Réseaux IP

Renaud Rioboo

1992-2014

X Window, un exemple d'application réseau

X Window est une application qui permet d'utiliser des écrans de type graphique, et de faire tourner des programmes sur d'autres machines en ayant la visualisation sur son écran.

Par exemple si notre écran est celui d'une machine **Serv** donnée, on peut faire en même temps :

- de l'édition de graphiques xfig (machine **CI1**)
- de l'édition de texte emacs (machine **CI2**)
- de la gestion de courrier xmh (machine **CI3**)

Partage de fichiers

Les applications comme NFS, PC-NFS, Apple File Exchange, AUFS permettent de partager des ressources de disques entre plusieurs machines.

On a alors accès depuis son Macintosh, son PC, voire sa station de travail de façon transparente à des fichiers qui sont stockés sur le disque physique d'une autre machine, qui n'a pas nécessairement le même système d'exploitation.

On peut ainsi travailler sans se soucier de l'endroit où sont stockés ses fichiers.

Les couches **ISO**

L'**ISO** (International Standards Organization) définit 7 niveaux de fonctionnalités (couches) **OSI** (Open Systems Interconnexion) qui définissent des normes pour la communication entre machines.

Ces normes permettent de dégager la notion de fonctionnalité logique en classant les couches par niveaux.

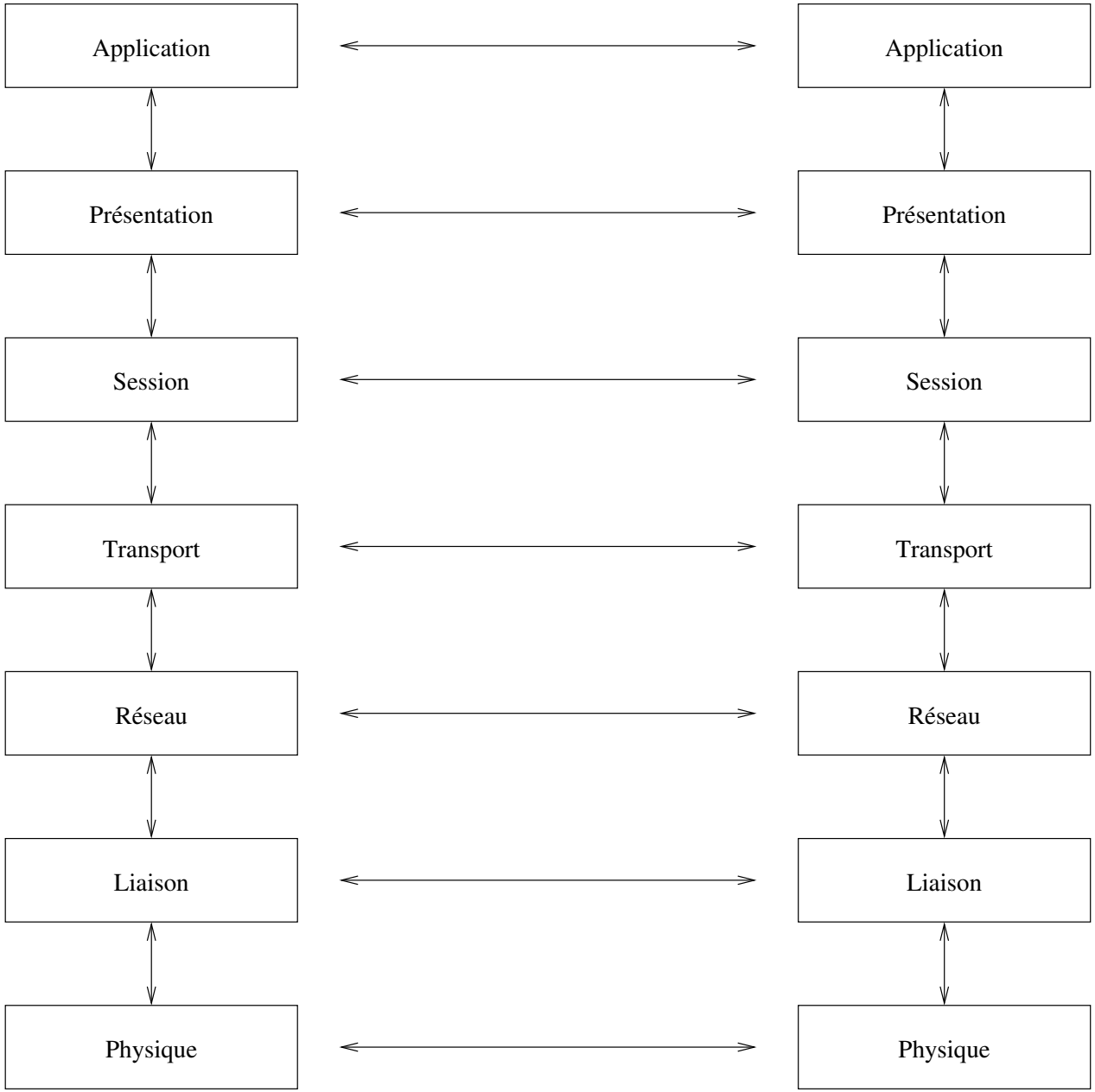
Dans chaque couche, on définit un ensemble de règles (protocole) pour assurer les fonctionnalités.

Les procoles définissent le comportement vis à vis des couches supérieures et inférieures ainsi que des règles internes à la couche.

Un protocole de niveau donné doit alors :

- prendre des données du niveau inférieur,
- traiter ces données en respectant ses règles, en dialoguant éventuellement avec un autre protocole de même niveau,
- rendre les données traitées au niveau supérieur.

Communications



La couche **Physique**

Elle permet de connecter des matériels.

