = détention du carrefour)
Entier n = 0    (nombre de voitures qu'il reste dans le carrefour avant de pouvoir allumer le feu)
```

TRAVERSEE 1 (même chose pour Traversée 2)

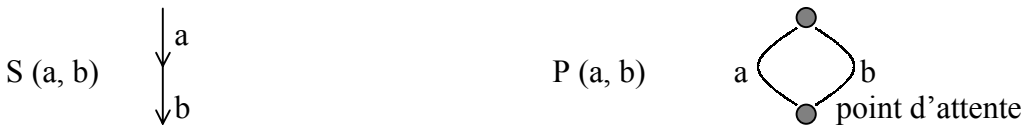
```
P(mutex1)                  (protège le nombre k de voitures max. dans le carrefour)
  P(feu1)                  (protège l'engagement dans le carrefour 1)
  P(mutex)                 (protection de la variable n)
    n = n +1               (incrémentation du nombre de voiture : je rentre dans le carrefour)
    si n = 1 alors P(w)    (si je suis le seul dans le carrefour, je protège la détention du carrefour)
  V(mutex)                 (libération de la variable n)
  V(feu1)                  (libération de l'engagement dans le carrefour 1)
  Traversée 1
  P(mutex)                 (protection de la variable n)
    n = n –1               (décrémentation du nombre de voiture : je sort du le carrefour)
    si n = 0 alors V(w)    (s'il n'y a plus de voiture engagée dans le carrefour, je libère sa détention)
  V(mutex)                 (libération de la variable n)
V(mutex1)                  (libère le nombre k de voitures max. dans le carrefour)
```

CHANGEMENT

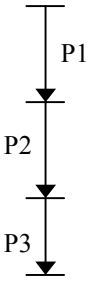
On n'a pas besoin de se protéger contre les dead lock puisqu'on a qu'un processus changeant.

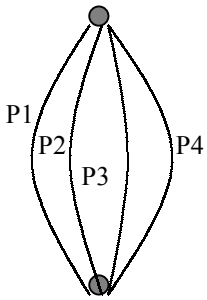
```
Attendre (30 sec)
  P(feu1)                  (protège l'engagement dans le carrefour 1)
  P(w)                     (protège la détention du carrefour)
    V(feu2)                (libération du carrefour 2)
  V(w)                     (libère la détention du carrefour)
  Attendre (45 sec)
  P(feu2)                  (protège l'engagement dans le carrefour 2)
    P(w)                   (protège la détention du carrefour)
    V(mutex2)
    V(feu1)                (libération du carrefour 1)
  V(w)                     (libère la détention du carrefour)
```

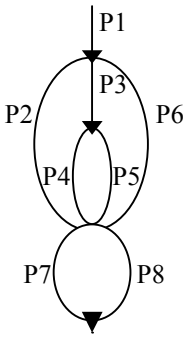
3.9) EXERCICE : Coopération de tâches.

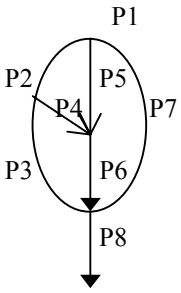


1) Exprimer les cas a, b ,c et d avec S et P.









a) S (P1, S (P2, S (P3, P4)))
b) P (P1, P (P2, P (P3, P4)))
c) S {P1, S [P (P2, P {S [P3, P (P4, P5)], P6}), P (P7, P8)]}
d) Impossible ! ! ! car on n'a rien pour représenter :

2) Exprimer S et P avec des sémaphores.

S (P1, P2) se réécrit en

P1

P2

Pour S et P, mettre en place la synchronisation :

encaspuler P1 et P2,
démarrer P1 et P2 encaspulés.

Sémaphore sérieP1P2 = 0

P1	*P (sérieP1P2)*
V (sérieP1P2)	*P2*

S (P1, P2) = { sémaphore sérieP1P2 ; spaw (P1, V (sérieP1P2)) ; spaw (P (sérieP1P2), P2)}

Avec spaw (P1) : démarre un processeur à partir du programme P1.

P (a, b) = { sémaphore a=0 ; sémaphore b=0 ; spaw (a ; V(b) ; P(a)) ; spaw (b ; V(a) ; P(b)) }

3) Exprimer d avec des sémaphores.

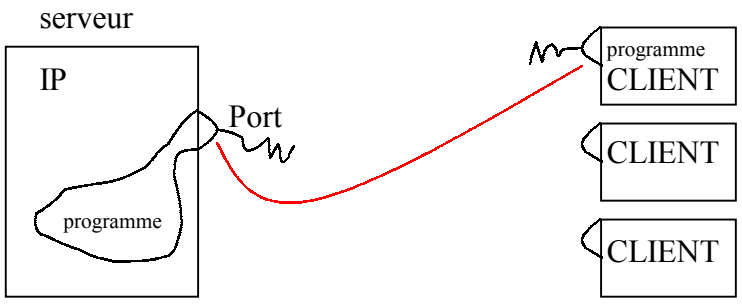
Soit les sémaphores : x3, x4, x5, x60, x61, x71, x72, x80, x81 = 0.

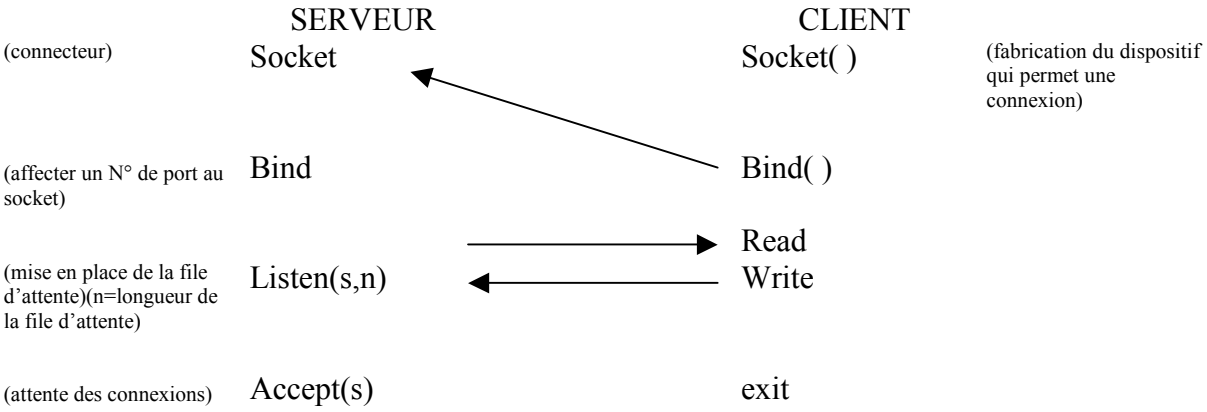
				~~P(x61)~~
	P(x3)			P(x60)
P2	P3	P4	P5	P6
V(x3)	V(x71)	V(x80)	V(x81)	V(x72)
V(x4)	P(x72)	P(x81)	P(x80)	P(x71)
		V(x60)	~~V(x61)~~	
			Facultatif car même rôle que V(x60)	

IV. TRAVAUX PRATIQUES SOCKET (Corrigé)

4.1) Rappels sur les SOCKETS

- Modèle client-serveur et établir des canaux de communication entre les clients et le serveur.
- Inventé par UNIX BSD.
- S'appuie sur TCP ou UDP, appletalk, IP, X25…
- Communication bidirectionnelle.
- Transmission des données paramétrables : flots, paquets, datagrammes.
-





4.2) Test des ports d'une machine

Ce programme teste les ports qui répondent sur une machine (il utilise des sockets).

```
#define SERVER "rantanplan'
#define PORT 1500
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define ZERO(obj) (bzero((char*)&(obj), sizeof(obj)))      (macro procédure qui met des 0 binaires..
#define VAL(t) (t.tv_sec+t.tv_usec/1000000.0)                                  ..dans la zone obj)

main (int argc, char *argv[ ])          (argc : nbre de paramètres passés ; argv : tableau de paramètres)
{   int i;
    int s;
    struct sockaddr_in sin;
    struct hostent *p;

    ********** Création de socket **********
    for  (i=0;i<10000;++i)
        { if ((s=socket (AF_INET, SOCK_STREAM,0)) < 0)      (appel système pour créer un socket)
            {   perror ("pas moyen de creer le socket");          (SOCK_STREAM : type de socket)
                exit (l) ; }                                      (AF_INET : on veut accéder à TCP/IP)
                                                                  (si <0 : erreur ça n'a pas marché !)
        **** Initialisation de la structure de communication ***
         ZERO (sin) ;                                  (Appel à la macro procédure définie dans défine)

        **** Remplissage de la structure de communication INTERNET, on utilise le résolveur
```

```
        pour passer du nom symbolique a l'adresse IP  ***
      if ((p=gethostbyname((argc= =2)?argv[l]:SERVER)) = = 0 )   ( 2 arguments : nom serveur)
        { fprintf (stderr, ‘‘pas moyen de resoudre le nom\n’’) ;        ( 1 argument : nom machine)
          exit(4) ;  }                          (appel système pour récupérer le nom de la machine)

     **** Copie du numéro IP  ***
      bcopy (p->h_addr, (char *)&sin.sin_addr, p->h_length);     (recopie le n°IP dans la zone sin)
      sin sin_family=AF_INET;
      sin. sin_port=htons (i)                          (htons : uniformise la présentation d’un entier)
                                    (pour s’adapter à n’importe quel système (couche présentation)
     ****  Connexion au serveur  ***
      if ( connect(s, (struct sockaddr *)&sîn, sizeof(sin)) >= 0) (se connecte s : adresse du socket)
         printf(‘‘%d\n’’, i) ;                                          (on mémorise le nom du port)
      close (s);
     }
}
********** On obtient le résultat suivant :  **********
exécution:

rantanp1an:~/slide/sys4>  ./scan
7
9
11
13
15
19
21
23
37
79
111
…
```

4.3) Serveur de synchronisation d’horloges

Ce programme illustre l'utilisation de sockets. Il est le serveur d'un système de synchronisation des horloges sur différentes machines.

```
#define PORT 1500
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb h>
#define ZERO(obj) (bzero((char*)&(obj),sizeof(obj)))

main ( )
{
```

```
int s;
struct sockaddr_in Sm;

********** Création de socket **********
if  ((s=socket(AF_INET, SOCK_STREAM, 0)) < 0)
  { perror ("pas moyen de creer le socket");
    exit(l) ; }

********** Initialisation de la structure de communication **********
ZERO (sin)

********** Remplissage de la structure de communication INTERNET **********
sin. sin_family=AF_INET;
sin.sin_port=htons (PORT);
if  (bind (s, (struct sockaddr *)&sin, sizeof(sin)))          (on raccorde le socket au système)
  { perror ("pas moyen de se connecter au port");
    exit(2) ; }

********** On indique 10 connexions possibles en attente **********
if (listen (s,10) < 0 )                          (on indique un maximum de dix connexions en attente)
  { perror ('pas moyen  de prévoir 10 connexions en attente");
    exit(2) ; }

********** Boucle d'attente **********
while (1)                                                    (boucle infinie comme while(vrai))
  { int g, n;
    char  buf [40];
    if ((g=accept (s,0,0)) < 0)
       { perror ("erreur dans la réception d'un appel");
         exit(3) ; }
    *** On a franchi l'accept maintenant g est nouveau descripteur pour la comm. ***
    if (fork ()) close(g);                    (fork : fourche qui dédouble en deux parties le système)
    else { close (s) ;
           while (1)
             { read (g, buf, l);
               if (buf[0]= ='s') write (g, "o", l);    (si on a s : c'est une synchronisation et on renvoit 0)
               else { struct timeval t;                (récupération du temps)
                      struct timezone z;
                      gettimeofday(&t, &z);
                      write (g, (char *)&t, sizeof(t));
                      sleep(2) ;
                      exit(0) ;
                    }
             }
         }
  }
}
```

4.4) Client de synchronisation d'horloges

Ce programme illustre l'utilisation des sockets. Il est le client d'un système de synchronisation des horloges sur différentes machines.

```
#define SERVER "rantanplan"
#define PORT 1500
#include <stdio.h>
#include <sys/time.h>
#include <unistd.h>
# include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#define ZERO(obj) (bzero ((char*)&(obj),sizeof (obj)))
#define VAL(t) (t.tv~sec+t.tv~usec/l000000.0)

main(int argc, char *argv[]
{
   int s;
   struct sockaddr_in sin;
   struct hostent *p;

   ********** Création de socket **********
   if((s=socket (AF_INET, SOCK_STREAM, 0)) < 0)
     { perror ("pas moyen de créer le socket") ;
       exit(l) ; }

   ********** Initialisation de la structure de communication **********
   ZERO (sin)

   **********  Remplissage de la structure de communication INTERNET, on utilise le
                résolveur pour passer du nom symbolique a l'adresse IP **********
   if ((p=gethostbyname((argc= =2)?argv[l]:SERVER)) = = 0 )      ( 2 arguments : nom serveur)
     { fprintf (stderr, ‘‘pas moyen de resoudre le nom\n’’) ;     ( 1 argument : nom machine)
       exit(4) ; }                                  (appel système pour récupérer le nom de la machine)
   }

   **** Copie du numéro IP ***
   bcopy (p->h_addr, (char *)&sin.sin_addr, p->h_length);      (recopie le n°IP dans la zone sin)
   sin sin_family=AF_INET;
   sin. sin_port=htons (i)                                     (htons : uniformise la présentation d'un entier)
                                          (pour s'adapter à n'importe quel système (couche présentation)
   ****  Connexion au serveur ***
   if ( connect(s, (struct sockaddr *)&sin, sizeof (sin)) >= 0)    (se connecte s : adresse du socket)
       { perror("pas moyen de se connecter au serveur de temps");
         exit(5) ; }

   ********** On a réussi a se connecter. On étalonne la vitesse de la connexion en faisant
   une moyenne sur dix synchronisation **********
```

```
    { float t=0;
      int i;
      char buf[l00];
      struct timeval tl,t2,ref;
      struct timezone zl,z2;

      for (i=0;i<l0;++i)          (on mesure le temps d'aller-retour pour synchroniser le réseau : dix fois)
        { gettimeofday (&tl, &zl);
        | write (s, "s", l) ;
        | read (s, buf, l) ;
        | gettimeofday (&t2, &z2);
        | t+=VAL (t2)-VAL (tl) ;
        }
      t/=l0.0;  (fait la moyenne pour avoir le délai du décalage horaire et mettre en synchro par settimeofday)
      printf (délai moyen : %f (seconde)\n", t);
      **** Maintenant, on récupère l'horloge du serveur  ***
      write (s, "t", l) ;
      read (s, (char *)&ref, sizeof (struct timeval)) ;
      gettimeofday(&tl, &zl);
      printf ("temps du serveur  : %.3f\n",VAL (ref)+t);
      printf ("temps local       : %.3f\n",VAL (tl)) ;
    }
}
```

********** *On obtient le résultat suivant :* **********

exécution:

```
rantanplan:~/slide/sys4> . /timeclient
pas moyen de se connecter au serveur de temps: Connection refused
rantanplan:~/slide/sys4> . /timeserver