- transmission de données (**bits**) sur le support physique
- le codage(Manchester, bande de base), la durée d'un bit, la vitesse d'échange
- le mode de transmission (**(a)synchrone**)
- transmission dans une ou deux directions (**full** ou **half duplex**)
- l'initialisation et la terminaison
- le brochage des interfaces
- le voltage ...

La couche **Liaison**

Elle permet de faire transiter des informations entre deux matériels connectés. Une sous couche **MAC** (Medium Acces Control) d'accès au support et sous chouche **LLC** (Logical Link Control) de services à la couche Réseau.

- transformation en ligne de données (**trame**, **cellule**)
- transmission en séquence avec acquittement
- séparer, assembler les trames
- retransmission en cas de perte
- régularisation du trafic
- gestion des erreurs
- services à la couche réseau

La couche **Réseau**

Permet l'interconnexion de réseaux hétérogènes. Trouve un chemin dans le réseau entre deux machines.

- acheminement des données (**paquets**) entre l'émetteur et le récepteur
- adresses de réseau
- contrôle des paquets envoyés
- détermination d'une liaison pour envoyer les trames
- adaptation des paquets pour une liaison donnée

La couche **Transport**

Assure les services de bout en bout (machine à machine).

- association adresse de transport adresse de réseau
- établissement et libération des connexions
- contrôle du séquencement des données
- contrôle d'erreurs, reprise sur erreur
- segmentation, groupage des données pour la couche réseau

La couche **Session**

Permet d'établir une liaison durable entre deux machines.

- établissement, libération de connexions pour une session
- échange de données
- gestion des interactions
- synchronisation de la connexion

La couche **Présentation**

Interprète la syntaxe des données qui transitent pendant une session.

- demande d'établissement et de terminaison de sessions
- transfert de données
- négociation de la syntaxe des données
- transformation de la syntaxe des données

La couche **Application**

C'est la couche de plus haut niveau qui interfère avec l'utilisateur.

- identification des partenaires de réseau
- détermination de leur disponibilité
- autorisation de communiquer, sécurité
- synchronisation des différentes applications

Topologie des réseaux

La disposition des entités et les supports physiques dans les réseaux peuvent prendre des formes diverses et variées suivant les applications.

Les réseaux spécifiques sont ceux utilisés pour connecter différents matériels entre eux (terminal-machine centrale), ils connectent des élément homogènes.

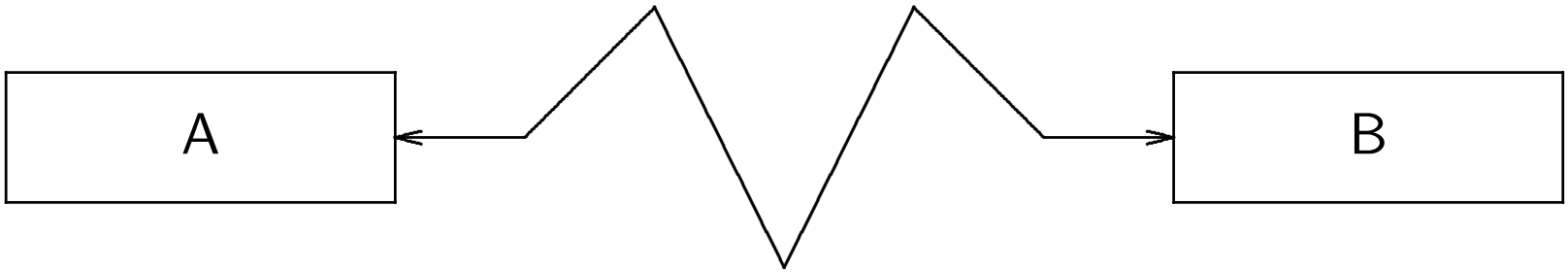
En plus des réseaux spécifiques on distingue souvent trois grands types de réseaux hétérogènes : les **réseaux locaux** (**LAN** Local Area Network), les **réseaux distants** (**WAN** Wide Area Network) et plus récement les **MAN** Middle area Network.

Notion d'adressage

Lorsque plus de deux machines sont connectées à un support il est nécéssaire d'indentifier les partenaires par une **adresse**. Un réseau peut être

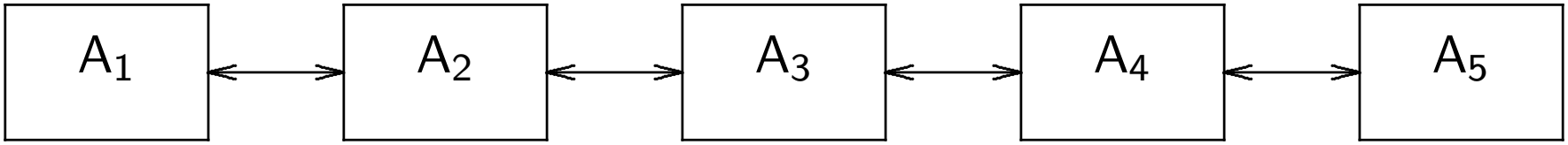
- **Unicast** où les communications se font entre deux points (Téléphone).
- **Broadcast** où une notion de diffusion vers tous les points existe.
- **Multicast** où un point peut envoyer à plusieurs en une seule fois.

Réseau **Point à point**



C'est par exemple le cas d'un terminal connecté directement à une machine, ou celui de deux machines qui communiquent par l'intermédiaire d'un **modem** ou d'une connection entre deux machine de l'internet.

Réseau **Multipoint**

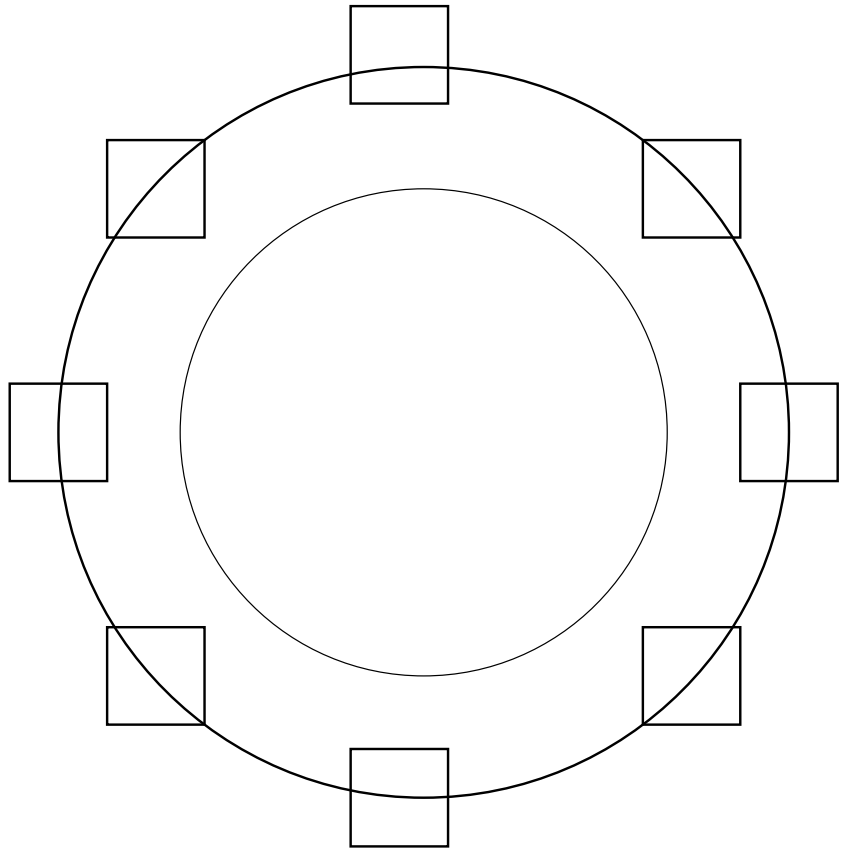


C'est une chaîne de machines qui relaient les données, chaque machine est connectée à au plus deux autres.

Pour aller de A_1 à A_5 les données doivent transiter et être traitées par tous les maillons de la chaîne.

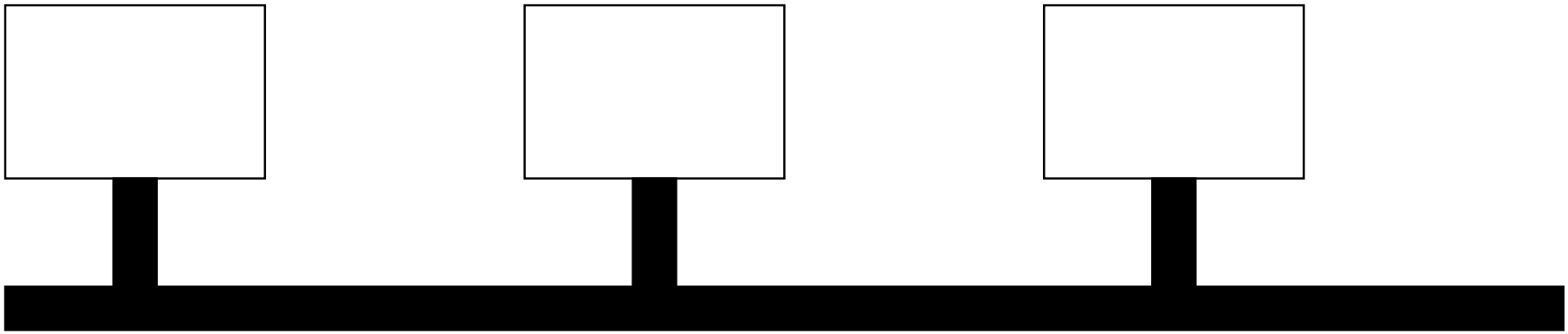
Une machine intermédiaire doit-elle stocker toute la donnée ?

Réseau **Anneau**



La dernière machine est connectée à la première. Une machine mère envoie un signal d'un côté et l'absorbe de l'autre. Modèle de jeton logique (Token Ring) ou de signal physique (FDDI).