Suspended
rantanplan:~/slide/sys4> bg
[1]      ./timeserver &
rantanplan: ~/slide/sys4>
rantanplan: ~/slide/sys4>
rantanplan: ~/slide/sys4> ./timeclient
delai moyen : 0.001172 (seconde)
temps du serveur : 843685406.588
temps local      : 843685406.588
rantanplan:~/slide/sys4> ./timeclient
delai moyen : 0.001157 (seconde)
temps du serveur : 843685410.728
temps local      : 843685410.728
rantanplan: /slide/sys4>
```

SNA
&
OSI / DSA

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

SNA ET OSI/DSA

Architectures propriétaires

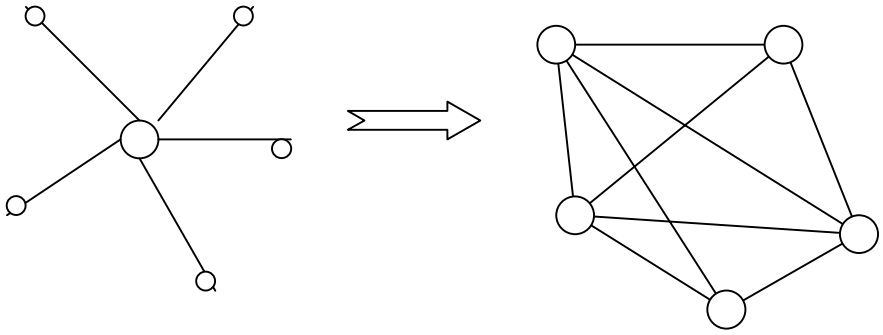
SNA : System Network Architecture
Développé par IBM, antérieur à la norme OSI.

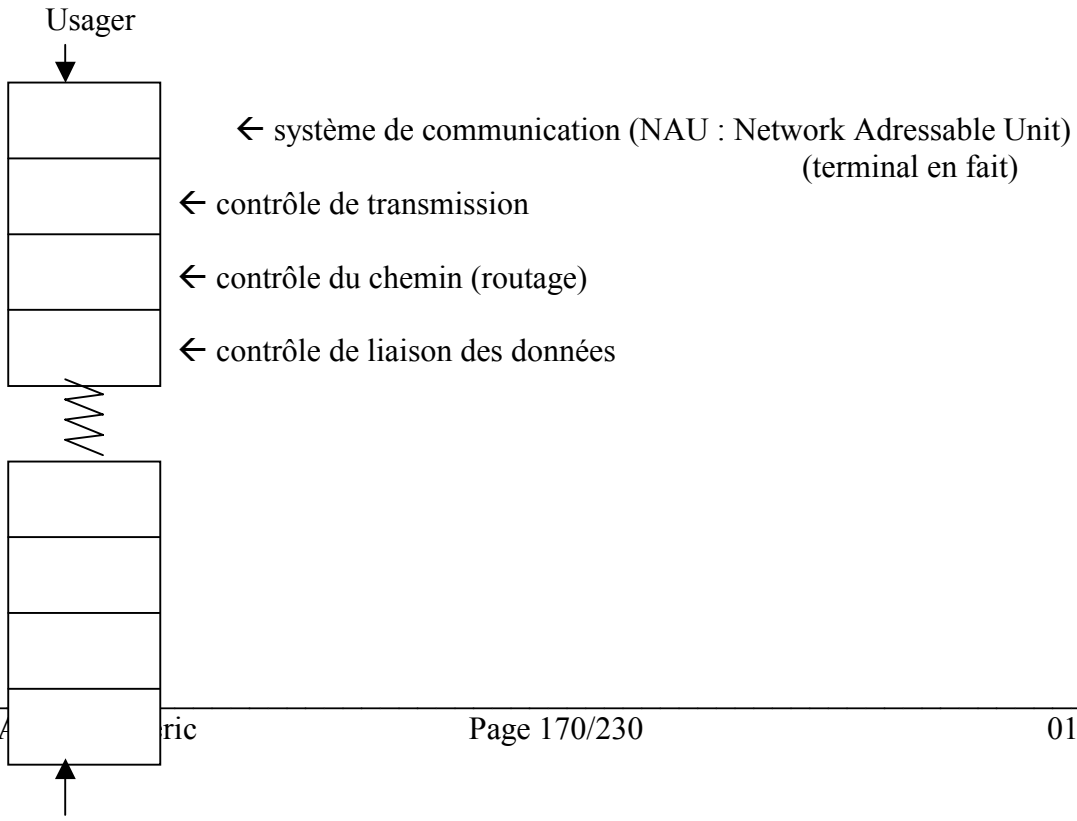
DSA : Distributed System Architecture
Développé par BULL, respecte le modèle OSI.

I. SNA

La plus utilisée dans le monde il y a encore 2 ou 3 ans.

1974 : « architecture » autour de laquelle s'articulerait toutes les structures matérielles et logicielles du monde.





Usager

- Les NAU sont les seules entités capables d'établir une connexion :
  → LU (logical unit) : présentation et session.
  → SSCP (System Service Control Point) : gère ressource d'un réseau ou d'un sous réseau.
  → PU (physical unit).
- Les usagers communiquent à travers des LU.

- Session LU – LU ou session LU : connexion entre deux usagers.

- 2 types de réseaux SNA :
  → SAN : Sub Area Networ (ancien).
  → APPN : Advanced Peer to Peer Networking

- Le responsable des ressources physiques et logiques du réseau est le centre directeur SSCP.

1.1) Réseau SAN

- Le SSCP (Mainframe) gère les ressources du domaine ainsi que toutes les connexions (architecture centralisée, en étoile).

- Les PU sont adaptées au besoin
  - PU5 : SSCP
  - PU4 : frontal
  - PU2 : contrôleur de grappe
  - PU1 : terminaux

- 5 types de LU : LU0, LU1, LU2, LU3, LU4, LU6.1, LU6.2

session LU

Demande d'un PLU (Primary Logical Unit) au SSCP, le SSCP vérifie et demande au SLU (Secondary logical unit) la connexion. Etablissement (quand les deux sont d'accord).

1.2) APPN (non hiérarchique)

- Le SSCP n'est pas consulté.
- Chaque nœud établi directement la communication.
- Nécessité matérielle et logicielle de PU2.1 et LU6.2.
- Mode de communication entre LU défini par APPC (advanced Program to Program Communication).
- APPC désigne un ensemble de fonctions, formats, protocoles pour les coordinations des communications entre programmes de différents sites.

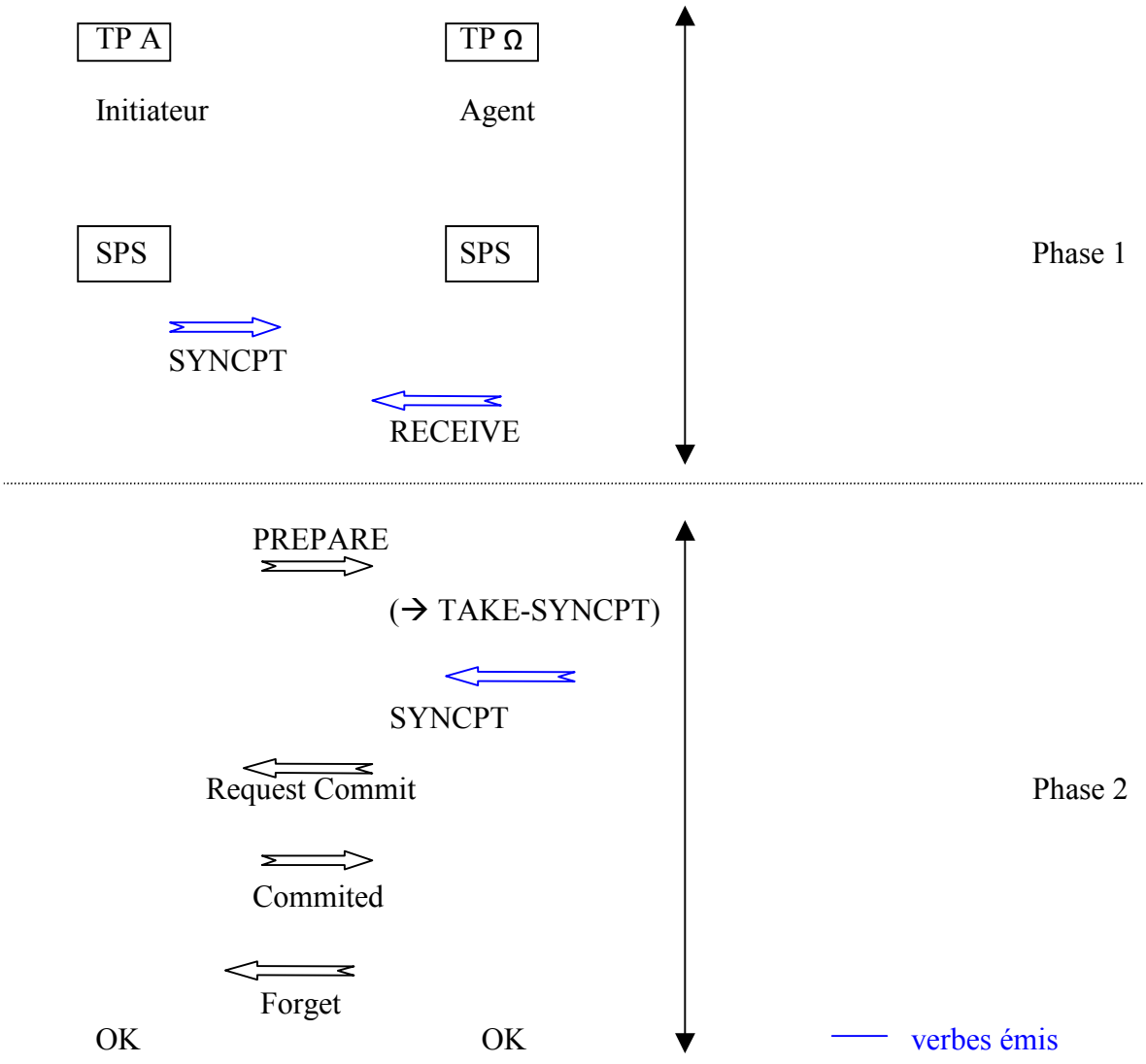
Les session LU dans APPN

Les TP (Transaction Program) communiquent quand les deux LU sont en session. Le dialogue entre TP est appelé « conversation ».

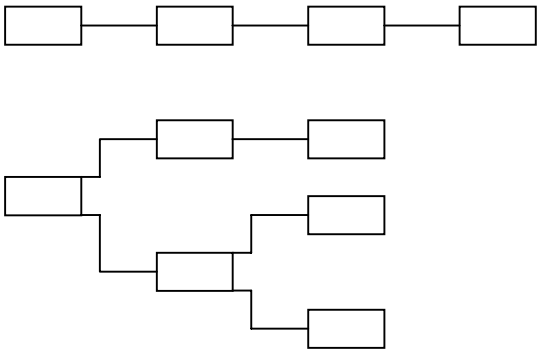
1.3) Protocole de la LU6.2

- Frontière : interface de l'utilisateur.
- Communication par messagerie (« verbe »).
- L'émission d'un verbe est bloquante.
- 2 modes de synchronisation ;
  → synchronisation simple
  CONFIRM
  CONFIRMED
  → points de synchronisation
  SYNCPT

SPS : Service des Points de Synchronisation



Structure distribuée



Conclusion

SNA : modèle très simple, protocoles synchrones (rustique, robuste).

II. DSA

- Dernière version des systèmes repartis proposes par BULL.
- Conforme au découpage du modèle OSI de la norme ISO
- Juin 1976.
- Construit autour des datanets : famille de machines réseau.

1) 1976 → 1985 : DSA

- Protocoles conformes à OSI dans les couches basses.
- Propriétaire dans les couches hautes.

2) 1985 → 1989 : OSI/DSA

- Compatible avec ce qui à été fait avant.

3) 1989 →

- Les normes ISO des 7 couches sont progressivement intégrées dans l'architecture.

2.1) Terminologies dans OSI/DSA

- Réseau primaire : ensemble de dispositifs d'interconnexion qui respectent OSI/DSA.
- Réseau secondaire : tout ce qui est non conforme à OSI/DSA. L'accès à OSI se fait par passerelle.
- Activité : programme ou activité utilisateur qui utilises des ressources.
- Point d'accès : frontière entre une activité et le réseau. (login + password sur un terminal).
- Connexion logique : établissement d'une voie de communication point à point.
- Chemin : établissement d'une série de connexion physique point à point pour arriver à établir une connexion logique.
- Boite aux lettres : permet aux applicatifs de communiquer. Une boite aux lettres à un nom unique.

2.2) Fonctionnalités de OSI/DSA

- Conforme au modèle OSI de la norme ISO.
- 3 couches basse = OSI .
- Couche transport : ajoute une QOS (Quality Of Service) aux utilisateurs .
- Couche session : élément clé de l'architecture. Fonctionne par boite aux lettres (qui peuvent être reparties). Guide et gère les dialogues.
- Couche présentation : mise en forme des données (syntaxe ASN-1).
- Couche application : comprend des programmes d'application et les sous systèmes qui coordonnent les applications. FTAM, X400, X500, transfert de document ODA (Office Document Architecture), EDI et EDIFACT.
  Peut accueillir des applications non OSI par l'intermédiaire des API (Application Program Interface). Egalement protocole DCM (Distributed Computing model) : généralise la transparence des communications entre les applications.

2.3) Les datanets

Famille de processeurs de réseau (dédié BULL).

2.4) Conclusion

Misé dès le départ sur l'ouverture des réseaux , évolution par remplacement des protocoles propriétaires par équivalent OSI.

EXERCICE

Q : on suppose que l'on a SNA hiérarchique .

1) Un usager connaît il la position de son correspondant ?
2) Une LU connaît elle la position de la LU avec laquelle elle désire communiquer ?

R :

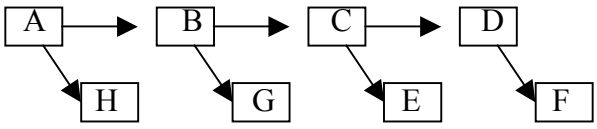
1) Non. L'usager ne voit que la LU qui lui permet de communiquer.
2) Au départ non, mais à terme oui car il y a entre les deux un arbitre (SSCP) qui lui permet de le savoir.

EXERCICE

Une structure de transaction.
La conversation utilise des points de synchronisation à l'aide du verbe SYNCPT.

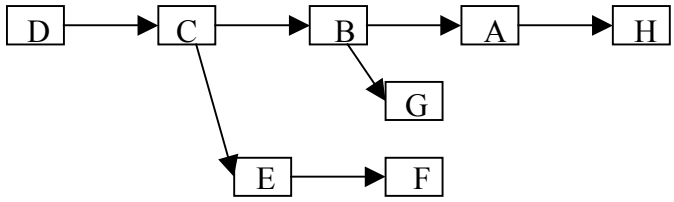
1) représenter l'arbre de synchronisation quand D est l'initiateur de la synchronisation.

Arbre des transactions :



2) Peut il y avoir 2 initiateurs de synchronisation dans une structure de transaction quelconque ?

1)



2) non car cela ne fonctionne pas → interblocage.
Le verbe et bloquant dans ce protocole alors il seraient deux à attendre la réponse avant de répondre à leur tour.

ADMINISTRATION DES RESEAUX

ADMINISTRATION DE RESEAUX.....

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

ADMINISTRATION DE RESEAUX

Norme ISO 7498-4 (OSI) : norme de gestion des réseaux.

I. PROBLEMATIQUE

- Fiabilité du réseau de communication
  → qualité de services (dimensions fonctionnelles, organisationnelles et technico-économiques).

  - Dimension organisationnelle :
    - Niveau opérationnel : activité de « routine » des opérateurs pour la surveillance des alarmes, configuration des routeurs, information des utilisateurs, rétablissement de liaisons, sauvegardes de la comptabilité. Taches quotidiennes et répétitives.
    - Niveau tactique : activités liées à l'évolution du réseau à moyen terme. Achats de matériel, mesures de saturations.
    - Niveau stratégique : schéma directeur (objectifs à long terme).
  - Dimension technico-économique :
    - Sous traitance ou non.
    - Fournisseur unique ou non.
    - Politique de sécurité du réseau, interconnexion, modèle d'architecture.

II. LA GESTION VUE PAR L'ISO

2.1) Les 5 domaines fonctionnels :

1) Gestion de la configuration :
   Gestion des noms des ressources *(ex : URL, noms des machines)*, gestion des paramètres des matériels, collecte des matériels du réseau, collecte des informations sur la topologie et les changements d'état des composants, routage, reconfigurer les ressources...

2) Gestion des performances : *« évaluer en permanence »*
   Evaluer les performances pour agir sur la qualité de service (QoS), collecte des indicateurs de mesure et des performances, gestion des journaux d'historique (aide au dimensionnement), surveillance des latences temps de transit, débit réel...

3) Gestion des fautes : *SNMP*

Gestion des pannes, détection des pannes, actions manuelles ou automatiques, maintenir la qualité de service offerte, collecte des anomalies (alarmes) → journalisation, activation de tests pour localiser l'origine des pannes.

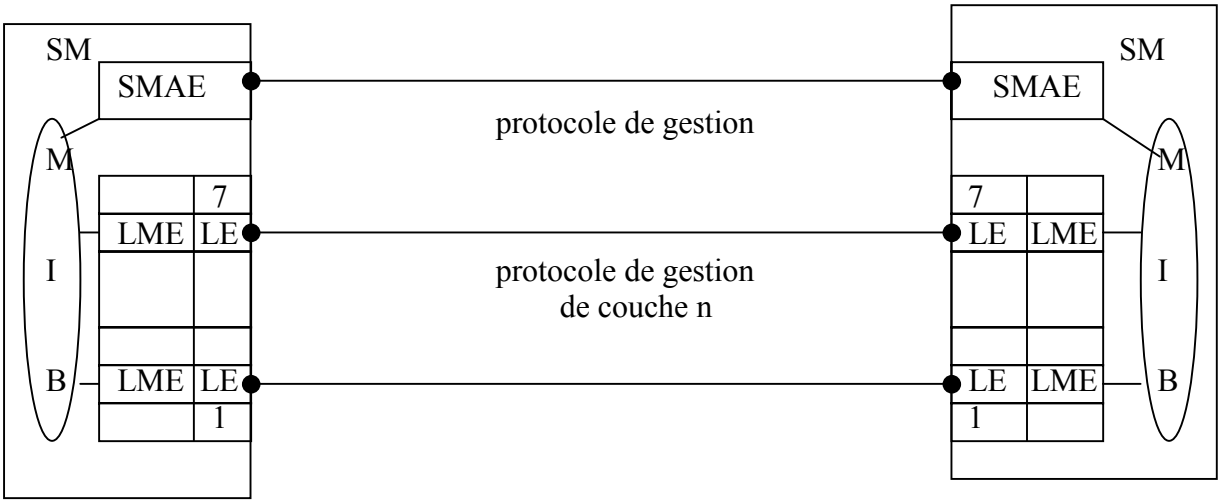
4) Gestion des coûts :
Compatibilité des flux dans le but de les facturer à leur utilisateur ou bien pour faire des statistique (*ex :octets d'entrées limités)*.

5) Gestion de la sécurité :
Sécurité d'intégrité des informations qui circulent sur le réseau : fiabilite des informations, confidentialité, authentification, non répudiation. Attention à la compatibilité CNIL ! !

2.2) CADRE ARCHITECTURAL DE LA GESTION OSI

Norme ISO 10040 / CCITT X701. « System Management Overview »

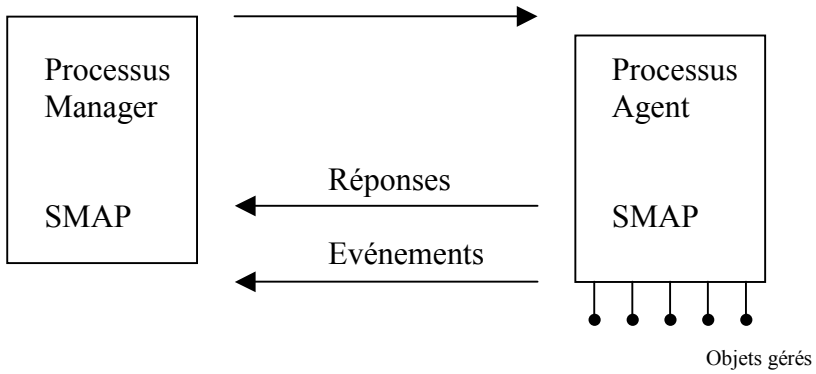
- Structure de la gestion
- Fonctionnalité de la gestion
- Base des informations



SMAE : System Management Application Entity
MIB : Management Information Base
LME : Mayer Management Entity
LE : Layer Entity

Modèle conceptuel client-serveur (ou agent manager) :

Système de gestion — Commandes — système géré



LE MODELE D'INFORMATION DE GESTION

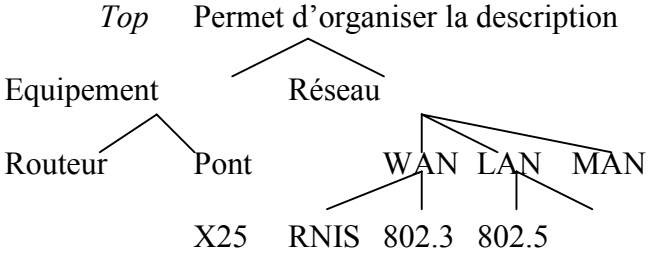
SMI (Structure of Management Information) – Norme ISO 10165-1 à 4

Les objets d'un réseau se classent en 3 catégories :

- Ressources physiques (cartes, liaisons, stations, …)
- Eléments logiques et virtuels (logiciel, canal, circuit, alarme …)
- Objets relatifs à l'environnement et à l'organisation

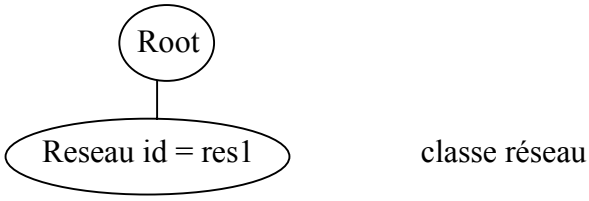
Les objets de gestion sont caractérisées par des propriétés et liés entre eux par des relations :

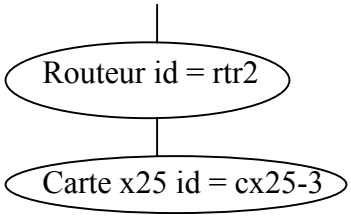
- Propriétés : attributs, opérations de gestion (applicables soit à un objet, soit à l'attribut), comportement de l'objet lui même, notification qu'il génère sur occurrence d'événements externes ou internes.
- Une classe d'objet est un ensemble d'objets qui partagent les mêmes propriétés. Instance d'une classe = objet. Chaque classe contient au moins un package obligatoire.
- Une relation d'héritage. Une classe d'objet peut bénéficier des propriétés d'une ou de plusieurs autres classes. *Ex d'arbre d'héritage* :



- Une relation de contenance exprime le fait qu'un objet d'une classe peut contenir un ou plusieurs autres objets pouvant appartenir à la même classe ou à des classes différentes. Ex : un objet « routeur » contient un « châssis » et une carte « x25 ».
- Une relation de nommage donne le moyen de nommer les objets.

Nom relatif (RDN : *Relative Distinguished Name*) : nom affecté à l'objet lors de sa création.
Nom global (GDN : *Global Distinguished Name*) : repérer de manière unique un objet.





[reseau id = res1, routeur id = rtr2, carte x25 id = cx25-3]

METHODOLOGIE POUR LA DEFINITION DE CLASSES D'OBJETS

Document GDMO (Guide lines for the Definition of Managed Objects)
Norme ISO 10165-4 / UIT X722

La syntaxe utilisée est ASN-1 (couche application)

1- <class_label> MANAGED OBJECT CLASS
   -- déclaration du nom de la classe --
2- DERIVED FROM <class_label> [,<class_label>, …] ;
   -- déclaration des super classes dont elle hérite --
3- CHARACTERISED BY <package_label> [,<package_label>,…] ;
   -- déclaration du package obligatoire de propriétés --
4- CONDITIONAL PACKAGE
   -- déclaration du package conditionnels –
   <package_label> PRESENT IF <condition_definition> [,<package_label> PRESENT IF <condition_definition>, …] ;
5- REGISTERED AS <object_identifier>
   -- déclaration du nom d'enregistrement de la classe --

III. GESTION DE RESEAU TCP/IP

Protocole SNMP (Simple Network Management Protocol)

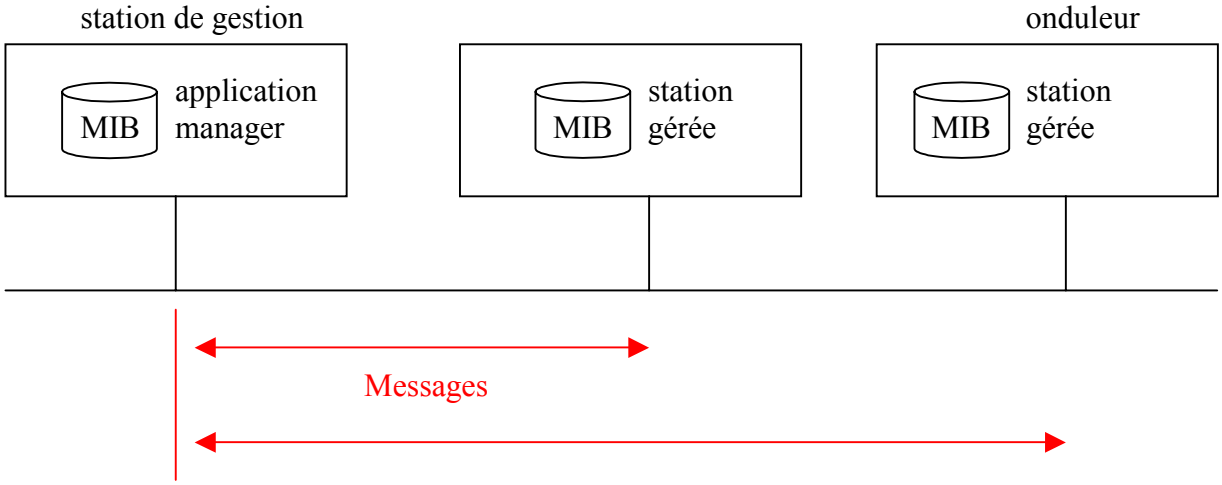
3.1) Introduction à SNMP

Le protocole SNMP est le langage que les agents et les stations de gestion (managers) utilisent pour communiquer. Très simple à assimiler et à mettre en œuvre (voir simpliste), c'est un protocole de type question/réponse asynchrone. Ce protocole est situé au niveau application du modèle OSI, c'est lui qui définit la structure formelle des communications. SNMP est encapsulé dans des trames UDP. SNMP est fourni avec tous les équipements à l'heure actuelle. Crée en 1989, bien après TCP/IP (1982 sous Unix).

SNMP s'inscrit dans la continuité de ICMP (Internet Control Management Protocol). L'IAB (Internet Activity Board) coordonne les travaux de normalisation de SNMP. La faiblesse de SNMP est la gestion de la sécurité (absence) pour la version 1. La sécurité va être gérée dans la version 2.

Documents de normalisation : RFC 155, RFC 1157, RFC 1213 (ftp.inria.fr, ftp.lip6.fr)

3.2) Architecture de la gestion SNMP



Messages en broadcast: GET: récupération d'une info
SET : programmer à distance
GETNEXT: récupération d'une suite de valeur
TRAP : avertisseur.

La MIB (Management Information Base) regroupe l'ensemble des variables relatives aux matériels et aux logiciels supportés par le réseau, et définit les objets de gestion dans l'environnement TCP/IP. La SMI (Structure of Management Information), définit comment sont représentées, dans la MIB, les informations relatives aux objets de gestion et comment sont obtenues ces informations.

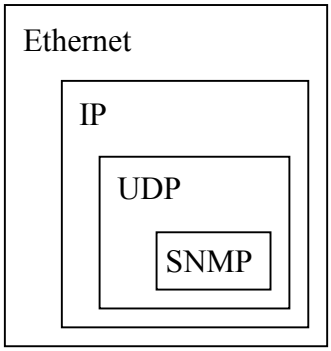
Les stations interrogent donc les agents pour observer leur fonctionnement et leur envoient des commandes pour leur faire exécuter certaines taches. Les agents renvoient les informations requises aux stations de gestion. Certains événements du réseau, tels que des erreurs de transmission, peuvent déclencher des alarmes envoyées aux stations de gestion. Cependant, l'envoi de messages de façon spontanée de l'agent vers le manager est limité. Les managers effectuent une interrogation périodique des agents de manière à vérifier leur état. La structure des paquets est définie en utilisant la syntaxe ASN1 (Abstract Syntax Notation).

SNMP a l'avantage d'être simple, cependant il a des capacités très limités au niveau sécurité, principalement pour l'authentification. Tous les systèmes SNMP doivent également supporter les protocoles DUPER et IP pour transporter les données entre les agents et les stations de gestion.

Agent de gestion (station d'administration) :

- Collecte des informations
- Programmer les équipements
- Traitements des TRAP (évenements répercutés par les agents). Un Trap est un traitement d'exception qui peut se produire n'importe quand.

Protocole simpliste : spécifie le mode et la nature de l'échange entre agent et superviseur ; datagramme sur UDP (sur IP)



3 services : GET, SET et TRAP

manager accède en lecture aux variables des objet de gestion

manager modifie les variables des objets de gestion

Le format de la trame SNMP est décrit ci-dessous :

Version	Communauté	PDU

Champs

Version : numéro de version SNMP. Le manager et l'agent doivent utiliser le même numéro.
Communauté : ce champ sert à identifier auprès du manager l'agent avant de lui accorder un accès.
PDU : il y a 5 types de PDU : GetRequest, GetNextRequest, GetResponse, SetRequest, et TRAP.
Une description de ces PDU est données ensuite.
Un premier format est utilisé pour les PDU du genre GET, ou SET :

Type de PDU	ID de requête	Statut d'erreur	Index d'erreur	Ojj 1, val 1

Champs

Type de PDU :
0 : GetRequest
1 : GetNextRequest
2 : GetResponse
3 : SetRequest

ID de requête : champ qui coordonne la requête du manager et la réponse de l'agent.
Statut d'erreur : entier qui indique une opération normale (cas 0) ou bien une erreur (cas ¹0)
Index d'erreur : identifie les entées avec la liste des variables qui ont causé l'erreur.
Obj/Val : association du nom de la variable à transmettre avec sa valeur.
Un second format est utilisé pour la TRAP PDU :

Type de PDU	Entreprise	Adresse Agent	Type générique	Type spécifique	Timestamp Obj 1, Val1

Champs

Type de PDU : dans ce cas toujours égal à 4.
Entreprise : identifie l'entreprise de management qui a défini la Trap .
Adresse Agent : adresse IP de l'agent.
Type Générique : décrit quel type de problème est survenu. (7 valeurs sont possibles).
Type Spécifique : est utilisé afin d'identifier une TRAP spécifique à une entreprise.
Timestamp : contient la valeur de l'objet sysUptime représentant le temps écoulé depuis la dernière initialisation..
Obj/Val : association du nom de la variable à transmettre avec sa valeur.

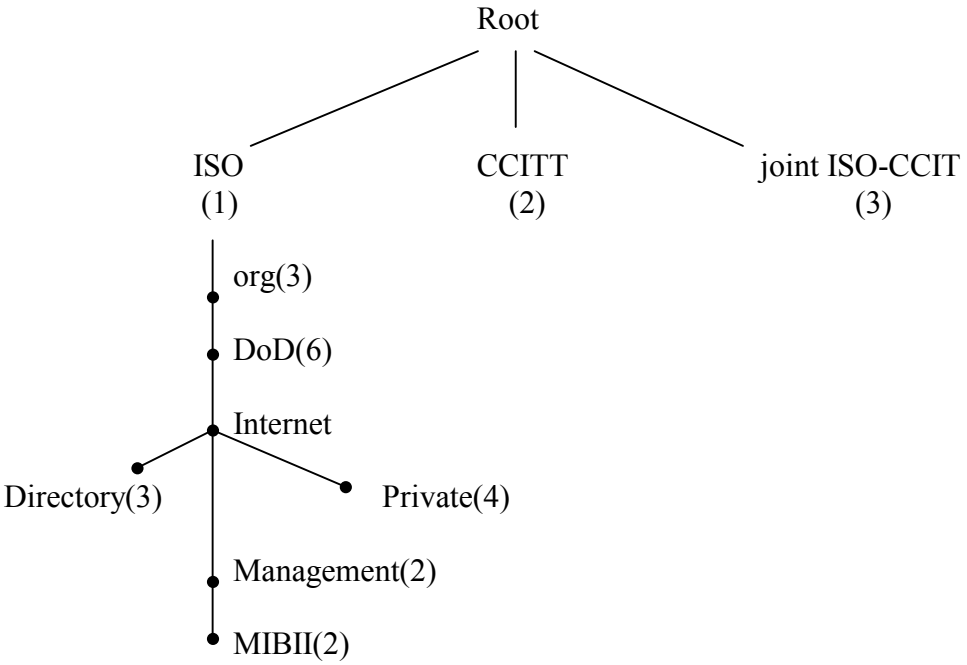
3.3) Les informations de gestion

Modélise sous forme de variables simples de type ASN-1. Classiques ou spécifiques (ex :format d'adresse(ex : IPV4, IPV6)). En voici quelques types :

- counter : entier naturel.
- jauge : entier non signé qui peut varier (charge d'un système, tension aux bornes d'une batterie).
- time ticks : temps écoulé.

Méthodologie pour définir des objets SNMP

La définition se fait par un langage de macros. RFC 1155 Chaque objet de la MIB SNMP est identifié de manière unique par un OBJECT IDENTIFIER.



1.3.6.1.2.2 → nom absolu d'un objet dans la MIB
1.3.6.1.2.3

MIB I :

Organisés en groupes (RFC 1156)

Informations générales sur le système géré (Sys Contact) (Sys Location)
Traduction d'adresse IP en adresse physique MAC (tables ARP).
Objets du protocole IP lui même.
Interfaces physiques de l'entité gérée (If Descr) (if Oper *Status = état opérationnel de l'équipement*).

MIB II :

Sur ensemble de la MIB I (RFC 1213). Avec différents groupes :

- Groupes system : info générale sur le système global
- Interfaces : info sur chaque interface station-serveur
- Adresse translation : description de la correspondance entre les adresses IP et le réseau
- IP
- ICMP
- TCP
- UDP
- EGP
- SNMP
- Transmission

Autres MIB :

Tout éditeur de logiciel réseau peut définir ses propres objets. Le sous arbre « private » est prévu à cet effet. RFC 1304, RFC 1289, RFC 1286, RFC 1284, RFC 1285… décrivent comment faire cela (le mode d'emploi).

3.4) Le protocole SNMD

Avec trois types de services :

GET : get_request : lecture d'une variable (demande)
get_next_request : lecture variable suivante
get_response : réponse à une opération GET
SET : set_request : écriture de la valeur d'un objet
set_response : réponse à un SET
TRAP : message spontanné de l'agent vers le messager + suite d'objets gérés.

Format des messages

En ASN.1, SNMP_message : - version (RFC 1157),
- data,
- community (ensemble de machines + manager).

Argument des messages GET, SET et TRAP (paramètres du champ data) :

Request_id : identificateur de la requête
Var_bind : suite de couple nom de variable, valeur
Error_status : résultat de la requête : 0 = pas d'erreur
Error_index : info supplémentaire (ex : cause de l'erreur)
Enterprise : type de l'objet ayant généré le TRAP
Agent_addr : adresse de l'agent
Generic_trap : type de trap générique (code)
Specific_trap : code d'un trap spécifique
Time_stamp : temps écoulé depuis la dernière initialisation

3.5) Autres protocoles d'administration :

CMOT (CMIP Over TCP/IP) : Implémentation au dessus du protocole UDP d'une couche présentation LPP (Light Weight Protocol Presentation)

OMG (Object Managment Group) : Groupe indépendant de 200 industriels du logiciels pour promouvoir les logiciels Objets distribués.

Architecture CORBA (Common Object Request Architecture) : Intéêt général pour des applications distribués (*ex : jeux en réseaux*).

X/OPEN et les API XMP/XOM : XMP est une interface en langage C qui décrit les en-têtes et spécifie le mode et la syntaxe d'appel.

OSF (Open Software Fundation)et plate-forme DME…

LA SECURITE DANS LES RESEAUX

LA SECURITE DANS LES RESEAUX

EXERCICES SUR LA SECURITE

SECURISATION D'UN RESEAU LOCAL

PROTOCOLES TRANSPORT SECURISES

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

LA SECURITE DANS LES RESEAUX

De tout temps, on a mis en place des dispositifs pour protéger des ressources. L'information est quelque chose de nouveau qui est impalpable et qui a pris de la valeur. On peut parler du piratage de logiciel qui va nuire à l'éditeur. Mais les distributeurs sont au courant de ces agissements et sans servent pour inonder le marché et pousser les utilisateurs "pirates" à faire acheter ces logiciels à leur entreprise. Tout ceci fait apparaître des risques et des menaces concernant la sensibilité, la vulnérabilité et l'intégrité des données.

Principaux concepts :
- risques et menaces
- sensibilité et vulnérabilité
- intégrité des données, authenticité des correspondants

IV. RISQUES et MENACES

1.1) Les Risques

Le risque dépend de paramètres que l'on peut maîtriser:

La vulnérabilité: degré d'exposition au danger (si l'on peut facilement rentrer dedans)

La sensibilité: caractère stratégique (valeur de l'information) = confidentialité.

Deux types de risques:

Le risque structurel: dépend de l'organisation de l'entreprise.

Le risque accidentel: indépendant de tous les facteurs de l'entreprise.

Quatre niveaux de risque:

- Acceptables: pas de conséquences graves pour les utilisateurs du réseau
  ex: panne électrique, perte de liaison, engorgement...
- Courants: pas de préjudices graves au réseau, on peut réparer facilement
  ex : gestion du réseau, mauvaise configuration, erreur utilisateur...
- Majeurs: dus à des facteurs graves et qui causent de gros dégâts mais récupérables
  ex : foudre qui tombe sur un routeur…
- Inacceptables: fatals pour l'entreprise, ils peuvent entraîner son dépôt de bilan
  ex : perte ou corruption des informations importantes…

1.2) Les Menaces

Ce sont les résultantes d'actions et d'opérations du fait d'autrui. Deux catégories:

- Passives: atteinte à la confidentialité (prélèvement par copie, écoute de l'information sur les voies de communication) souvent indétectables.
- Actives: nuisent à l'intégrité des données (brouillage, déguisement (se faire passer pour quelqu'un d'autre), interposition (vol de session)).

Niveaux de compétence: le débutant devine les mots de passe, le professionnel les décrypte.

Statistiquement, ces menaces se répartissent de cette façon:
26% accidents (incendies, inondations, pannes, catastrophes naturelles…)
17% erreurs (défauts de qualité, erreurs humaines…)
57% malveillance (d'origine interne à 80%: vols des équipements, copies illicites de logiciels, sabotages + attaques logiques, intrusions et écoutes, acte de vengeance...)

V. Normalisation ISO

Emetteur: envoie le message
Récepteur: reçoit le message
Réseau de transport: infrastructure

2.1) Garanties aux différents niveaux

Emetteur:

- Le message émis ne peut être compris ou utilisé que par le destinataire désigné.
- Le destinataire ne peut pas nier avoir reçu le message s'il a vraiment reçu (non répudiation).
- Le message parvient bien au destinataire désigné.
- L'identité du destinataire n'est pas déguisé.
- Le destinataire ne peut pas prétendre avoir reçu un message non envoyé.

Récepteur:

- Le message provient de l'émetteur déclaré et authentifié.
- Le message n'a pas été modifié pendant le transfert.
- Le message n'a pas été intercepté ou dévoilé pendant le transfert.
- L'émetteur ne peut pas nier avoir effectivement envoyé un message s'il a été reçu.
- Le récepteur ne veut pas recevoir de messages d'émetteurs non autorisés.

Réseau:

- L'émetteur est authentifié et ne peut pas nier avoir émis le message.
- Le récepteur est authentifié et ne peut pas nier avoir reçu le message.
- Les copies des messages transmis sont détruites après leur livraison au destinataire.

2.2) Besoins des entités émetteur + récepteur

- Disponibilité (ou continuité de service),
- Confidentialité des informations,
- Intégrité des données,
- L'authentification (assurance qu'il n'y a pas tromperie sur l'identité),
- La non répudiation (ne pas avoir la possibilité de dire c'est pas moi!).

VI. SERVICES DE SECURITE

3.1) Authentification

Le but: garantir l'identité des correspondants. Authentification de l'entité homologue (lutte contre le déguisement). Authentification de l'origine.

3.2) Contrôle d'accès

Il empêche l'utilisation (lecture, écriture, création, suppression) non autorisée des ressources (utilise l'authentification).

3.3) Confidentialité des données

Elle empêche les données d'être consultées par identités non autorisées. La plupart du temps, ce sont des fraudes passives (consultation sans modification ou altération).
Quatre secteurs possibles à protéger:

- Mode orienté connexion,
- Mode datagramme (non connecté),
- Champ spécifique (ex: code d'accès)
- Secret du flux.

3.4) Intégrité des données

Ce sont des fraudes actives. Le service détecte les altérations de données.
Cinq classe de service d'intégrité:

- Intégrité en mode connexion avec récupération (contrôle sur la totalité de la transmission),
- Intégrité en mode connexion sans récupération (simple réponse pour dire si OK ou pas),
- Intégrité de champs spécifiques,
- Intégrité des données en mode sans connexion (détection de modifications mais l'insertion ou les répétitions ne sont détectées que partiellement),
- Intégrité d'un champ spécifique en mode sans connexion.

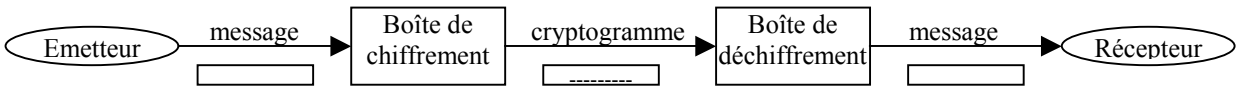
3.5) Non répudiation

Elle fournit au récepteur/émetteur une preuve qui empêche l'émetteur/récepteur de contester l'envoi du message.

VII. Les mécanismes de sécurité

4.1) Le Chiffrement (c'est légal, mais le cryptage est interdit)

Opération qui transforme le codage d'une formation en un autre codage (sans perte d'informations). Opération très proche du compactage (dans ce cas, la clé est contenue dans l'algorithme). Les données sont brouillées.

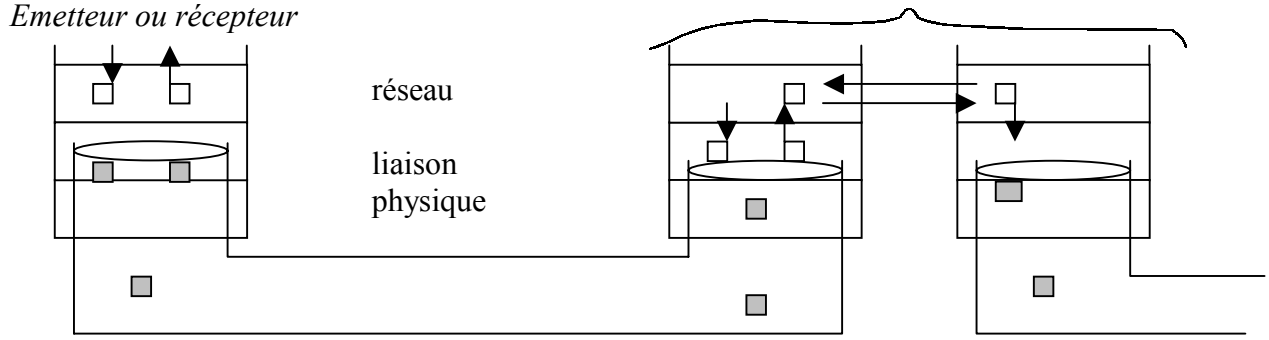


Ils faut que les associations de code soit de bout en bout.
Et il faut prévoir la communication des codes.

Le chiffrement voie par voie dans le réseau

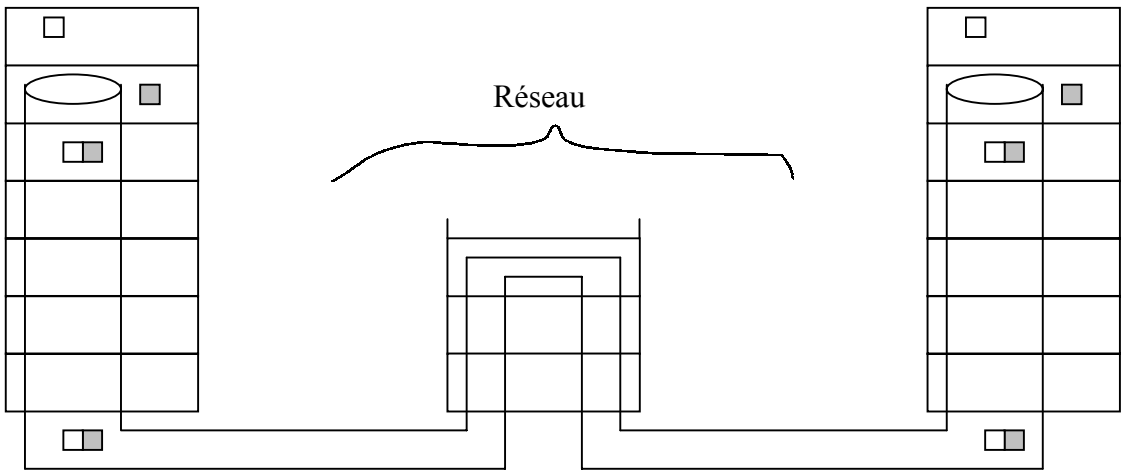
Niveaux OSI:
- physique: boîtes noires
- Liaison: toutes les trames sont chiffrées,
- Réseau: par circuit virtuel.

Routeur : il est obligé de décoder puis de recoder l'entête pour voir le destinataire



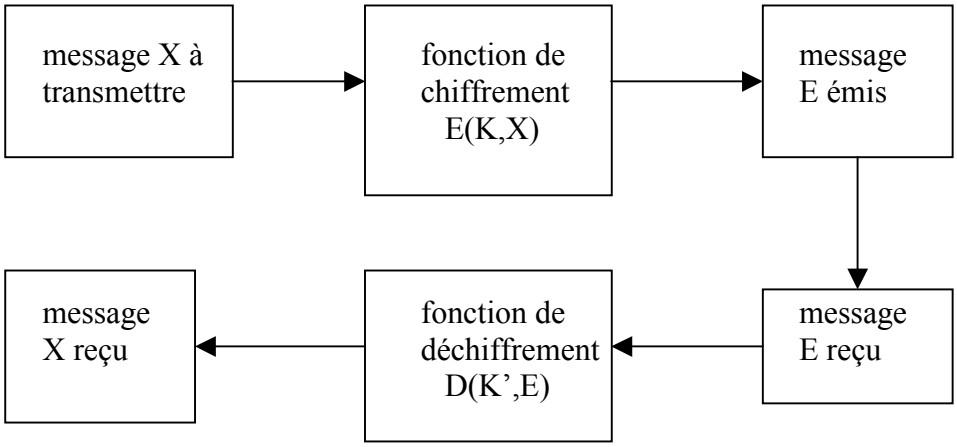
Chiffrement de bout en bout

Routage en clair mais les données sont chiffrées.



Principe du chiffrement

❖ *Message en clair*



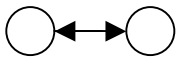
X : message d'origine à transmettre
K : clé de chiffrement E(K,X) : message chiffré
K' : clé de déchiffrement D(K',E) : message déchiffré

Exemple : *D=E= Ou exclusif*
K=K'= une chaine de caractères
A= (a + x) + x

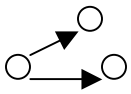
La solidité ici dépend de la longueur de la clef. L'idéal c'est qu'il n'y ai pas de répétitions de motifs, en général on compacte avant de chiffrer.

❖ *Système à clé secrète (ou chiffrement symétrique)*

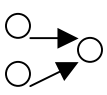
Tout le secret réside dans la clé ce qui implique un problème de diffusion de la clé.



❖ *Système à clé publique (ou chiffrement asymétrique)*



Une clé privée, l'autre publique.
Ex: *Signature électronique* (je chiffre avec ma clé et je diffuse la clé publique pour qu'on puisse vérifier l'authentification).
Mail liste où tout le monde a le droit d'écrire mais seul le destinataire peut lire.



❖ *Gestion des clés*

- Création
- Affectation
- Distribution : Manuelle (guichet, pistolet chargeur…)
  Automatique : skey (affectation de clé à usage unique)
  CDC (centre de distribution des clés)
- Révocation.

Exemples d'algorithme de chiffrement

DES (Data Encryption Standard)

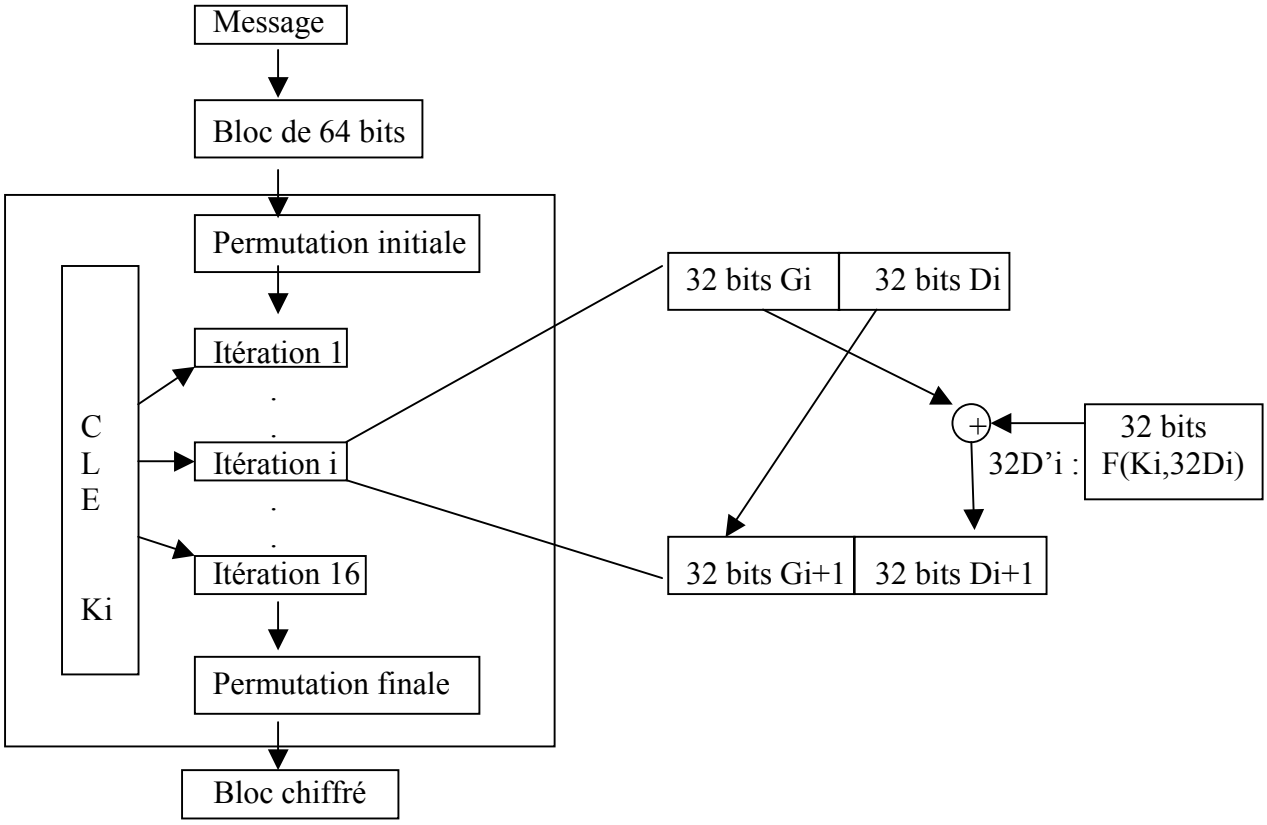
Mis au point dans les laboratoires d'IBM et adopté par le NBS en 1977.
ISO: DEA (Data Encryption Algorithm). Norme ISO 8731.
Cet algorithme est répandu dans le monde industriel et bancaire (cartes bleues).

- Principe de fonctionnement

On découpe un message M (suite binaire) en un ensemble de n blocs U de 64 bits. Chaque bloc est chiffré et déchiffré indépendamment des autres avec une clé de 56 bits.

- Chiffrement

On génère, à partir de K, 16 clés intermédiaires Ki avec i ∈ [1,16]



Etapes : 1) permutation initiale
transposition de U en U' (16 itérations)
2) permutation finale (transformation)

TELEPASS

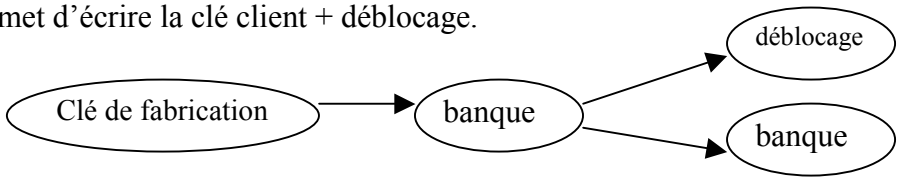
Conçu par Bull pour les cartes à microprocesseur (cartes bancaires).
Telepass 1 : principe :

R = f (E, C, I)

R = résultat
E = nombre ou message provenant de l'extérieur (64 bits)
C = clé secrète de 92 bits
I = information interne ou externe de 32 bits

Carte Bancaire B1

Cela fonctionne avec une clé de fabrication qui autorise l'écriture de la clé banque qui à son tour permet d'écrire la clé client + déblocage.



Autres clés : ouverture de zones prestataires, transaction, prestataire.
(3 faux codes = bloqué, propriétaire de la carte, certification d'une transaction)

EXEMPLE D'UTILISATION : Un terminal A veut authentifier une carte B.

A envoie à B un nombre aléatoire E
B calcule R=f(E,C,I) *(avec I la date d'expiration de la carte par exemple)*
B envoie le résultat à A $f^{-1}(R,I,E)$.
La banque dispose d'une carte « mère » référence et calcule les clés de chaque carte « porteur ».

EXEMPLE D'UTILISATION :

Un émetteur d'ordre forme un message d'entrée T et calcule $f^{-1}(T,I,C) = E$
E est envoyé à la carte R = f (E,C,I)
Si R = T, la carte est prête à télécharger l'ordre.

RSA (Rivest, Shamir et Adleman)

Algorithme asymétrique proposé par Rivest, Shamir et Adleman en 1977 et mis au point sous forme informatique en février 1988.

➢ Principe de fonctionnement

- L'utilisateur choisit 2 grands nombres premiers (p et q) et calcule leur produit (n = pq).