Réseau **Bus**

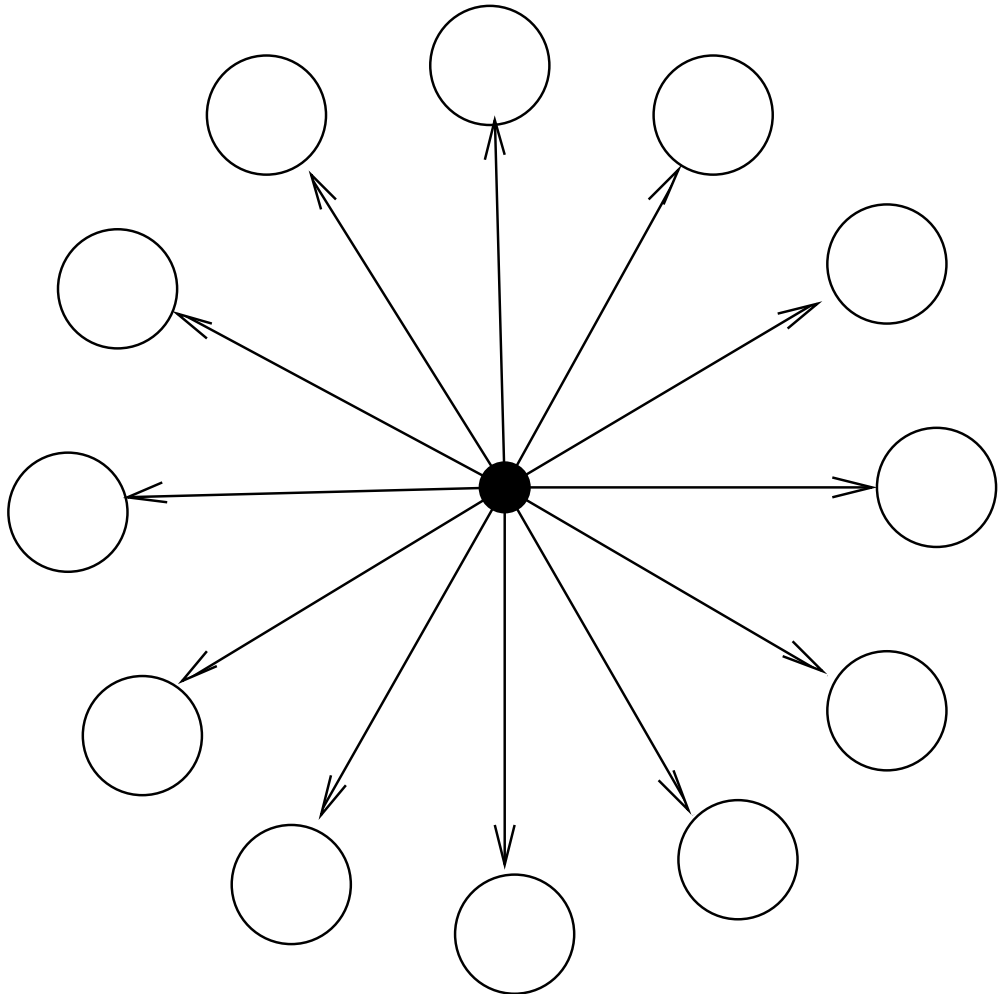


C'est une ligne virtuelle sur laquelle viennent se brancher un certain nombre (variable) de machines.

Chaque point du réseau dispose du bus de manière exclusive. Efrficace mais source de conflit, blocage ...

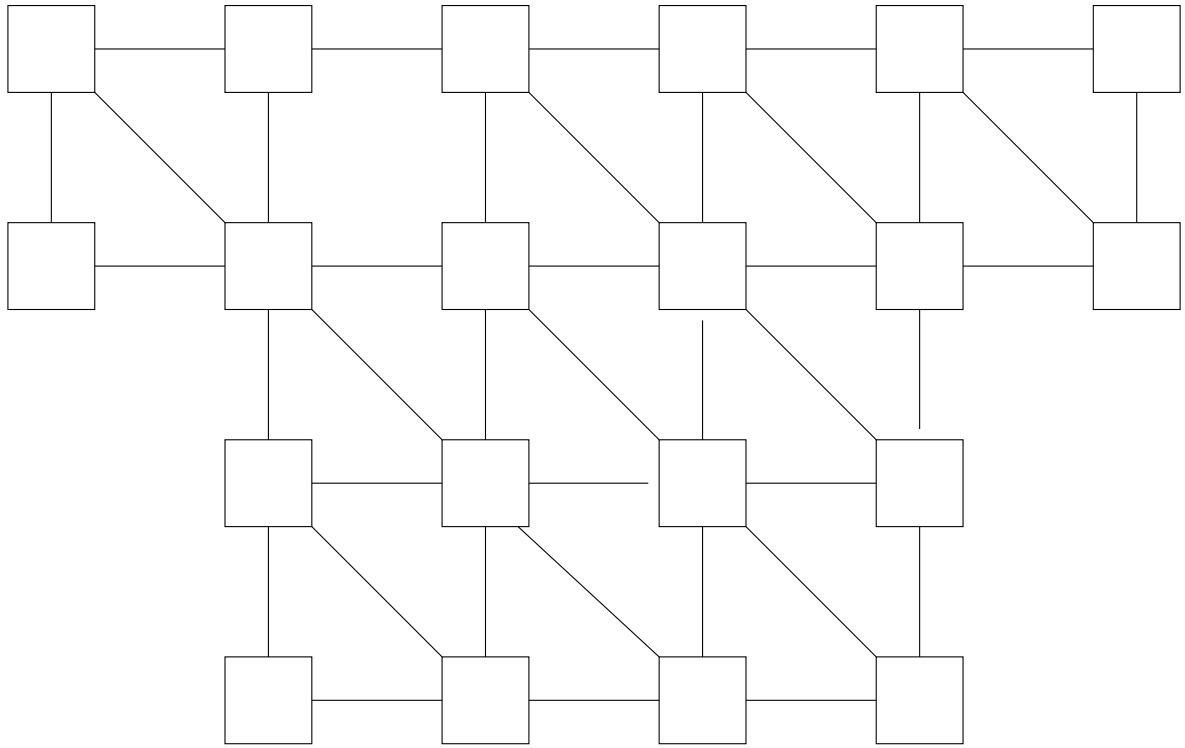
Par exemple les entrées sorties (disque, bande ...) d'une machine se font souvent via un **Bus**.

Réseau **Étoile**



Le point central peut être soit une machine soit un simple boîtier de connexions (Hub, Répeteur).

Réseau **Maillé**



La topologie des liaisons peut être variable. On distingue la **commutation de circuit** dans laquelle une communication passe toujours par les mêmes noeuds (téléphone), la **commutation de paquets** (IP) dans laquelle chaque noeud décide où il doit envoyer ce qu'il reçoit et plus récemment la **commutation de cellules** (ATM) analogue à la commutation de paquets mais sur des données plus petites.

Terminologie (physique)

Le **DTE** (Data Terminal Equipment) désigne la partie d'un connecteur qui se branche sur l'équipement à relier au réseau (souvent male).