- Il choisi un e < n et publie (e, n) qui constitue sa clé publique. Il conserve p et q secrets.
- Il obtient sa clé secrète en cherchant un nombre d tel que

quelque soit $x \in \square \mid 0 \leq x \leq n$ $\quad x^{ed} = x^{de} = x \bmod n$

Selon la théorie des nombres, d existe toujours (avec une condition sur e) mais son calcul est impossible si on ne connaît pas p et q.

Chiffrement :	$E(x) = x^e$ (modulo n)	fonction publique
Déchiffrement :	$D(x) = x^d$ (modulo n)	fonction secrète

Donc l'envoi d'un message m se fait en calculant m^e *(modulo n)*. Et la lecture du message x se fait en calculant $(m^e \text{ (modulo n)})^d$ *(modulo n)*.

La signature du message se fait en envoyant m^d *(modulo n)*. Et le test de la signature(publique) se fait en envoyant $(m^d \text{ (modulo n)})^e$ *(modulo n)*.

Le temps de réponse de cet algorithme étant important (temps de chiffrement de la fonction D), on ne peut pas l'utiliser pour des applications temps réel mais plutôt pour de l'authentification / signature.

EXEMPLE D'UTILISATION :

Un utilisateur A a rendu sa clé (e, n) publique = (11, 11023). Il garde sa clé secrète d = 5891 et n = 73 * 151 . L'utilisateur B veut envoyer un message à A (par exemple son code de carte bleue x=2814) et souhaite protéger son message.

Le chiffrement se fait avec $y = E(x) = 2814^{11}$ modulo 11023 = 1473

A est le seul à pouvoir lire le message en appliquant sa clé secrète

$x = D(y) = 1473^{5891}$ mod 11023 = 2814. (très long !)

4.2) Signature électronique

Jadis, c'était le cachet de cire (sceau) sur les courriers. Désormais on parle de chiffrement asymétrique des données avec une clé privée (ex : message + time stamp (heure chiffrement)). L'algorithme consiste en un booléen pour l'authentifier la signature (avec une clé publique pour le déchiffrement). Le système est souvent sécurisé par notarisation et la mise en place de checksum pour vérifier qu'un message n'est pas modifié par exemple (SVM de BSD, MD5 de SYS5).

Le message signé est sous la forme : **E(K, h(M))** = signature
Avec M = le message et h(M) = image du message par une fonction de condensation

Ce système est vulnérable car il existe probablement une autre forme mathématique du message qui donne la même signature…

4.3) Contrôle d'accès

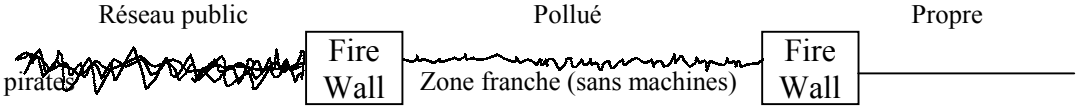
Les informations utilisées sont :
- listes de droits d'accès (Access Control List)
- mots de passe
- jetons de droits d'accès (tickets)
- certificats
- libellés de sensibilité de données

4.4) Intégrité des données

Ceci consiste à vérifier que la donnée n'a pas été modifiée. Pour cela, sur un réseau non sécurisé, on effectue un chiffrement des données avec en général le rajout d'un horodatage pour ne pas pouvoir rejouer les codes une fois chiffrés. Sur un réseau sécurisé, les données sont en clair mais on effectue un contrôle d'accès aux ressources.

4.5) Echange d'authentification

Sur un réseau sécurisé, cela peut se faire par mot de passe (login + password). Mais sur un réseau non sécurisé, on utilise un notaire avec des certificats. Le notaire décide si l'on peut accéder à la ressource (et il faut fournir un certificat pour y accéder). On peut par exemple utiliser une carte à puce pour stocker l'identité.

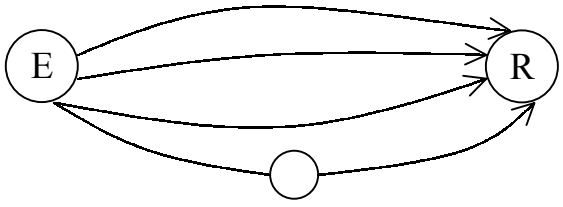


4.6) Bourrage de flux

Cela consiste à noyer une information sensible dans un flux d'informations peu sensibles (camouflage). Aussi bien sur des liaisons (usine qui transmet en permanence) ou des fichiers (image d'un million de pixel contenant l'information). Il faut quand même que l'alphabet utilisé soit le plus proche possible de message sensible.

4.7) Contrôle de routage

C'est la possibilité de choisir et de modifier la route pour éviter l'observation du support. Généralement, l'information est répartie sur plusieurs lignes puis recollée au bout.



4.8) Récapitulatif

Mécanismes / Services	Chiffrement	Signature	Contrôle	Intégrité d'accès	Bourrage de flux	Contrôle de routage	notarisation
Authentification de l'entité homologue	S	S					S
Contrôle d'accès			S			S	
Confidentialité	Y					S	
Confidentialité d'un champ	Y				S	S	
Secret de flux	Y						
Intégrité sans connexion	S	Y		Y			
Intégrité avec connexion	S						
Authentification de l'origine	S	S		S			S
Non répudiation		S		S			S

Case blanche : le service n'est jamais rendu par le mécanisme.
Y : le mécanisme suffit à assurer le service (service rendu).
S : aide à assurer le service (service rendu partiellement).

VIII. Le système Kerberos

Problématique : Comment échanger des données « confidentielles » et l'utilisation des ressources sur un réseau non sécurisé.

5.1) Description du système Kerberos

Kerberos (cerbère) est un système d'authentification conçu par Miller et Neuman [KAAS] pour les environnements réseaux ouverts dans le cadre du projet **Athena** du MIT (partie sécurité du projet - années 1978-80). Il met en jeu 3 types d'acteurs : le serveur Kerberos, des serveurs qui fournissent des services comme telnet, ftp, mail..., et des clients qui veulent accéder à ces services. Un scénario type est un client qui pour accéder aux services d'un serveur *S* doit utiliser le serveur Kerberos pour s'identifier auprès du serveur *S*.

Toute cette architecture repose donc sur la confiance accordée par les clients et les serveurs à Kerberos en matière d'authentification. Kerberos repose sur l'utilisation de clé privée. Il utilise un chiffrement basé sur DES (Data Encryption Standard). Les clients sont des programmes ou bien des utilisateurs, les serveurs sont des machines qui fournissent un ensemble de services (qui sont eux-mêmes des clients pour Kerberos). Kerberos possède une base de donnée pour établir la correspondance Client-Clé Privée. Cette clé privée est connue seulement du client et de Kerberos. Dans le cas où le client est un utilisateur, la clé privée est en fait le mot de passe de l'utilisateur crypté (comme ce que l'on peut trouver dans /etc/passwd ou /etc/shadow). Les services des serveurs doivent eux aussi s'enregistrer auprès de Kerberos.

Kerberos fournit aussi des clés de session (*session key*) qui permettent à deux clients de dialoguer ensemble en chiffrant leurs messages.

Ainsi Kerberos peut fournir trois niveaux de sécurité entre lesquels le programmeur de l'application pourra choisir :

- Une authentification à l'établissement de la connexion
- Une authentification pour chaque message envoyé
- Une authentification et un chiffrement de chaque message

Le serveur Kerberos se compose en fait de deux serveurs. D'une part, un serveur d'administration (**KDBM server** = Kerberos Data Base Managment) qui fournit un accès en lecture/écriture sur la base de donnée de Kerberos. Ce serveur permet en autre la mise à jour des clients. D'autre part, un serveur d'authentification (**Kerberos server**) qui fournit un accès en lecture seul à la base de donnée Kerberos. Ce serveur est celui auquel s'adresse les clients. Il est parfaitement envisageable d'avoir plusieurs Kerberos server avec chacun une copie de la base de données Kerberos. Les copies sont mises à jour par le KDBM server lorsque cela est nécessaire.

Il existe un ensemble de programmes pour l'utilisateur final permettant de se connecter à Kerberos et notamment de changer un mot de passe Kerberos. Kerberos fournit aussi une API pour être utilisé à partir d'un programme (pour « kerberiser un service »). Enfin, le système Kerberos doit pouvoir nommer ses clients, pour cela la convention suivante est utilisée : name.instance@realm. ex : rlogin.priam@ATHENA.MIT.EDU

service.machine@……royaume

Il s'agit du service rlogin de l'hôte priam du **realm** ATHENA.MIT.EDU. Un **realm** est l'ensemble des machines protégées par Kerberos (royaume).

5.2) Fonctionnement de Kerberos 4

Dans un premier temps, nous allons nous intéresser à la version 4 du système **Kerberos**. Cette section aborde les principes de base du système **Kerberos**.

5.2.1) Abréviations

Pour alléger textes et schémas un certain nombre d'abréviations seront utilisées par la suite :

C	client
S	serveur
Addr	adresse réseau du client
Life	durée de vie du ticket
Tgs	serveur de ticket
Kerberos	serveur d'authentification
KDBM	serveur d'administration
K_x	clé privée de x
$K_{x,y}$	clé de session entre x et y
$\{abc\}K_x$	abc crypté avec la clé de x
$T_{x,y}$	ticket de x pour utiliser y
A_x	authentifieur de x
WS	station de travail

5.2.2. Pièces justificatives

Lorsqu'un utilisateur veut utiliser le service fournit par un serveur, il doit s'identifier auprès de celui-ci. Pour cela, il utilise deux pièces justificatives : les **tickets** et les **authentifieurs**. Toutes deux utilisent le chiffrement par clé privée, mais elles sont différentes.

- **Ticket**

Le ticket a la forme suivante : $\{s, c, addr, timestamp, life, K_{s,c}\}K_s$

Il permet de donner au serveur *s* l'identité du client *c*, l'adresse de ce client *addr* de façon sécurisée. Il permet aussi de donner au serveur la clé de session $K_{s,c}$. Les tickets peuvent être utilisés plusieurs fois. Ils ont une date limite de validité fournit par le *timestamp*.

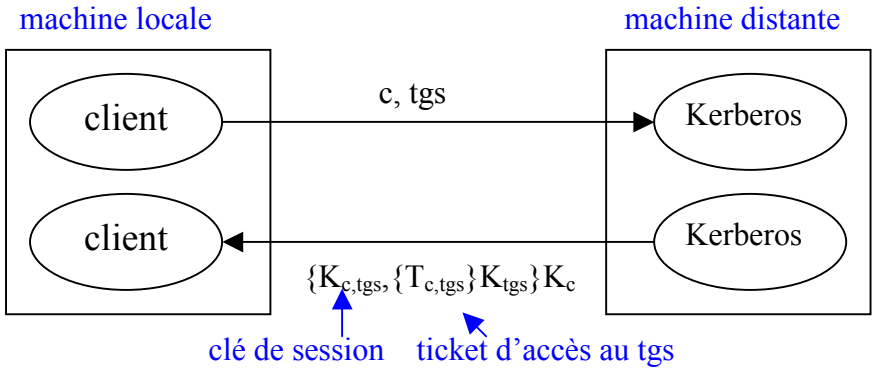
- **Authentifieur**

L'authentifieur a la forme suivante : $\{c, addr, timestamp\}K_{s,c}$

Contrairement au ticket, il ne peut être utilisé qu'une seule fois. Il doit être généré à chaque fois que l'utilisateur souhaite utiliser un service. C'est l'utilisateur lui-même qui la génère.

5.2.3. Obtention du ticket Initial

L'un des objectifs de Kerberos est d'éviter que l'utilisateur ne doive taper son mot de passe à chaque fois qu'une authentification est nécessaire, c'est à dire potentiellement à chaque requête envoyée au serveur. C'est pour cela que le serveur Kerberos va fournir un ticket au client (l'utilisateur doit entrer son mot de passe), qui permettra ensuite d'obtenir tous les autres tickets auprès du **TGS** (serveur de tickets) sans avoir à utiliser de nouveau le mot de passe. Après tout, obtenir des tickets auprès du TGS est un service comme un autre. Le même protocole sera donc utilisé pour obtenir des tickets auprès du TGS que pour accéder à n'importe quel autre service. Il faut donc obtenir un ticket d'accès au TGS auprès du serveur Kerberos.

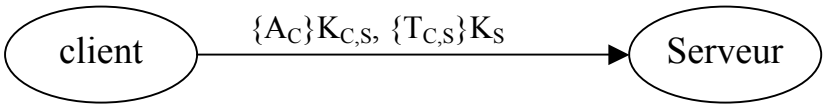


L'utilisateur utilise le programme de login sur le serveur Kerberos. On lui demande son nom. Une fois celui-ci entré, une requête est envoyée auprès du serveur Kerberos contenant le nom du client et le nom du service TGS. Si le client est connu, le serveur Kerberos renvoie une clé de session $K_{c,tgs}$, et le ticket d'accès au TGS $\{T_{c,tgs}\}K_{tgs}$, le tout crypté par la clé du client. Lorsque la réponse est reçue par le client, on demande à l'utilisateur son mot de passe, ce qui permet de décrypter la réponse et de récupérer ainsi la clé de session et le ticket.

Explication: Le système Kerberos assure de la part de l'utilisateur une authentification explicite unique qui utilise son mot de passe. Les autres authentifications reposent sur le fait que celle faite par le serveur Kerberos est bien correcte. De plus, le système Kerberos évite le transit sur le réseau du mot de passe de l'utilisateur ou de sa clé privée. En fait, l'authentification repose sur le fait que si le client est un pirate, il ne pourra pas décoder la réponse, et ne pourra donc pas récupérer ni le ticket, ni la clé de session.

5.2.4. Authentification auprès d'un service

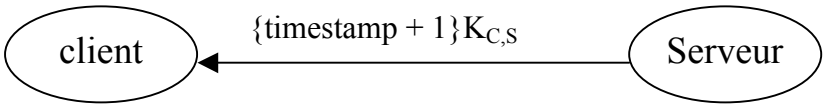
Dans un premier temps, on va supposer que le client possède un ticket pour accéder à un service. Ce ticket lui a été envoyé par le **TGS**. Il a aussi reçu une clé de session. Le client construit un authentifieur de la façon décrite précédemment. Cet authentifieur est codé avec la clé de session qui a été reçue avec le ticket.



L'ensemble ticket-authentifieur est envoyé au serveur. Celui-ci décode le ticket et utilise la clé de session pour décoder l'authentifieur. Il peut alors comparer les informations contenues dans le ticket et dans l'authentifieur, l'adresse IP à partir de laquelle la requête a été envoyée et l'heure. Si tout correspond, le client est considéré comme étant authentifié.

L'utilisation des timestamps suppose que les horloges du serveur et du client soient synchronisées. De plus, pour éviter des attaques du type *replay* (un pirate ayant capturé le ticket et l'authentifieur renvoie ceux-ci pour accéder au service) il convient que le serveur garde une trace des authentifieurs déjà utilisés.

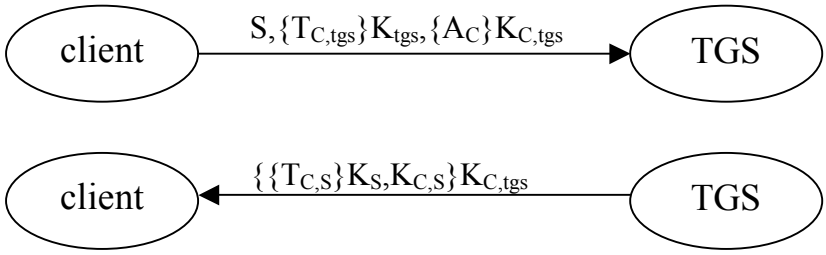
Il est possible au serveur de prouver son identité au client, il lui suffit de renvoyer le timestamp du client incrémenté le tout codé avec la clé de session. Un serveur pirate n'aurait pas pu décoder le ticket et donc se procurer le timestamp de l'authentifieur.



On parle alors d'authentification mutuelle.

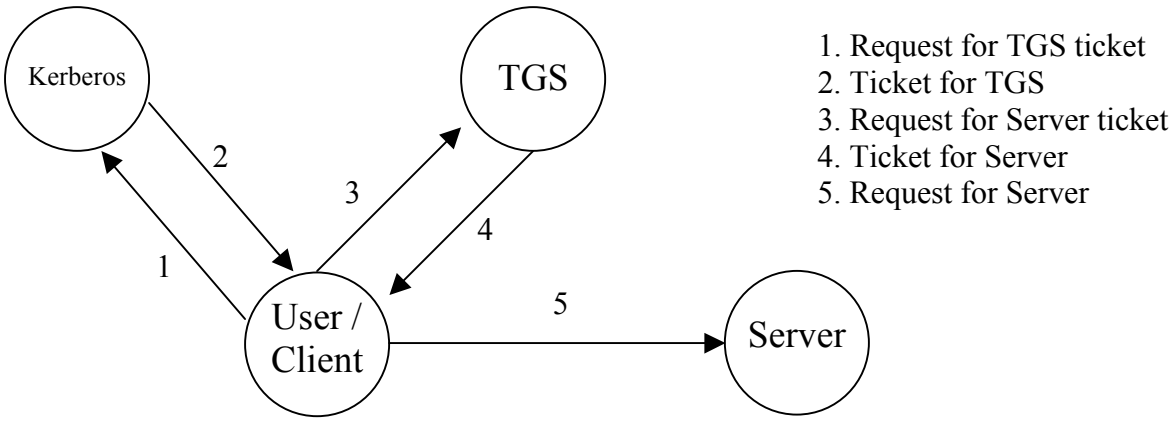
5.2.5. Obtention d'un ticket pour un service

Reprenons l'ordre chronologique des choses. Avant de pouvoir accéder au service, il faut que le client possède un ticket. Pour cela il doit le demander au **TGS**. En fait, on procède de la même façon que pour un service normal :



On envoie une requête contenant le serveur auquel il veut s'adresser, le ticket permettant d'accéder au **TGS** $\{T_{c,tgs}\}K_{tgs}$ ainsi qu'un authentifieur crypté par la clé de session. Le TGS se base sur les même critères pour s'assurer de l'identité du client, puis il lui envoie le ticket pour le serveur demandé ainsi qu'une clé de session encodée par la clé de session entre le TGS et le client. Tout est basé sur la sécurité du serveur de ticket (Kerberos).

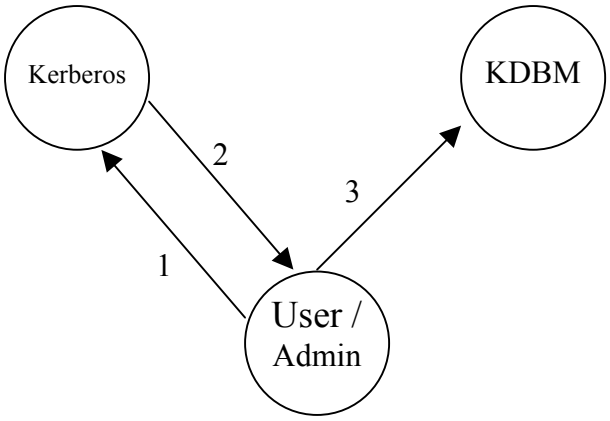
5.2.6. Résumé du protocole d'authentification Kerberos



5.2.7. Aspects d'administration

Nous allons maintenant aborder l'aspect de l'administration d'un serveur Kerberos. C'est le contrôle des ressources que l'on met à la disposition des gens. Nous avons déjà vu que celui-ci utilise une base de donnée contenant l'ensemble des clients ainsi que leur clé privée. Il faut donc être capable de gérer cette base de donnée de façon sécurisée, et donc notamment d'authentifier les accès à cette base de données.

Pour cela on utilise le programme d'administration *kadmin*. En fait, de façon logique, ce programme va s'adresser au serveur Kerberos pour lui demander un ticket d'accès au serveur KDBM.



1. Request for KDBM ticket
2. Ticket for KDBM
3. *kadmin* ou *kpasswd* request

Comme nous l'avons vu la base de donnée peut être répliquée. Il faut donc assurer la cohérence entre la copie maître et les copies esclaves. La cohérence des données est assurée automatiquement par des démons *kpropd*, les mises à jour sont transférées sur le réseau de façon chiffrée avec la clé de la base de donnée. Pour éviter qu'un site pirate n'envoie des mises à jour de la base de donnée (par exemple des anciennes mises à jour) un checksum (crypté avec la clé privée de la base de donnée) est d'abord envoyé. Ainsi, le checksum des données reçues est comparé à celui précédemment envoyé.

5.3) Fonctionnement de Kerberos 5

La version 4 de Kerberos comporte un certain nombres de défauts. Certains sont liés au manque d'universalité de Kerberos car celui-ci a d'abord été développé dans le cadre du projet ATHENA, d'autres sont liés à des aspects techniques. Nous présentons ici quelques-uns des défauts de Kerberos 4, ainsi que quelques-unes des améliorations apportées par Kerberos 5.

5.3.1. Défauts relatifs à l'universalité de Kerberos

- Utilisation de DES (*Data Encryption Standard*) pour crypter les messages. Or l'exportation de DES est limitée hors des Etats-Unis à l'acceptation du gouvernement américain.
- Utilisation d'adresses IP seulement.
- L'ordre des bits des messages est dépendant de la plate-forme (pas de couche présentation).
- Pas de possibilité de propager l'authentification d'hôte en hôte. Si un client authentifié fait un rlogin sur une machine, de celle-ci il n'est pas authentifié pour accéder à une autre.
- Restriction du système de nommage : name.instance@realm (3x39 caractères).
- L'authentification inter-realm (royaume) est possible mais coûteuse: $O(n^2)$ échanges de clé.

5.3.2. Défauts techniques

- Double chriffrement : il se produit lorsqu'un serveur de ticket envoie sa réponse : $\{K_{c,tgs}, \{T_{c,tgs}\}K_{tgs}\}K_c$ en fait il suffit d'envoyer la réponse : $\{K_{c,tgs}\}K_c$, $\{T_{c,tgs}\}K_{tgs}$ En effet, il ne sert à rien de coder le ticket car celui-ci passe ensuite en clair sur le réseau lors de la requête du client auprès du serveur.
- Le stockage des authentifieurs déjà utilisés n'a pas été implémenté (pas de stockage).
- Le checksum utilisé n'est pas un checksum standard de chiffrement, ses propriétés ne sont pas connues et il n'y a donc pas eu d'étude de robustesse.

5.3.3. Changements opérés dans Kerberos 5

- Possibilité de pouvoir utiliser des algorithmes de chiffrement différents et donc exportable.
- Possibilité d'utiliser des adresses réseaux différentes de IP.
- Utilisation de ASN.1 (*Abstract Syntax Notation One*) pour permettre une indépendance vis à vis de l'ordre des bits des messages quelque soit la plate-forme utilisée.
- Compatibilité avec la version 4, une application peut supporter à la fois Kerberos 4 et 5.
- Modifications dans la structure des tickets (rendre l'utilisation des timestamps plus souple).
- Meilleur support des authentifications inter-realm (royaume) par une hiérarchisation de ceux-ci. L'échange de clé est en O(n.log n).

5.4) Failles dans le système Kerberos 5

Même si, Kerberos 5 apporte de nombreuses améliorations par rapport à la version précédente, il subsiste tout de même un certain nombre de failles.

5.4.1. Replay Attacks (attaques par répetition)

Kerberos utilise les authentifieurs pour se protéger contre les attaques basées sur la réutilisation des tickets (il suffit de sniffer le réseau pour récupérer un ticket). Les authentifieurs reposent sur l'utilisation de *timestamps*, ceci est problématique pour plusieurs raisons. L'utilisation des *timestamps* suppose que l'attaque ne se produira pas pendant la durée de vie de l'authentifieur (typiquement 5 minutes), or ce délai de 5 minutes est relativement long. De plus, le serveur devrait stocker ces authentifieurs, or dans les implémentations de Kerberos cela n'est pas fait. En effet, cela est techniquement difficile à faire notamment dans le cas de connexion TCP. Le serveur TCP crée un processus fils pour chaque nouvelle connexion, et ces processus fils n'ont pas de mémoire partagée avec leur père, il leur est donc difficile de tenir à jour l'ensemble des authentifieurs déjà utilisés. Il existe des solutions pour faire communiquer les processus entre eux, mais toutes ont des failles et certaines soulèvent à leurs tours des problèmes d'authentification.

5.4.2. Service de temps sécurisé

Comme nous l'avons vu précédemment, l'authentification dans Kerberos repose sur la synchronisation des horloges des machines. Si un hôte peut être trompé sur l'heure, un authentifieur pourra être réutilisé à volonté. Tout le problème est que Kerberos repose sur le protocole de synchronisation que les différents hôtes utilisent, et donc Kerberos fait confiance à la capacité d'authentification de celui-ci. Comme Kerberos est destiné à être utilisé dans des environnements très différents cela pose un réel problème. Il suffit d'attaquer en rejouant un vieux ticket après avoir changé l'heure des machines.

5.4.3. Attaque sur le password de l'utilisateur

Lors du premier échange avec le TGS, celui-ci répond en envoyant un message encodé avec la clé secrète du client. Cette clé est calculée par un algorithme public à partir du mot de passe de l'utilisateur. Or les mots de passe utilisateur ne sont souvent pas très bons car trop simples. Il est facile à un pirate d'essayer de deviner le mot de passe, de calculer une clé puis de décrypter le message. Si le mot de passe est trop simple, ce type d'attaque peut fonctionner.

5.4.4. Spoofing login

La faille se situe cette fois-ci sur la machine du client. Si un intrus arrive à en prendre le contrôle ou simplement à remplacer le programme de login par un qui enregistre les mots de passe des utilisateurs, Kerberos est mis en défaut. Le lecteur pourra remarquer qu'il existe de multiples façons de récupérer un mot de passe d'un utilisateur : au travers du serveur X, lors de rlogin non sécurisé.

Une solution à ce problème pourrait être les One-time passwords, mais ceux-ci sont difficiles à mettre en oeuvre avec Kerberos. En effet, la base de données contenant les mots de passes devrait tenir compte du fait que ceux-ci changent à chaque fois. Il s'agit d'une faille très importante dont la solution réside principalement dans l'utilisation de périphérique de sécurité comme des cartes à puces qui permettent de protéger le mot de passe de façon plus efficace.

5.5) De l'utilisation de Kerberos

Au vu de ce qui précède, peut-on conclure que Kerberos est un système valide d'authentification ?

Bien qu'il existe de nombreuses failles, elles sont pour la plupart résolvables (carte à puce) ou bien atténuable (on remplace une grosse faille par une plus petite). Il est de toute façon difficile d'envisager un protocole totalement sûr dès lors que l'intégrité de toutes les machines a été violée ainsi que celle du réseau.

Le code de Kerberos est public, ce qui renforce sa robustesse. En effet, de nombreux groupes ont étudié les implémentations en détails, relevant ainsi des failles. Mais ils ont aussi proposé des corrections à apporter;. De plus, il est facile de se faire une idée précise du niveau de sécurité offert par un tel système lorsque l'on a accès à son implémentation

5.6) Références

[KAAS] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, *Section E.2.1 : Kerberos Authentication and Authorization System*, M.I.T. Project Athena, Cambridge, Massachusetts (December 21, 1987).

[LKAS] Steven M. Bellovin, Michael Merritt : *Limitations of the Kerberos Authentication System* AT&T Bell Laboratories

J. G. Steiner, C. Neuman, J.I. Schiller : *Kerberos: An Authentification Service for Open Network Systems* M.I.T Project Athena, Cambridge, Massachusetts

J. T. Kohl, B. C. Neuman, T. Y. Ts'o : *The Evolution of the Kerberos Authentification Service*

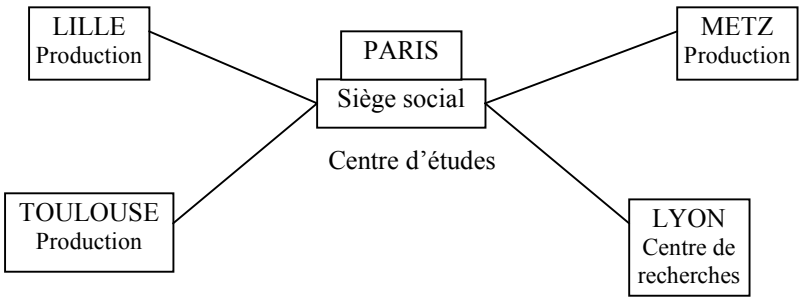
Site Web à voir:

- Kerberos Ref. Page: *http://www.contrib.andrew.cmu.edu/~shadow/kerberos.html*
- The Kerberos Network Authentification Service: *http://gost.isi.edu/info/kerberos/*
- Pour accéder à tous ces documents: *ftp://ftp.nada.kth.se/pub/krb/doc/*
- Ce cours : *http://sirac.imag.fr/~riveill/cours/ensimag3-GL-SI/1998-99/gr15/index.html*

EXERCICES SUR LA SECURITE

EXERCICE: MESSAGERIE ELECTRONIQUE

Application de messagerie électronique qui permet aux cadres d'une entreprise de communiquer entre eux.



1. Tous les cadres ont accès au réseau pour utiliser les applications mais la messagerie est réservée aux cadres supérieurs.
2. Pas de communication entre centre de recherche et usines, ni entre usines directement.
3. Un ordre de fabrication envoyé à l'usine ne peut provenir que du centre d'études. Destiné au directeur de production, il est diffusé pour information aux membres de direction de l'usine.
4. Le directeur du centre d'études définit le prix de revient et le communique au directeur commercial du siège (c'est le prix de vente qu'il voit avec le responsable marketing).
5. La connexion vers l'extérieur à l'entreprise est uniquement réservée aux directeurs des six établissements.

Question : Trouver les services de sécurité pour que chaque règle soit satisfaite.

1. Contrôle d'accès par ACL (Access Control List = liste nominative pour l'accès).
2. Contrôle d'accès au niveau du routage avec une gestion centralisée (siège).
3. Il faut assurer l'authentification, la non répudiation, et l'intégrité des données donc on assure ce service par un mécanisme de signature chiffré à clé asymétrique (clé privé pour chiffrer et clé publique pour déchiffrer).
4. Même chose que le 3 + la confidentialité donc les données aussi doivent être chiffrées.
5. Même chose que le 1 (par contrôle d'accès).

EXERCICE: QUESTIONS DE COURS

Comment éviter de faux messages sur un réseau ?

Soit par authentification.
Soit par la mise en œuvre : signature électronique ou chiffrement.

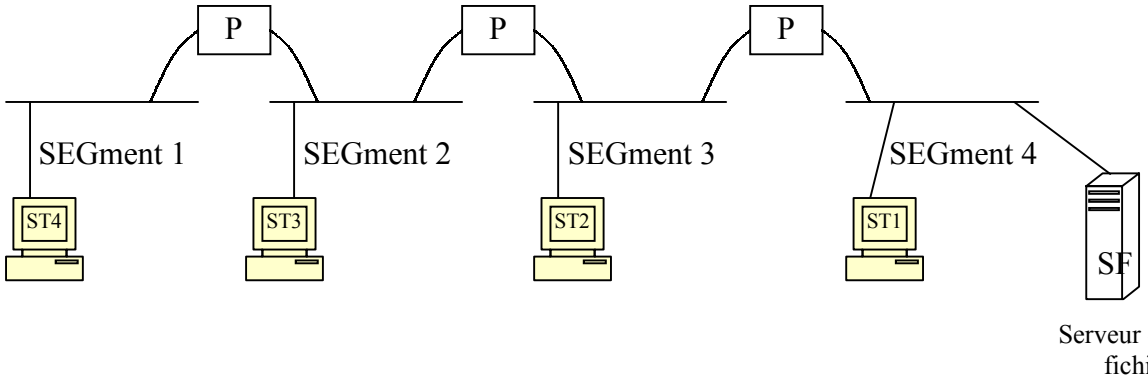
Expliquer pourquoi le chiffrement implémenté sur chaque lien du réseau est plus difficile à mettre en œuvre que le chiffrement de bout en bout ?

Il entraîne un trou de sécurité au niveau de chaque routeur car si l'un des routeurs est sniffé, le codage peut être connu. En plus, il faut diffuser les codes du chiffrement/déchiffrement de manière automatique aux routeurs. On retombe donc sur un mécanisme du type « notaire ».

Par contre si l'on chiffre de bout en bout, le routage se passe exactement comme si les messages n'avaient pas été chiffrés. Mais il faut donc des clés générées de façon aléatoire et transmettre les clés de façon autre (voyage par avion par exemple).

EXERCICE: RESEAUX LOCAUX ETHERNET

Un réseau d'entreprise est constitué de quatre segments Ethernet.



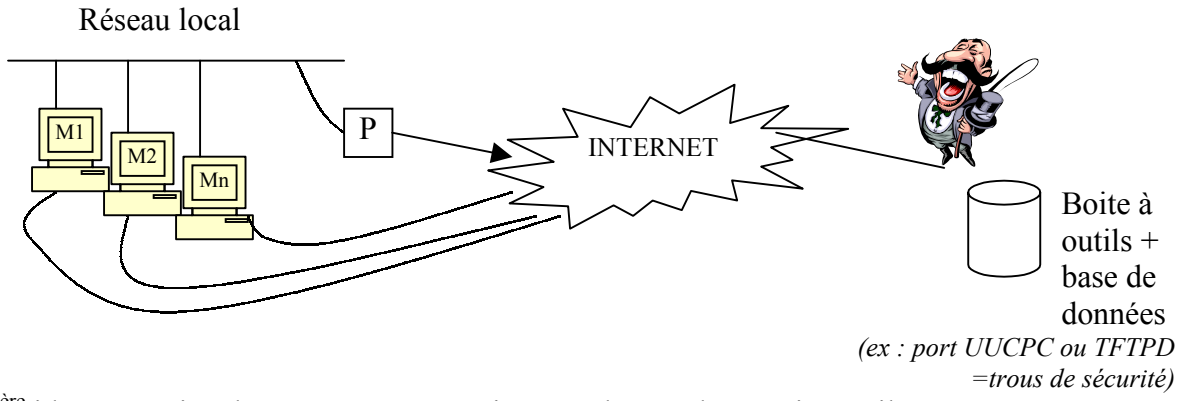
Toutes les stations de SEG4 ont accès à SF (le serveur de fichier) et seulement certaines stations des autres segments ont accès à SF. Quelle solution fiable faut-il mettre en œuvre pour mettre en place une politique fiable de sécurité ?

Il faut déjà remplacer les ponts par des ponts filtrants et fixer la table des adresses MAC (filtrage de niveau 2). De plus en plus, les routeurs sont capables de filtrer au niveau 3 voir 4 du modèle OSI. On peut aussi installer un wrapper à l'entrée du SF (qui intercepte les appels selon une table).

RESEAUX – TD du CNAM BORDEAUX 1999-2000

SECURISATION D'UN RESEAU LOCAL

« Un seul pirate peut faire beaucoup de dégât sur un réseau même s'il y a beaucoup plus de machines ».



- 1ère idée : sécuriser le routeur 1 en autorisant seulement les services utiles.
- Journaliser les transactions (on a le droit d'avoir l'émetteur + l'adresse distante mais pas les données). L'objectif est là aussi de s'assurer des traces pour les recours administratifs.
- Contrôler l'accès (l'idéal est que le routeur ne soit pas configurable à distance par le réseau).

Routage (adresse). Ports : filtrage sur l'adresse, le port ou les données (car certains protocoles ne sont reconnaissable que sur les données). Ceci est fait dans le noyau du système.

I. Firewall (ou garde barrière, pare-feu, ~bastion)

L'objectif est de contrôler (à partir d'une configuration) les accès à un réseau local.

1.1) Définition Générale

Un firewall Internet est un système ou un groupe de systèmes qui renforce la politique de sécurité entre le réseau d'une organisation et Internet. Le firewall détermine:

- quels services internes peuvent être accédés de l'extérieur
- quels éléments externes peuvent accéder aux services internes autorisés
- quels services externes peuvent être accédés par les éléments internes.

Afin qu'un firewall soit effectif, tout le trafic de et vers Internet doit passer par le firewall. Malheureusement, il est important de signaler que le firewall n'offre plus aucune protection une fois celui-ci contourné. Un pare-feu n'est pas seulement un *routeur*, un *bastion host* ou une combinaison de composants sécurisant le réseau. Le firewall est une part intégrante d'une politique de sécurité qui crée un périmètre de défense autour des informations de l'organisation. Cette politique doit inclure un guide de sécurité destiné à informer les utilisateurs de leurs responsabilités, ainsi que des mesures plus classiques de définition d'accès au réseau, aux services, d'authentification des utilisateurs locaux ou non, d'encryptage de données et de protection contre les virus.

1.2) Les avantages du firewall

Les avantages du firewall sont nombreux. Nous pouvons en relever 5 principaux :

1. *Il concentre la sécurité réseau en un seul point (chocke point)* : l'administrateur peut définir un point de centralisation (chocke point) à partir duquel il pourra protéger le réseau privé dans son entier.

2. *Il permet la génération d'alarmes et le monitoring* : le firewall est un système efficace de gestion des accès et de monitoring du réseau. Il est impératif que le responsable réseau évalue régulièrement le trafic qui transite par celui-ci. afin de s'assurer que le pare-feu n'a pas été contourné ou cracké.

3. *Il est l'endroit idéal pour déployer un translateur d'adresse réseau (NAT)* : depuis quelques temps, l'Internet subit une crise due au manque d'extension des adresses IP. Cela signifie que les organisations désirant se connecter à Internet n'obtiennent pas assez d'adresses IP pour réaliser la demande de leur population. Le firewall est la place logique pour monter un NAT (Network Address Transaltor) qui va permettre d'éliminer le besoin de nombreuses adresses IP ainsi que de devoir redéfinir les adresses en cas de changement de provider.

4. *Il peut être la vitrine idéale de l'organisation qu'il défend* : c'est l'endroit parfait pour déployer un serveur WWW ou FTP. Le firewall peut être configuré pour permettre l'accès à ces services tout en prévenant les accès aux autres systèmes du réseau privé.