Le **DCE** (Data Communication Equipment) désigne la partie qui se branche sur le réseau (souvent femmele).

La **bande passante** d'une connexion physique est l'ensemble des donnés qui peuvent transiter simultanément sur le support.

Une liaison **simplex** est une liaison unilatérale. Ainsi un satellite envoie aux récepteurs.

Une liaison **half duplex** est une liaison où les signaux transitent

Liaisons Asymétriques

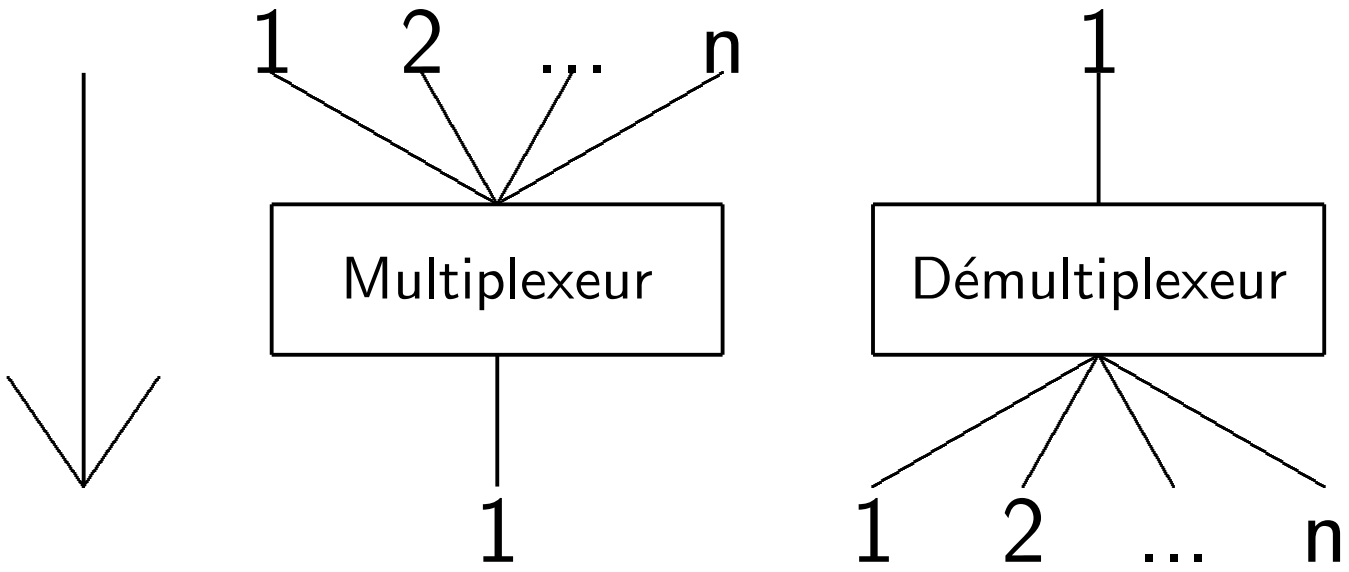
La vitesse d'une ligne se mesure en (K-M-G)**bps**, c'est le nombre de bits par seconde qui transitent sur le support. Il y a lieu de distinguer cette notion de celle de **baud** qui désigne le nombre de changement de phases par secondes d'un signal.

On peut ainsi décider de consacrer plus de modulations pour l'émission que pour la réception. Le **minitel** module à 300 bauds mais envoie à 75 bps et reçoit à 1200 bps. En **ADSL** (Asymetric Digital Subsrciber Line) on a un débit **ascendant** et un débit **descendant**.

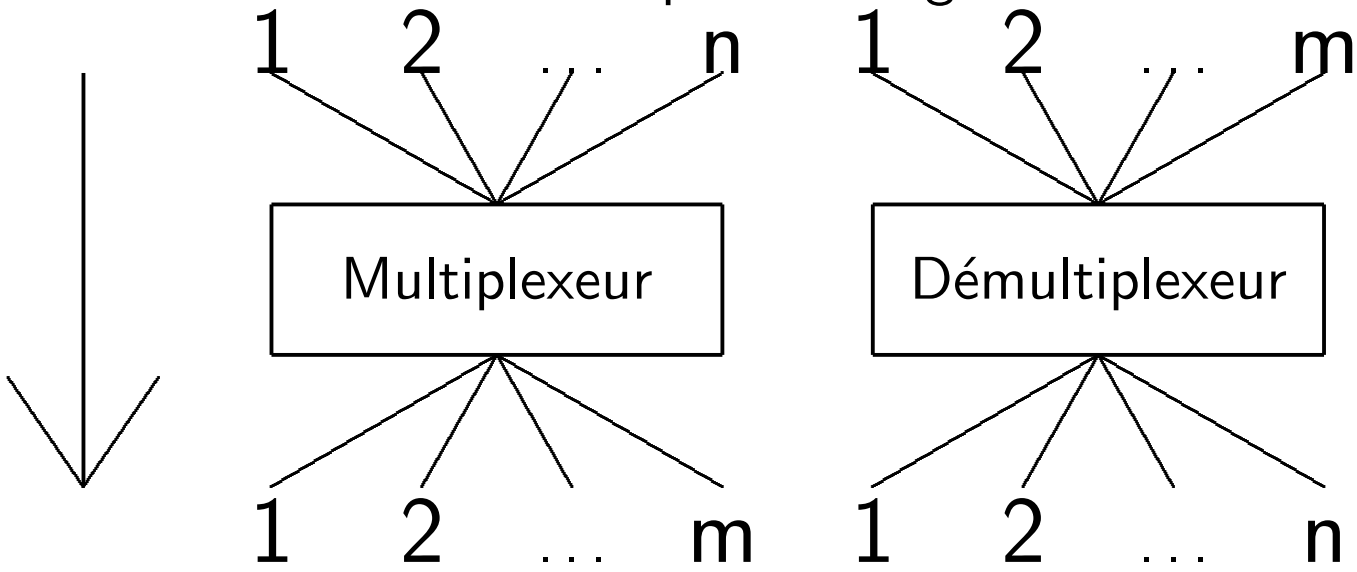
Une liaison est **synchrone** si une opération d'échange de données fait à la fois une lecture et une écriture. Il y a deux signaux.

Une liaison est **asynchrone** si une opération se fait soit dans un sens soit dans l'autre (lecture ou écriture). Il y a un seul signal.

Multiplexeur



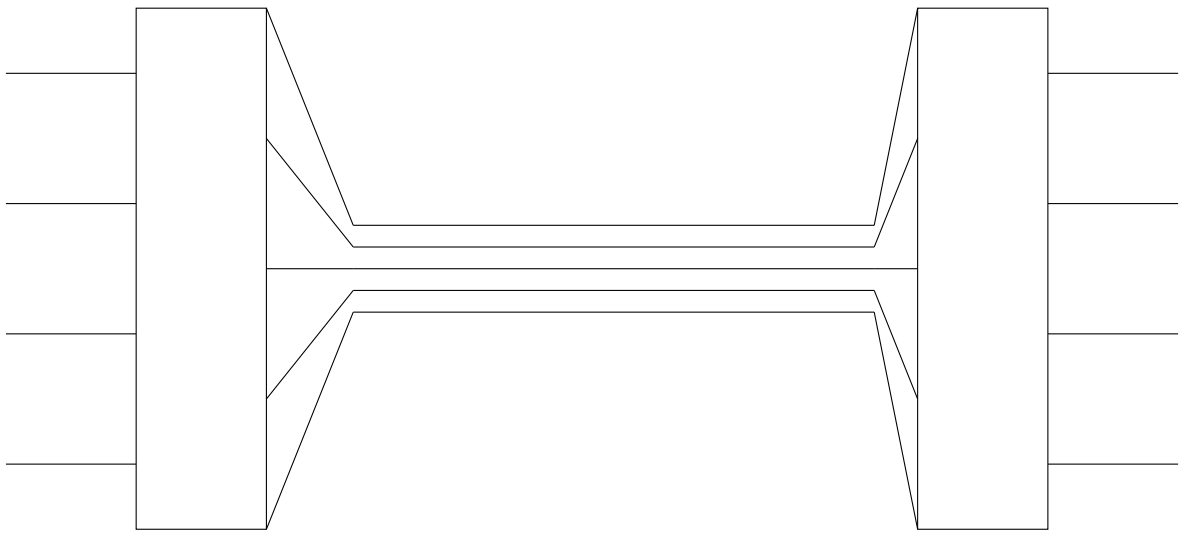
Concenter plusieurs lignes.



choisir une sortie et la restituer !

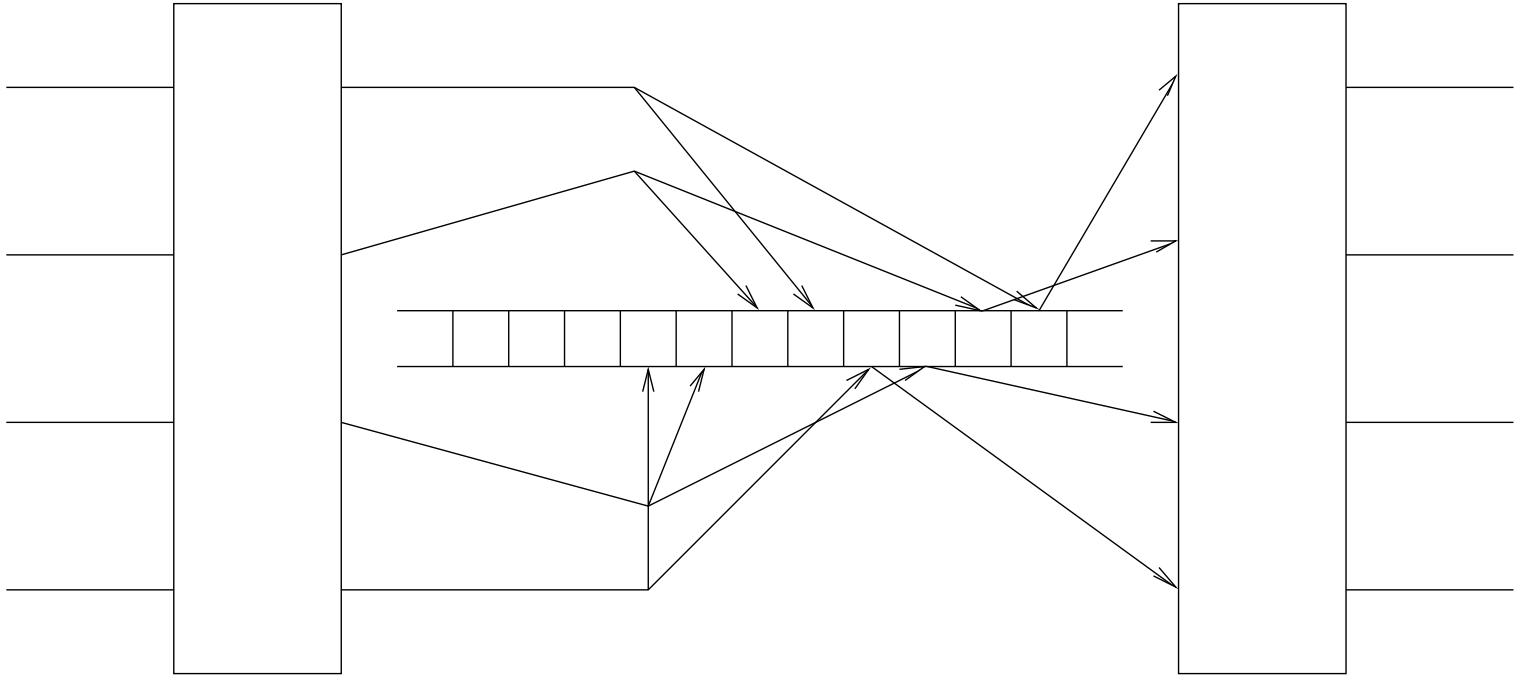
Un réseau multiplexeur

C'est par exemple le cas lorqu'on veut connecter une grappe de terminaux à un site central. On dispose d'une seule ligne qui a une large bande permettant de faire passer plusieurs signaux comme la fibre optique.