5. *Il est un point unique de panne* : en cas de panne, le réseau interne privé continuera cependant de fonctionner, seul l'accès inter-réseaux est perdu.

1.3) Les limitations du firewall

Nous pouvons relever 4 principales limitations au firewall :

1. *Il ne protège pas contre les attaques qui ne passent pas par lui*: les utilisateurs du réseau privé peuvent être tentés d'obtenir une connexion directe avec un provider Internet. Ce type de connexion ouvre un chemin d'accès potentiel à l'extérieur.

2. *Il ne prévient pas contre le transfert de fichiers infectés par un virus* : parce qu'il y a une multitude de virus différents, de systèmes d'exploitation et de manières d'encoder et de compresser les fichiers, un pare-feu peut difficilement analyser chaque fichier dans l'espoir incertain d'en déceler un. Il faut donc installer des antivirus sur chaque station.

3. *Il ne prévient pas contre les applications du type Cheval de Troie* : le Cheval de Troie est un programme à l'apparence sans dangers, mais qui se révèle, une fois exécuté, une véritable menace pour l'organisation. Par exemple, un Cheval de Troie exécuté sur une station hôte peut modifier les fichiers relatifs à la sécurité, rendant l'accès plus aisé pour un intrus désirant s'introduire dans le système.

1.4) Les décisions principales lors de la mise en oeuvre d'un firewall

- **La politique de sécurité de l'organisation** : le pare-feu n'est qu'un élément de la sécurité dans une entreprise. Il est donc nécessaire de mener une étude approfondie pour définir une politique cohérente de sécurité.

L'orientation du firewall : l'orientation du firewall décrit la philosophie fondamentale de sécurité de l'organisation. On peut distinguer deux types d'orientations opposées (ces deux orientations sont extrêmes et il est possible de choisir une solution intermédiaire):

  - *Tout ce qui n'est pas spécifiquement permis est interdit* : le firewall empêche tout trafic, tous les services désirés devant être implémentés au cas par cas. Si cette approche procure un environnement sain, elle est lourde et se fait au détriment de la facilité d'emploi.

  - *Tout ce qui n'est pas spécifiquement interdit est permis* : le firewall doit vérifier tout le trafic, tout service potentiellement dangereux étant éliminé cette fois aussi au cas par cas. Cette approche procure un environnement flexible et riche en services offerts. Mais la sécurité est alors beaucoup plus difficile à assurer.

- **Le coût du firewall** Cet aspect est un critère important de la mise en oeuvre d'un firewall. En effet, la complexité de celui-ci peut faire varier les coûts de façon très importante, sans compter la nécessaire maintenance et le monitoring.

Ces analyses permettent de définir les éléments que va requérir le futur firewall. Un pare-feu typique est composé d'un ou de plusieurs des composants suivants: un routeur filtre de paquets, une passerelle de niveau applicatif, une passerelle de niveau circuit. La manière dont ces composants vont se placer par rapport au réseau à défendre va définir le corps du firewall.

1.5) Les Composants des Firewalls

Le routeur filtre de paquets

Le rôle principal du routeur filtre de paquets est de permettre ou d'empêcher le passage des paquets de données reçus. Le routeur examine tous les datagrammes afin de déterminer s'ils ne contiennent pas d'informations dérogeant aux règles de filtrage, règles basées sur le contenu informationnel de l'en-tête du datagramme. L'en-tête contenant entre autre l'adresse IP source et destination, le protocole source et destination TCP/UDP (Transmission Control Protocol/User Datagram Protocol), ainsi qu'un message de type ICMP (Internet Control Message Protocol), il est aisé de vérifier si le paquet est destiné ou provient d'une localisation qui ne serait pas autorisée. Le routeur se positionne au niveau 3 du modèle OSI et le filtrage s'effectue généralement sur les datagrammes TCP, UDP et IP.

Principe de fonctionnement du routeur:

1. Stocker les règles de filtrage sur chaque port physique du routeur.
2. Analyser l'en-tête du datagramme.
3. Appliquer les règles de filtrage aux paquets reçus.
4. Si une règle indique que le paquet doit être bloqué, celui-ci est rejeté.
5. Si une règle indique que le paquet doit être accepté, celui-ci est routé conformément à sa table de routage.
6. Si aucune règle ne s'applique, ce sont des règles par défaut (configurables par l'administrateur) qui seront appliquées. Ainsi, si dans un tel cas on rejette les paquets, on peut bloquer tout trafic non explicitement permis.

Exemple de règles pour Telnet :

Règle	Direction	Adresse source	Adresse destination	Protocole	Port source	Port destination	Ack Set	Action
1	Entrée	155.547.*.*	195.874.12.5	TCP	14	>512	Oui	Router
2	Sortie	195.874.12.6	155.547.*.*	TCP	>512	14	Oui	Router
Défaut	Entrée et Sortie	*.*.*.*	*.*.*.*	Autre	Autre	Autre	Autre	Bloquer

- La règle 1 se lirait donc : "Router (permettre) le trafic en entrée dont les adresses sources débuteraient par 155.547 et dont l'adresse destination serait 195.874.12.5, dont le protocole serait TCP et dont le port source serait 14 avec un port destination supérieur à 512."

- Notons que la création de règles de filtrage est un moment important dans la mise en place d'un routeur et est directement tributaire de la politique de sécurité de l'entreprise. Si les règles définies sont insuffisantes ou mal conçues, le risque d'intrusion est grand, les méthodes de pénétration ne manquant pas. A titre d'illustration, évoquons le *spoofing* et la *fragmentation de paquets*.

Quelques exemples typiques de méthode d'intrusion:

1. *Le Spoofing* : le *spoofing* est une méthode commune d'intrusion. Cette méthode consiste à tromper le routeur sur l'adresse d'origine des paquets transmis : les paquets émis contiennent en fait l'adresse IP d'une machine interne au système que l'on désire pénétrer. Si le système emploie un service de défense simple comme celui d'un routeur, le stratagème a de bonnes chances de fonctionner, d'autant plus que les droits accordés en matière de sécurité dépendent généralement en partie de l'adresse IP. Ce genre d'attaque peut cependant être contourné en bloquant tout paquet qui viendrait d'un port source externe et qui contiendrait une adresse source interne au réseau.

2. *La fragmentation de paquet* : cette technique vise principalement à contourner les règles de filtrage définies par l'utilisateur. Elle consiste à utiliser la propriété de fragmentation de IP afin de créer de tout petits fragments et de forcer l'en-tête TCP à être fragmentée elle aussi, l'espoir étant que seul le premier fragment sera analysé par le routeur, laissant alors passer tout le reste. Ce genre d'attaque peut être contournée en bloquant tout paquet dont le protocole est IP et dont la taille est inférieure à une taille déterminée.

Les avantages du routeur filtre de paquets

Les firewalls, dans leur grande majorité, sont basés sur l'emploi unique de routeurs filtrants. Cela s'explique pour:

1. Le gain de temps
2. Le coût généralement faible d'un routeur filtrant
3. Les performances généralement bonnes
4. La transparence aux utilisateurs et aux applications

Les limitations du routeur filtre de paquets

Cependant, les limites des routeurs filtrants sont mis en évidence par le besoin croissant de sécurité et de contrôle du contenu informationnel des échanges :

1. L'élaboration de règles de filtrage peut être une opération complexe et pénible à partir du moment où l'administrateur réseau doit avoir une compréhension détaillée des différents services Internet et du format des en-têtes des paquets.
2. Le routeur filtre de paquets ne protège pas contre les applications du type Cheval de Troie.
3. Les performances du routeur diminuent avec le nombre de régles à appliquer. Il faut trouver un équilibre entre performance et sécurité.
4. Le routeur filtrant n'est pas capable de comprendre le contexte du service qu'il rend : il ne peut par exemple pas bloquer l'importation de mail, de newsgroup concernant certains sujets.

La passerelle de niveau application

Contrairement au filtrage de paquets par routeur qui se trouve à un niveau assez bas dans le modèle OSI, le filtrage de type applicatif se situe au niveau le plus élevé du même modèle. Ainsi, une passerelle de niveau application (PNA) va permettre à l'administrateur réseau d'instaurer une politique de sécurité beaucoup plus stricte que dans le cas du routeur filtre de paquets. Plutôt que de se reposer sur un simple outil-filtre pour gérer le flot de données provenant d'Internet, un service logiciel au code dédié (appelé aussi **service proxy**) est installé sur la passerelle pour chaque application voulue. Si l'administrateur réseau n'installe pas le service proxy d'une application donnée, le service ne sera pas supporté et ne passera pas le firewall. De plus, le service proxy d'une application donnée peut être configuré de manière à supporter certaines des propriétés spécifiques jugées acceptables par l'administrateur et éliminant l'accès à toutes les autres.

Il est important de signaler que les utilisateurs peuvent avoir accès au service proxy, mais ils ne doivent jamais pouvoir se connecter sur l'adresse IP de la passerelle elle-même. Si on leur permet de se connecter directement sur le firewall, c'est la sécurité même du firewall qui est menacée à partir du moment où un intrus peut potentiellement compromettre la sécurité du système (Cheval de Troie, Spoofing,...).

Exemple de service proxy Telnet

Dans cet exemple, le client extérieur désire utiliser les services Telnet vers un serveur interne protégé par une PNA. Le proxy Telnet ne permet pas à l'utilisateur externe de se connecter ou d'avoir accès au serveur interne. Après authentification de l'utilisateur externe, celui-ci va avoir accès à l'interface utilisateur du proxy Telnet. Le proxy Telnet ne permettant l'utilisation que d'un sous-ensemble des commandes Telnet et n'autorisant l'accès que vers un ensemble d'hôtes internes déterminés, l'utilisateur externe ne sera pas connecté directement au serveur interne désiré. C'est le service proxy qui va s'y connecter et qui va ensuite faire suivre les commandes du client externe vers le serveur interne, le client externe croyant que le proxy Telnet est le serveur interne et inversement.

Le Bastion Host

Contrairement aux routeurs filtres de paquets, une PNA permet le flux d'informations entre systèmes, mais ne permet pas l'échange direct de paquets. Permettre à des paquets de s'échanger entre l'extérieur et l'intérieur d'un système est un risque majeur qui ne peut être pris que si l'application hôte résidant au sein du réseau protégé est sécurisée contre n'importe quel menace potentielle. On fait souvent référence aux PNA comme un Bastion Host qui en est en fait un cas particulier. C'est en effet un système spécifiquement armé et protégé contre les attaques. Certaines règles doivent cependant être respectées quand à la mise en oeuvre d'un Bastion Host :

1. La station d'accueil du Bastion Host doit exécuter une version sécurisée de son système d'exploitation. Cette version sera en effet une version spécifiquement écrite et optimisée contre ses propres faiblesses face aux intrusions possibles.
2. Un service non-installé ne pouvant pas être attaqué, seul les services que l'administrateur considère comme essentiels doivent être installés sur le Bastion Host. Généralement, un ensemble limité de services proxy comme Telnet, FTP, SMTP, DNS ainsi que l'identification utilisateur est installé sur le Bastion Host.
3. Le Bastion Host est l'endroit idéal pour installer un système d'authentification de haut niveau tel qu'une carte d'accès cryptographique à code unique. En plus, chaque service proxy peut demander une authentification utilisateur propre avant de permettre l'accès.
4. Chaque service proxy doit être configuré de manière à ne permettre l'accès qu'à un sous-ensemble des commandes standards de l'application, car, si une commande standard n'est pas supportée par le service proxy, elle ne sera tout simplement pas accessible aux utilisateurs authentifiés.
5. Chaque service proxy est configuré de manière à ne permettre l'accès qu'à un système hôte spécifique, l'ensemble limité des commandes ne pouvant être appliqué que sur une partie du réseau protégé.

6. L'édition des données de connexion, de trafic et de durée de connexion est une partie essentielle de la détection de tentatives d'infiltration. Chaque service proxy doit permettre ce genre de maintenance.
7. Les services proxy doivent être des programmes de petite taille spécifiquement écrits pour la sécurité, afin de permettre une évolution simple et rapide tout en améliorant la détection des bugs et des faiblesses (une application mail UNIX typique contient environ 20000 lignes de code. Son homologue proxy généralement moins de 1000).
8. Les services proxy doivent être indépendants les uns des autres, de manière à éviter de bloquer l'ensemble des services si un problème survenait ou une évolution devait être réalisée.
9. Un service proxy ne doit généralement pas faire d'accès disque autre que la lecture de son fichier de configuration et ce, afin de rendre la tâche plus dure pour une programme d'intrusion.

Les avantages des PNA

1. Les PNA donnent à l'administrateur un contrôle complet sur chaque service par la restriction sur les commandes utilisables et sur l'accès aux hôtes internes par ces services.
2. A partir du moment où un service proxy absent pour un service donné bloque complètement ce service, l'administrateur a un contrôle parfait des services qui seront disponibles de ceux qui ne le seront pas.
3. Les PNA permettent l'installation de procédures d'authentification extrêmement poussées.
4. Les services proxy facilitent grandement l'audit des détails des connexions.
5. Les règles de filtrage d'une PNA sont bien plus faciles à configurer et à tester que pour un routeur filtre de paquets.

Les limitations des PNA

1. Les PNA augmentent considérablement le coût du firewall ; ce coût étant principalement lié à celui de la plate-forme matérielle de la passerelle, des services proxy et du temps et des connaissances requises pour configurer cette passerelle.
2. Les PNA ont tendance à réduire la qualité du service offert aux utilisateurs tout en diminuant la transparence du système.
3. L'utilisation d'une PNA force l'utilisateur à changer ses habitudes ou à installer des logiciels spécialisés sur chaque système accédant à des services proxy. En effet, les accès Telnet requièrent deux étapes de connexion plutôt qu'une. Cependant, on peut aussi installer des logiciels spécialisés qui améliorent la transparence du PNA en permettant à l'utilisateur de spécifier la destination plutôt que la PNA via un commande Telnet.