On divise la bande passante en autant de **tranches** que l'on veut faire passer de lignes. Chacun dispose d'une bande passante dédiée.

La division de temps



Le principe est le même, mais on accorde à chaque entité un intervalle de temps donné.

La répartition peut ne pas être homogène voire dynamique...

Les erreurs

Les supports physiques se prêtent souvent mal aux communications fiables que l'informatique demande.

Des procédures de détection ou de corrections d'erreurs sont mises en œuvre pour pallier ces déficiences.

Le **contrôle de parité** permet de détecter une erreur dans une suite de bits :

Le **checksum** est analogue au contrôle de parité mais porte sur des mots.

Les **codes cycliques** sont utilsés pour détecter efficacement des erreurs.

Les codes de **Reed et Solomon** permettent de corriger des erreurs.

Terminologie (liaison)

Une liaison entre deux (ou plus) machines est soit **orientée caractère** soit **orientée bit**.

Dans une liaison de type caractère il faut être d'accord sur l'encodage d' :

- un jeu de caractères (ASCII, EBCDIC ...)
- un jeu de codes de contrôle qui sont utilisés comme commandes de la ligne.

Les liaisons de type bit laissent la détermination de l'alphabet aux couches supérieures (5, la couche présentation). Elles transforment les données en **trames** ou **cellules** qui ont souvent un délimiteur de début et de fin. On doit alors de reconnaître ces délimiteurs, de s'assurer qu'elles ont bien été envoyées.

Il y a différents types de liaison :

- Maître esclave(s)(**master-slave**) : une machine (toujours la

Terminologie

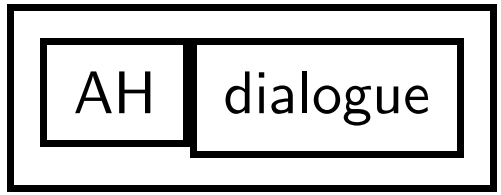
Lorsqu'elles traversent les couches de protocoles les donnés subissent des transformations en vue de franchir sans encombre les obstacles.

Le mécanisme **d'encapsulation-désencapsulation** permet de maintenir l'intégrité des données tout en assurant les services spécifiques d'une couche.

Une donnée | donnée | vient par exemple de la couche réseau (c'est un **paquet**) et doit être transformée pour la couche liaison (c'est alors une **trame** ou **cellule**). La couche réseau choisit la liaison par laquelle le paquet doit transiter, il faut ensuite en faire une trame.

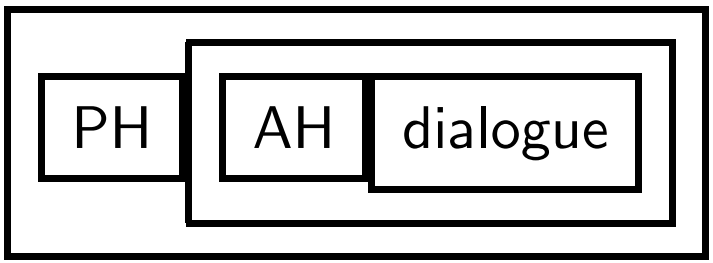
On ajoute alors un en-tête de liaison | el |, et éventuellement un délimiteur de fin de trame | tl |. La trame qui circule sur la liaison est

Fonctionnement schématique

Une application doit dialoguer avec une autre application distante, elle s'adresse à la couche présentation (locale) qu'elle veut utiliser en lui donnant des informations contenues dans un en-tête de présentation AH. le dialogue entre les deux applications aura alors la forme 

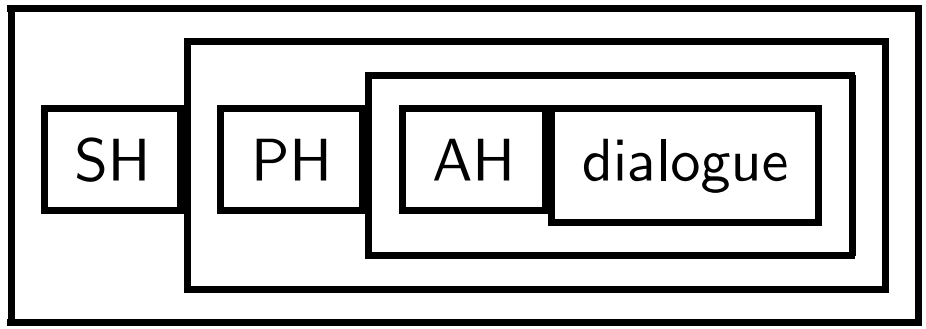
.

Au niveau présentation cet en-tête est examiné, la syntaxe en est interprétée, le niveau présentation décide d'établir une session entre les deux machines. Elle demande alors à la couche session qu'elle veut utiliser et ajoute un en-tête de présentation PH pour dialoguer avec la couche présentation distante. Le dialogue de présentation a alors la forme :



.