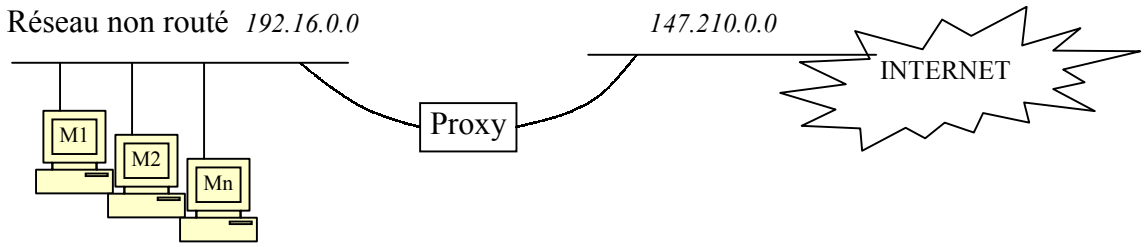
La passerelle de niveau circuit

Une passerelle au niveau circuit (PNC ou **serveur proxy**) est une fonction spécialisée qui peut être réalisée par une PNA. Une PNC est un simple relais de connexions TCP sans manipulation ou filtrage de paquets. La PNC relaie la connexion à travers le firewall sans faire d'examens supplémentaires, de filtrage ou de gestion de protocole. La PNC agit comme un fil, copiant telles quelles les données de l'extérieur vers l'intérieur (ou inversement). Cependant, les connexions semblent être originaires du firewall, ce qui a pour effet de cacher complètement les informations concernant le réseau protégé. Ce genre de passerelle est souvent utilisée pour les connexions sortantes lorsque l'administrateur à confiance en ses utilisateurs. Le principal avantage est qu'un Bastion Host peut être configuré comme une passerelle hybride supportant des services proxy pour les connexions en entrée et des fonctions du niveau circuit pour les connexions en sortie. Cela rend l'utilisation du firewall plus facile d'emploi pour les utilisateurs qui désirent un accès direct à Internet, tout en permettant une protection efficace contre les attaques de l'extérieur.

Notons enfin que c'est l'endroit idéal pour le déploiement d'un Network Adress Transaltor, tout comme c'est aussi une protection très efficace contre le spoofing, les adresses internes étant inconnues de l'extérieur.

II. Réseaux Privés NAT (Network Address Translation)

Définition d'un adressage non routé. RFC 1918

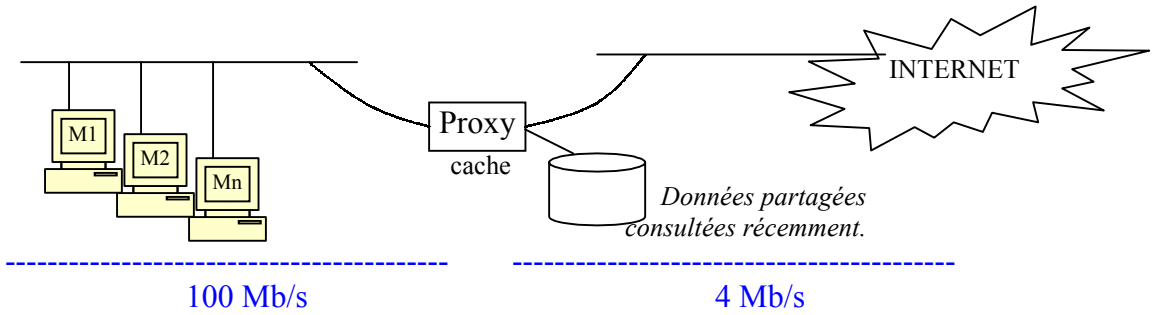


Il faut au proxy une table de translation qui contient :

- Adresse d'origine
- Adresse de destination
- Liaison

III. Proxy - CACHE

Un proxy est une machine et du logiciel (par service). Il y a une différence avec le cache. L'idée du cache est d'économiser de la bande passante par mutualisation des transferts pour un groupe de machines. Mais cela peut poser un problème avec les pages dynamiques (contenu qui change à chaque fois).



NB: les services privilégiés sont ceux dont le n° de port < 1024 (les autres ont leur port >1024).

EXERCICE: Architecture de RESEAU LOCAL SECURISE

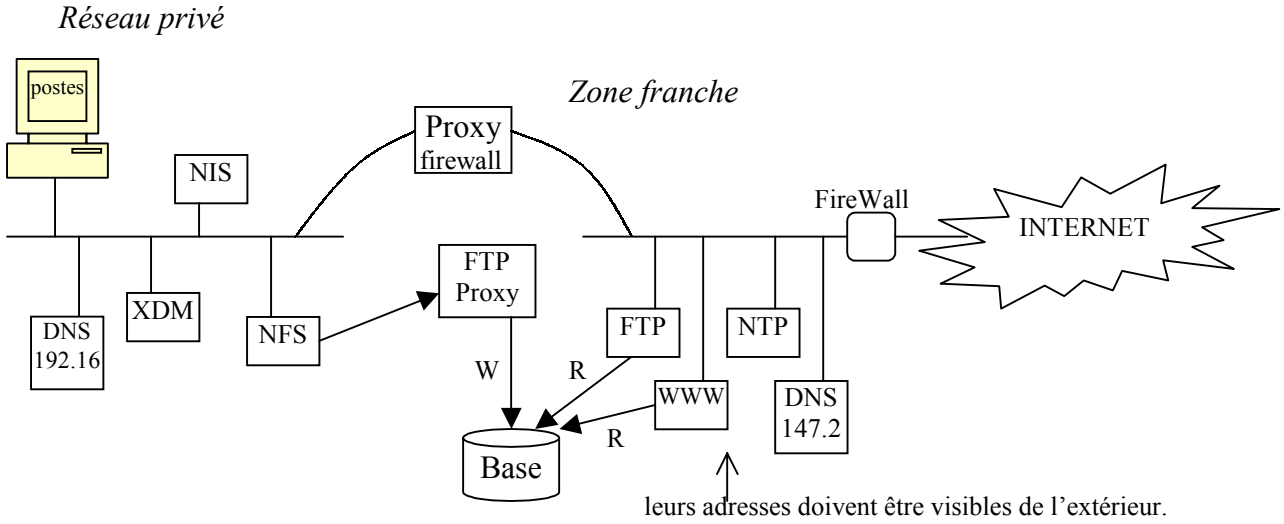
Concevoir une architecture de réseau local sécurisé en plaçant les différents composants :

- 50 postes de travail,
- 1 serveur WWW (Web),
- 1 serveur FTP anonyme (serveur de distribution de fichiers)
- 1 serveur SAMBA / NFS (fichiers partagés)
- 1 serveur DNS (de noms)
- 1 serveur NIS (de configuration)
- 1 serveur NTP (de temps)
- 1 serveur XDM (de terminaux X11)

Objectif : assurer une sécurité « raisonnable ». En cas de fraude :

- aucune modification des données
- identification de l'intrus
- fichiers de configuration invisibles de l'extérieur
- fichiers locaux invisibles de l'extérieur
- postes de travail invisibles de l'extérieur mais y peuvent y accéder
- le serveur de temps se synchronise à l'extérieur

Doivent être visibles de l'extérieur : serveurs FTP, DNS (peut être 2 serveurs), NTP mais unidirectionnel et le WWW.



Le DNS n'est habilité à communiquer qu'avec le DNS du domaine supérieur (ex : ftp.mit.ai.edu).

RESEAUX – TD du CNAM BORDEAUX 1999-2000

PROTOCOLES TRANSPORT SECURISES

Avant toute chose pour bien comprendre où se situe le problème, il faut commencer par définir précisément chacun des termes **protocoles**, **transport** et **sécurisé**. Si on cherche dans un dictionnaire Larousse, on trouve les définitions suivantes.

- **Protocole** : ensemble de règles observées en matière d'étiquette, de présence...
- **Transport** : action de transporter : c'est à dire de porter d'un lieu à un autre.
- **Sécurité** : situation où on n'a rien à craindre.

Un protocole de transport sécurisé permet donc de **porter** quelque chose (ici il s'agit **d'information**) d'un lieu à un autre suivant des **règles** prédéfinis **sans** que l'objet transporté ne soit jamais en **danger**. A priori on peut traiter le problème dans tous les domaines (transport ferroviaire, bancaire...). Dans cet exposé nous allons présenter quelques techniques (liste non exhaustive) permettant de répondre à certaines exigences que nous allons préciser dans la section suivante : Exigences de la sécurité informatique.

Etant donné que la majorité des réseaux utilisés à travers le monde sont de type TCP/IP et que le choix des entreprises se portent souvent vers ce type de réseaux lorsqu'elles décident de se connecter à l'Internet, nous avons choisi de présenter les solutions proposées pour ces types de réseaux.

I. Exigences de la sécurité informatique

Comme le souligne Andrew Tanenbaum: les problèmes de sécurité des réseaux peuvent être classés en 4 catégories non disjointes :

1. La **confidentialité** : lors d'un transfert d'informations entre deux entités à travers le réseau, cette information ne doit pas être lisible par une troisième partie. Un exemple type est le transfert de courrier électronique à travers le réseau.

2. L'**authentification** : lorsque deux entités dialoguent à travers le réseau il faut qu'elles soient sûres de l'identité de l'autre, ceci pour éviter toute usurpation. Une telle exigence se fait particulièrement sentir lors de transactions commerciales.

3. La **non-répudiation** : elle concerne les signatures. Il faut être sûr qu'un client qui passe une commande ne va pas se rétracter au dernier moment.

4. Le **contrôle d'intégrité** : il faut pouvoir être sûr que l'information n'a pas été modifiée par une tierce personne lors d'un transfert d'informations.

Il faut bien voir que toutes ces exigences ne sont pas disjointes. Mais il n'est cependant pas nécessaire de toutes les remplir. En effet dans le cas où seule l'authentification est nécessaire, on peut s'affranchir de la confidentialité qui peut être coûteuse à mettre en oeuvre (les algorithmes de cryptage peuvent être lents).
Nous allons voir rapidement comment, **théoriquement** ces exigences peuvent être satisfaites.

1.1 Confidentialité

Ce problème peut être résolu en utilisant des techniques de *cryptographie*. La cryptographie est un ensemble de techniques qui permettent de transcrire dans une écriture secrète un texte clair. Les algorithmes les plus utilisés sont les suivantes :

- DES : algorithme à clé secrète développé par IBM en 1977.
- IDEA algorithme à clé secrète.
- RSA inventé par Rivest, Shamir et Adleman en 1978.

1.2 Authentification

Elle peut être garantie par grâce à différentes techniques. Une première consiste à utiliser un algorithme à clé publique. On rappelle qu'un tel algorithme possède deux clés une est publique c'est à dire que tout le monde peut y avoir accès et l'autre est secrète : c'est à dire qu'elle n'est connue que par une seule personne ou une seule organisation. Ce type d'algorithme vérifie les trois conditions suivantes :

- Décryptage(Cryptage(Texte clair, clé publique), clé privée) = Texte clair
- La clé privée est très difficile à déduire de la clé publique.
- La clé privée ne peut pas être cassée par une attaque de type "texte clair choisi".

On peut utiliser ce type de cryptographie pour avoir un moyen authentifié de s'échanger des informations entre deux entités A et B de la façon suivante :

1. A crypte un nombre aléatoire RA de son choix avec la clé publique de B et lui envoie le tout.
2. B crypte RA, un autre nombre RB et une clé KS avec la clé publique de A et lui envoie le tout.
3. Finalement A renvoie RB chiffré avec la clé KS

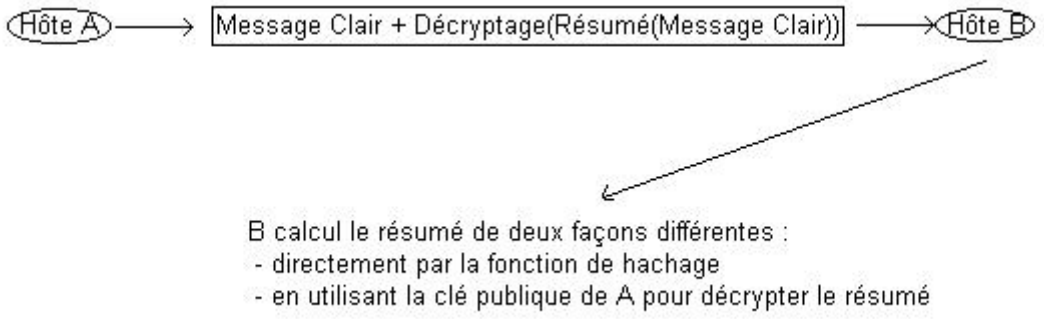
Cet échange a ainsi permis à A et B d'obtenir une clé commune (à priori secrète) KS. Une autre approche consiste à utiliser le protocole Kerberos. On peut aussi construire une clé secrète partagée grâce au protocole d'échange de clé de Diffie Hellman.

1.3 L'intégrité et non répudiation

Il est souvent difficile de découpler ces deux aspects des deux précédents. Nous allons présenter plusieurs techniques qui possèdent ces propriétés parfois mêlées avec les précédentes. Ces techniques sont des techniques de **signature numériques** : d'ailleurs si on réfléchit bien on peut se rendre compte que les propriétés vérifiées par ces signatures sont exactement les mêmes que celles des signatures classiques sur papier. Voici ces propriétés :

1. La signature numérique doit garantir d'une part **l'identité de l'émetteur** vis à vis du récepteur.
2. Elle doit également empêcher l'émetteur de **renier** le contenu du message.
3. Enfin il ne faut pas que le récepteur puisse **contrefaire** cette signature et ainsi attribuer la provenance des messages à un autre émetteur potentiel.

Une technique pour obtenir une signature numérique et celle qui utilise les résumés de messages (**messages digests** en anglais). Le résumé de message est obtenu en sortie d'un algorithme de hachage. Cet algorithme prend un texte de longueur variable en entrée et sort une séquence de bit de longueur fixe. La particularité d'un tel algorithme est qu'il est impossible de générer deux messages qui ont la même valeur de sortie pour l'algorithme de hachage. Si de plus on dispose d'un algorithme à clé publique qui vérifie la propriété suivante : Cryptage(Décryptage(Texte clair)) = Texte clair alors on peut procéder de la façon suivante :



Ce type d'échange rempli les trois propriétés précédemment énoncées : B pourra décrypter le résumé de message (avec la clé publique de A) et le comparer avec celui qu'il calculera lui-même, il pourra ainsi être sur que le message provient bien de A et qu'il n'a pas été modifié. Cela garantit également la non répudiation puisque A est la seule entité qui possède DA. Les deux algorithmes les plus connu dans ce domaine sont les algorithmes MD5 et SHA.

II. Solutions sécurisées du monde TCP/IP

Dans cette partie nous allons décrire différentes solutions pour protéger un réseau. Etant donne que les communications sur les réseaux sont organisées en couche (Cf. modèle OSI et modèle TCP/IP) il est tout naturel de retrouver des solutions a chacun de ses niveaux. Dans la suite de cet exposé nous allons détailler des solutions de niveaux **réseaux** (IPsec), **transport** (SSL) et **application** (S-HTTP, HTTP 1.1).
Voici un petit schéma récapitulatif des solutions sécurisées:

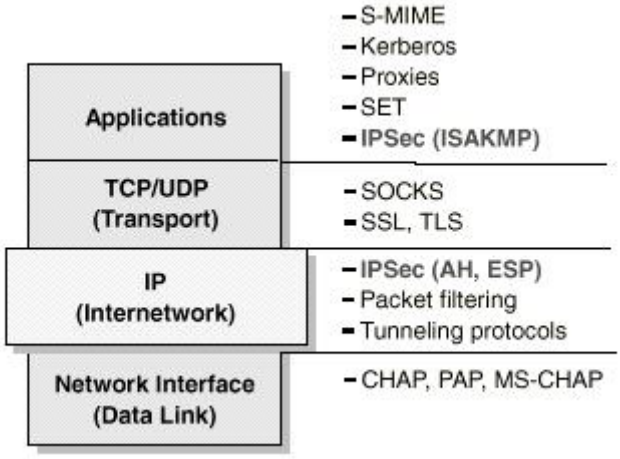


Figure 1

2.1 IPSec

Comme son nom l'indique, il s'agit d'une solution de niveau réseau. Dans cette partie nous allons étudier l'architecture de IPSec et trois de ses principaux composants : **AH** (**A**uthentication **H**eader), **ESP** (**E**ncapsulating **S**ecurity **P**ayload) et **IKE** (**I**nternet **K**ey **E**xchange). IPSec est totalement compatible avec les infrastructures IP existantes. Lorsqu'il est correctement implémenté, il n'affecte en rien les réseaux ou les hôtes qui ne le supporte pas. IPSec est indépendant des algorithmes courants de cryptographie. Il fonctionne avec Ipv4 et Ipv6.
Deux concepts majeurs de IPSec doivent être détaillés avant d'entrer dans les détails: **SA** (Security Associations) qui rend compte d'une connexion avec un hôte distant :
<Security Parameter Index, IP Destination Address, Security Protocol [AH ou ESP]>
Le deuxième concept est l'**encapsulation** et elle consiste à enfermer un paquet IP classique dans une autre paquet :
| *Champ de données* | *Entête IP* | *Nouvelle entête IP* |
|<--->|
Datagramme originel
Ces deux concepts ne sont pas nouveaux, en fait ils sont juste utilisés par IPSec.

2.1.1 L'entête d'authentification (AH)

AH est utilisé pour fournir l'**intégrité** et l'**authentification** pour les datagrammes IP. AH authentifie autant de paquets IP que possible. Certains champs dans l'entête IP changent en cours de route et leur valeur ne peut être prédis par le destinataire. Ces champs sont appelés *mutable* et ne sont pas protégés par AH. Pour Ipv4, il s'agit des champs suivants : type de service, flags, fragment offset, TTL, header checksum. Les paquets qui échouent à l'authentification sont rejetés et ne sont jamais délivrés aux couches supérieures. Ce mode d'opération réduit grandement les chances de succès des attaques visant à bloquer la communication d'un hôte ou d'une passerelle en l'inondant de paquets corrompus. La position de l'entête AH dans un paquet IP est la suivante :
| *IP entête* | ***AH entête*** | *Champ de données* |
Il existe deux façons d'utiliser cet entête.
AH en mode normal : dans ce mode l'entête AH est placé à droite de l'entête IP comme décrit ci-dessous:
| *Entête IP* | ***Entête AH*** | *Champ de données* |
Il est utilisé par les hôtes et non par les passerelles. L'avantage de ce mode réside dans un traitement réduit de l'entête. Le désavantage est que les champs mutables ne sont pas authentifiés.
AH en mode tunnel: avec ce mode le concept d'encapsulation est appliqué, un nouveau paquet IP est construit et le paquet IP d'origine joue le rôle du champ de données de ce nouveau paquet. Puis le AH en mode transport est appliqué au nouveau paquet, ce qui donne l'organisation suivante :
| ***Nouvelle Entête IP*** | ***Entête AH*** | *Entête IP* | *Champ de données* |

Ce mode est systématiquement utilisé entre deux firewalls. En mode tunnel, l'adresse IP de l'entête extérieure peut être différente de celle intérieure. Ainsi deux passerelles de sécurité peuvent réaliser un tunnel AH qui servira à authentifier le trafic entre les réseaux auxquels elles sont toutes deux reliées. Ceci est un exemple d'utilisation typique.
L'avantage de ce mode est qu'il assure une totale protection du datagramme IP encapsulé et la possibilité d'utiliser des adresses privées. Néanmoins, un lourd traitement des entêtes est associé à ce mode et une augmentation de la taille des paquets.

2.1.2 Encapsulating Security Payload (ESP) :

ESP est utilisé afin de fournir des services tels que le contrôle d'**intégrité**, l'**authentification** et le **cryptage** des datagrammes IP. Ces services ne nécessitent pas de connections, ils opèrent sur la base des paquets. Ils sont sélectionnés lors de l'établissement de la SA. Toutefois certaines restrictions doivent être apportées :

- Le contrôle d'intégrité et l'authentification vont ensemble.
- La protection contre les attaques par rejeu est indissociable du contrôle d'intégrité et de l'authentification.
- La protection contre les attaques par rejeu ne peut être sélectionnée que par le destinataire.

On peut choisir le type de cryptage indépendamment des autres services. Il est recommandé que lorsqu'on utilise ce service, les deux autres services soient aussi activés sinon un intrus peut falsifier des paquets pour réaliser des attaques contre le cryptage ; ce qui n'est pas possible lorsque les deux autres services fonctionnent en même temps.
Même si l'authentification et le cryptage sont optionnels, au moins un des deux est toujours sélectionné, sinon l'utilisation d'ESP n'aurait aucun sens.

Le traitement d'ESP s'applique uniquement aux paquets non fragmentés. Toutefois, un paquet IP utilisant ESP peut être fragmenté par des routeurs intermédiaires. Dans ce cas, le destinataire réassemble d'abord le paquet avant d'appliquer le traitement ESP. Pour les mêmes raisons qu'avec AH, tout paquet IP fragmenté arrivant en traitement ESP est rejeté.

Si le cryptage et l'authentification (avec le contrôle d'intégrité) sont sélectionnés tous les deux, alors le destinataire authentifie d'abord le paquet puis à condition que cette étape soit réussie, procède au décryptage. Ce mode d'opération protège les ressources du système et réduit la vulnérabilité aux attaques nuisibles.
Comme pour l'entête d'authentification il existe encore deux façons d'utiliser cet entête.
ESP en mode normal : L'organisation du nouveau paquet est la suivante:
| *Entête IP* | ***Entête ESP*** | *Champ de données* | ***ESP trailer*** | ***ESP authentification*** |
Ce mode ne fournit ni cryptage ni authentification pour l'entête IP. C'est un désavantage car les paquets erronés sont envoyés en traitement ESP.
Comme dans le cas de AH, ESP en mode transport est utilisé par les hôtes et non par les passerelles.
ESP en mode tunnel : Ce mode utilise le concept d'encapsulation. Un nouveau paquet IP est créé par l'insertion d'une nouvelle entête IP au paquet initial. Puis le mode transport est appliqué à ce nouveau paquet, d'où l'organisation du paquet final:
| *Nouvelle* entête IP| ***Entête ESP*** | *Entête IP* | *Champ de données* | ***ESP trailer*** |***ESP authentification***|
Ce mode tunnel est utilisé dès qu'une extrémité de l'association sécurité est une passerelle. De la même façon qu'avec AH, il n'est pas nécessaire que les deux adresses IP contenues dans le nouveau paquet soient identiques. De même, ce mode assure une totale protection du datagramme IP et permet l'utilisation d'adresses privées.

2.1.3 Pourquoi deux protocoles d'authentification ?

En effet on peut se demander si AH est vraiment nécessaire. En fait, il n'y a pas de réponse officielle, seulement quelques points qui pourraient justifier l'existence des deux protocoles:

- ESP nécessite d'importants algorithmes de cryptage. Mais le cryptage des données reste un sujet sensible dans certains pays où des règles strictes ont été mis en place et où il serait donc difficile d'implémenter des solutions à base de ESP. Néanmoins l'authentification n'est pas réglementée et AH peut être utilisé librement partout.
- Souvent l'authentification est seule nécessaire, AH est alors plus performant que ESP à cause de son format plus simple et d'un traitement de l'entête moins coûteux. Dans ces cas-là, il est plus sensé d'utiliser AH.
- Avoir deux protocoles signifie un contrôle plus fin d'un réseau IPSec et des options de sécurité plus flexibles.

2.1.4 Protocole d'échange de clé IKE

Comme expliqué dans IKE est un protocole qui permet la négociation automatique des Security Association (vue précédemment) ainsi que la génération et la mise à jour automatique de clés de cryptage. Ce protocole utilise les protocoles d'échange de clés ISAKMP, Oakley et SKEME.
Il est composé de deux phases principales :

- La première phase consiste principalement en un échange de clé de Diffie Hellman. La clé ainsi obtenue permet l'authentification des deux parties qui vont dialoguer. C'est la phase la plus coûteuse du protocole.

- La deuxième phase permet d'échanger des informations concernant les Associations Sécurisées (AS défini plus haut) ainsi que les clés qui seront utilisées par la suite.

La première phase à lieu une fois sur une période assez longue (une journée, une semaine), tandis que la deuxième peut avoir lieu toutes les 3 ou 4 minutes.

2.2 SSL

Ce protocole a été développé par Netscape. Comme l'indique la figure 1, il s'agit d'une solution de niveau **transport** dans la pile TCP/IP. Le principal but de cette couche est de remplacer une couche transport classique de façon transparente dans les applications utilisant HTTP, FTP, TELNET... Il remplit les trois contraintes expliquées plus dans ce document : **confidentialité** (assurée par l'utilisation d'algorithme d'un algorithme de cryptage DES ou IDEA ou triple DES...), **authentification** (du serveur et du client), l'**intégrité** (grâce à l'utilisation d'un résumé de message qui protège les données).

Elle est principalement constituée de deux protocoles : le protocole d'échange de données (**SSL Record Protocol**) et le protocole de "poignée de main" (**SLL Handshake Protocol**). Ce dernier est utilisé au début du dialogue entre les deux entités qui communiquent à travers SSL. Il permet l'authentification des deux parties : c'est une étape d'**initiation** du dialogue. Le protocole d'échange de données permet d'échanger des informations de façon confidentielle et authentifiée après l'étape d'**initiation**.

Le **protocole d'échange de données** définit le format des informations transmises lors d'échanges de données entre deux entités sur le réseau. Les données échangées sont séparées en deux parties : un entête et une partie de données. Le format simplifié se présente de la façon suivante :

- Entête
    - La longueur de la partie concernant les données
    - Un champ taille de complétion qui contient la taille du champ complétion (padding) de la partie données (cf. deuxième point).
- Données
    - Une partie résumé de message qui permet de garantir l'intégrité des informations véhiculées.
    - Une partie dédiée aux données (cryptées)
    - Une partie de complétion (car les algorithmes de hachage impose parfois que la longueur du texte soit un multiple d'une certaine quantité ex : 512 octets pour MD5).

Le **protocole "handshake"** est composé de deux phases. La première permet l'établissement d'un canal de communication sécurisé tandis que la deuxième permet d'authentifier l'identité du client.
SSL est concrètement implanté dans la navigateur Netscape Communicator et par les serveurs du site de Netscape. Pour pouvoir utilisez ce type de service les URLs utilisée sont de la forme https://... Si vous possédez Netscape Communicator et que vous cliquez sur https://developer.netscape.com/docs/manuals/security.html alors vous accéderez à une page obtenu grâce à SSL (un message vous préviens que les données qui vont être échangées entre le serveur et votre navigateur seront cryptées : on peut obtenir des informations complémentaire sur cette page en cliquant sur le menu **View** puis **Page Info**).

2.3 S-HTTP et HTTP 1.1

S-HTTP un protocole de niveau application qui est une extension de HTTP. Il offre des fonctionnalités de **chiffrement**, la possibilité d'intégrer des **signatures** ainsi que des mécanismes d'**authentification**. HTTP 1.1 offre un mécanisme d'**authentification** appelé (Digest Access Authentication en anglais) cette méthode permet une authentification simple des données HTTP mais ne permet de les chiffrer. Cette solution est beaucoup moins sûre que SSL ou SHTTP. Comme aucune possibilité de chiffrement n'est offerte une attaque par rejeu est inutile : en effet pourquoi quelqu'un voudrait-il récupérer une page qu'il peu lire en clair ? Par contre HTTP 1.1 est vulnérable à l'attaque du passeur de saut.

Globalement on peut dire que le mécanisme d'authentification Digest Access Authentication permet de palier à quelques énormes failles du protocole d'authentification déjà utilisé dans HTTP 1.0 (Basic Access Authentication) mais qu'il ne constitue en rien un protocole sûr.

2.4 Autres

TLS : il s'agit d'un protocole sensiblement identique à SSL 3.0.
SET : il s'agit d'un protocole utilisé pour les transactions bancaires (**S**ecure **E**lectronic **T**ransactions) qui est né d'un accord entre *MasterCard International*. Il a été conçu dans le but d'uniformiser les transactions bancaires par carte de crédit. En effet avant cela chacune de ces entreprises possédait son propre protocole sécurisé.

IV. Conclusion

Tout au long de ce document nous avons vu différentes solutions qui permettent d'échanger des informations à travers Internet de façon sécurisé. Une question semble se poser : quelle solution adopter pour quels besoins ? IPSec semble être une solution lourde à mettre en oeuvre car elle nécessite souvent des traitements spécifiques des entêtes supplémentaires au niveau des routeurs. Elle est donc plus destinée pour des grandes entreprises ou de grands organismes. La solution SSL quant à elle semble plus simple à mettre en oeuvre : en effet cette technologie est accessible à tous à travers Netscape et l'interface de programmation ressemble à l'interface des Sockets Unix(en fait il est battit au dessus de ces sockets).

Les solutions de niveau application quant à elles semblent plus marginales et moins utilisées. Elle commencent à être intégrés petit à petit dans le protocole HTTP.

EVALUATION DE RESEAUX ET PROTOCOLES

RESEAUX – Cours du CNAM BORDEAUX 1999-2000

EVALUATION DE RESEAUX ET DE PROTOCOLES DE COMMUNICATION

Il y a deux approches pour l'évaluation : qualitative et quantitative (évoluer les performances de débit, temps de réponse…).

IX. EVALUATION QUALITATIVE

1.1) Modèles formels

L'objectif est d'obtenir un modèle qui permet de calculer des propriétés sur le protocole.

a) Langages et grammaires formels

Les messages du protocole sont des phrases de la grammaire. Pour définir un message :

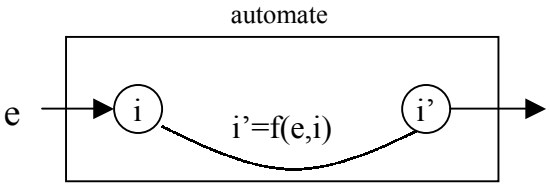
Se réecrit en variables
<message> ::= <entête> <texte> <fin> | <fin de transfert>
<entête> ::= 0 | SOH <bloc>
<texte> ::= STX <bloc> ETX (terminaux ou jeton)
<fin> ::= <caractère> <caractère>
<bloc> ::= <caractère> <bloc> (récursif)
<fin de transfert> ::= EOT
<caractère< ::= SOH |….| EOT |….| |….|a|….|Z| DEL
(256 codes possibles)

Intérêt : on sait que la machine sera certaine de la signification du message. L'analyse syntaxique permet de valider ou de voir des erreurs du message.

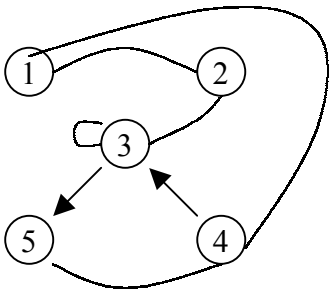
b) Description de machines

Description des machines + interaction entre elles.
Modèle « flow charts » - Description des flux.

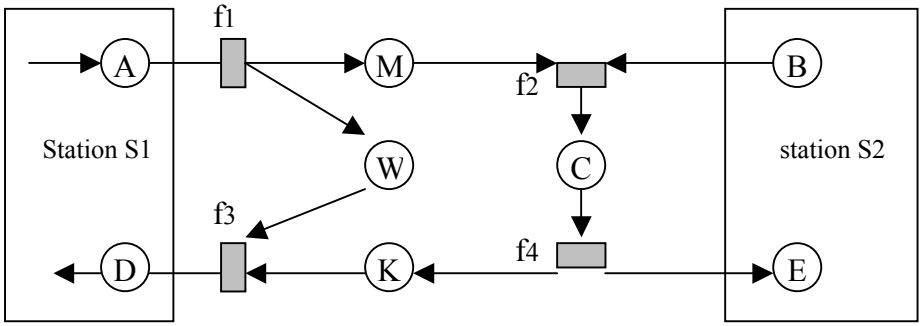
c) Automates d'états finis



	C	•	*	)	autre
1	2	-	-	-	-
2	-	-	3	-	-
3	3	3	5	3	3
4	1	3	1	1	1
5	-	-	-	4	-



d) Réseaux de Petri (MIT 1972)



A : la station S1 a un message à envoyer à S2
M : la station S1 a envoyé le message
W : la station S1 attend un accusé de réception
B : la station S2 est prête à recevoir un message
C : la station S2 a reçu un message.
D : la station S1 sait que le transfert du message est terminé.
K/E : S2 a informé S1 de la réception du message par l'émission d'un accusé de réception.

1.2) Techniques de description formelle de protocole

SDL : *Spécification &Description Language* (textuel + graphique).
Normalisé en 1976 par le CCITT

Estelle : Normalisé en 1988 par l'ISO. Principe de stimuli-réaction. Description en Pascal.

LOTOS : Ordonnancement temporel des interactions. On va faire un assemblage de « morceaux de communications ». Série, parallèle, synchro…

EXAMEN DU 13/09/1997

Machine parallèle

a)

- Faire des liaisons entre tous les PCs (si 16 Pcs → 15 interfaces par PC et 120 liaisons en tout ! !) **Erreur !**
- Switch à 100 Mbits

Calculer débit binaire (octets/seconde) et latence.
Latence minimum sur trame de longueur minimum ? ?

→ 1 trame + préambule + SFD + @ destination
→ 64 + 7 + 1 + 6
→ 78 octets → 78 octets * 8 bits * 10^{-8} 10Mbits = 6,24 µs

Latence normale ? ? 2 trames → 128 octets = 10,24 µs

EXAMEN DU 06/06/1998

a)

- 1 réseau local
  - adresser une machine
  - ouvrir un canal de communication
- 1 protocole de communication
  - établir un liaison
  - transmission des paramètres par valeur (dans les deux sens)
- synchronisation
  - sémaphores ou jetons/tubes (plus pratique)

description de la solution

- parallèle (tube, calcul, P1, P2, P3)
  - créer ou accéder au tube
  - trouver une machine la moins chargée (dans une liste)
  - transférer le calcul sur la machine
  - transférer les paramètres (dire à la machine distante où renvoyer les résultats)
  - transférer le nom du tube
  - lancer le calcul

- parallele_d (daemon)
  - attente de connexion
  - récupération de :
    - tube
    - calcul
    - paramètre
  - lancer le calcul
  - renvoyer les paramètres
  - déposer un jeton dans le tube

b) attente au point de rendez vous → attente passive

c)

- remettre un jeton dans le tube
- arrêter le calcul complet
- relancer le sous calcul sur une autre machine
- lancer en parallèle et attendre la première réponse

la fiabilité va être de recommencer un calcul qui s'est mal passé ou de donner un maximum de tentatives possibles.

d) SNMP → informations collectées par une station sur toutes les machines du réseau.

EXAMEN DU 19/09/1998

1.1 codage Manchester
oui (câble coaxial)

1.2 25 * 624 * 600 = 10 Méga pixels

1.3 entête etc…(56 bits à zéro pour indiquer le début) cf. P48

1.4 environ la moitié de 10 Mbits (a et 6 Mbits/s acceptés)

1.5 sur ethernet, il y a un cheksum pour détecter les erreurs
- code auto correcteur qui peut lui même être erroné donc il faut l'écarter.
- numérotation des trames et faire de la redondance.

1.6 Répéter plusieurs fois de suite le fichier ou répéter plusieurs morceaux du fichier de suite…

1.7 maximum : taille du fichier. Si redondance de 1 Mo, taille = 1 Mo

1.8 hypothèse : si bande passante 10 Mb/s → bande passante 5 Mb/s
10^6 caractères →

1518 * 8 bits =	12144
-26	+ 10 bits
1492	12154

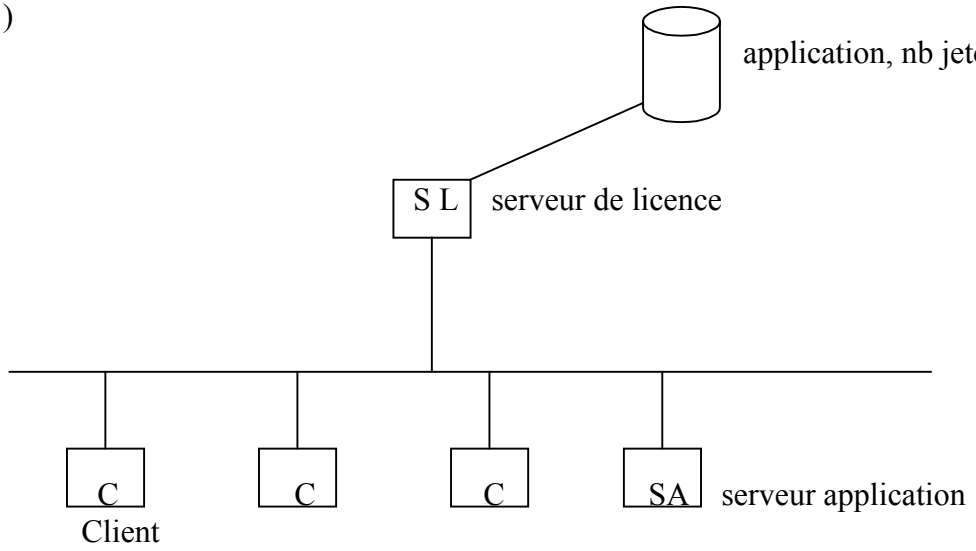
↑
débit utile

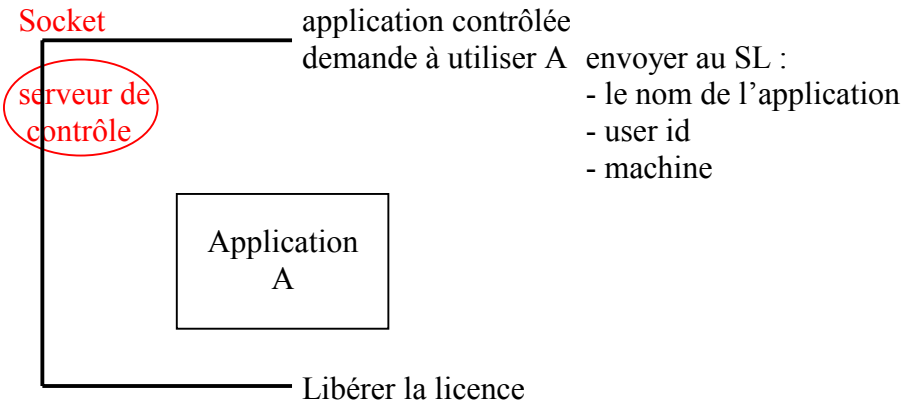
⇨ **Erreur !**= 41 trame/sec
⇨ 41 * 1492 = 61 Ko/s
⇨ soit 16 secondes pour 1 Mo

1.9 On réutilise ces champs pour numéroter les paquets (3 octets du numéro et 3 octets du nombre total de trames à transmettre). On envoi des infos directement affichables sur l'écran du micro.

EXAMEN DU 31/05/1997

1)





4)

cf au dessous en rouge

Connexion à un port socket de la machine cliente par le serveur de licence pour vérifier que la connexion est toujours OK.

5)
Chiffrage des transactions, par utilisation de clés à usage unique par exemple. Utilisation de clé publique (pour coder) par les clients et de clé privée pour le serveur de licence (pour décoder) + retour du serveur vers le client (clé asymétrique).

Pour éviter à un SL pirate de s'installer
sur le réseau en saturant le SL d'origine.