De même pour contrôler la session entre les deux machines la couche session rajoute son en-tête SH et ouvre une ou plusieurs connexions entre les deux machines via la couche transport.

Les données 

sont vues comme un flot flot de données par la couche transport. Ce flot est fragmenté en vue de transmission par le réseau, chaque fragment est accompagné d'un en-tête de transport :

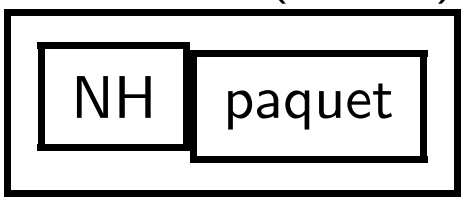
TH_1 flot, TH_2 de, TH_3 donn, TH_4 ées.

L'entité de transport receveuse devra refaire les morceaux et s'assurer qu'il ont bien été transmis.

Le réseau achemine ensuite chacun de ces fragments vers sa destination finale, éventuellement via des passerelles de réseau. Il peut décider de couper ou de grouper plusieurs fragments qui peuvent provenir de différentes entités de transport. Il doit donc acheminer un paquet 

vers sa (ou ses) destination(s) finale(s) via une ou plsieurs liaisons.

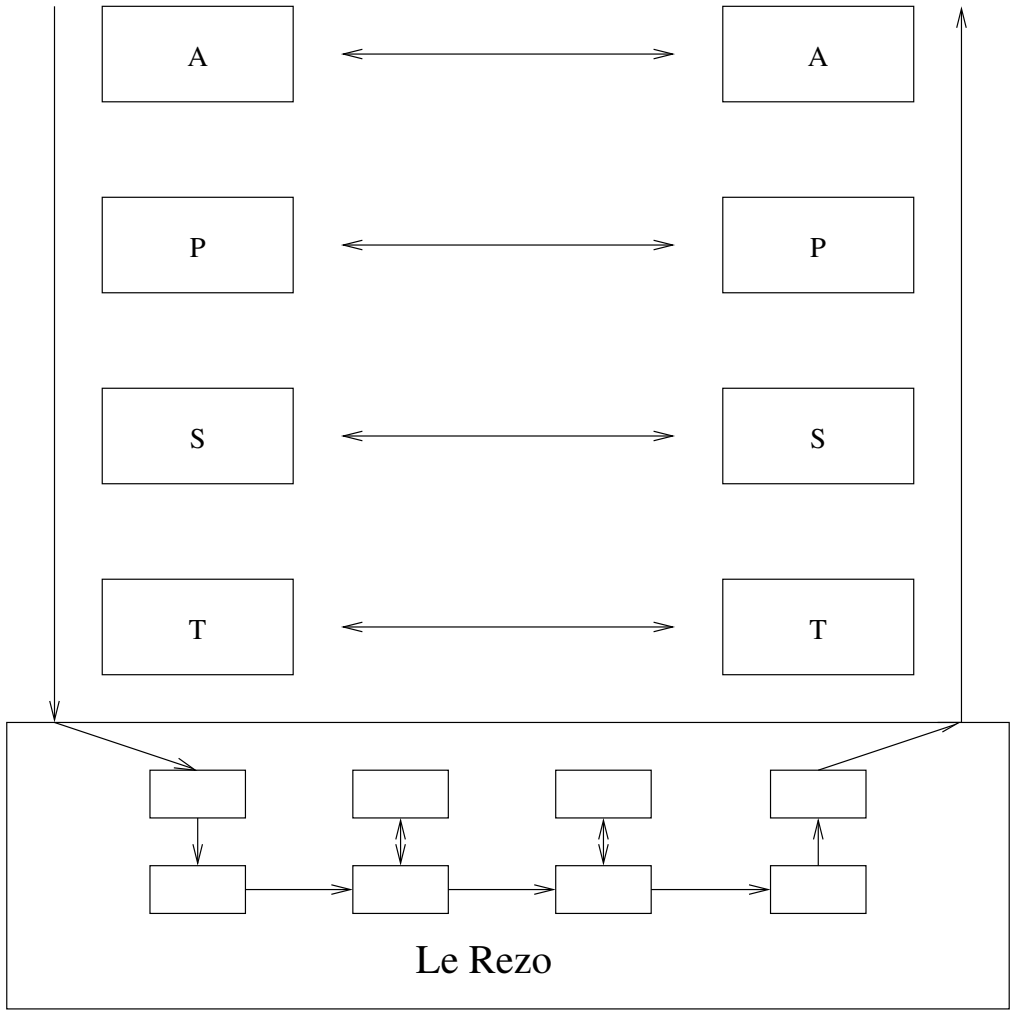
Il ajoute donc un en-tête de réseau 

qui va permettre de trouver la (ou les) destination(s). Le paquet a alors la forme 

et est acheminé comme une trame.

Schéma logique

La communication entre deux applications se fait donc de la façon suivante :



Les Modems

(**MOD**uleurs-**DEM**oduleurs)

Un **modem** permet de connecter des équipements numériques via des lignes analogiques ; c'est un **convertisseur analogique-numérique**.

Deux équipements reliés par modems sont équivalents à ces deux équipements reliés par un canal numérique.

Les vitesses de modulation vont de 75 bits/s à 64 Kbits/s suivant les équipements (**V22** = 1200-1200 **minitel** = 75-1200 (V22bis)).

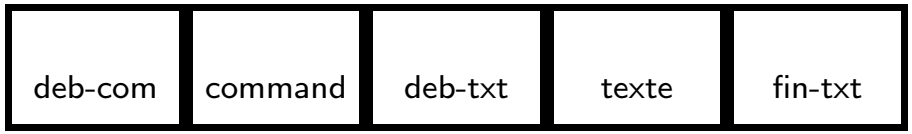
Les modems fonctionnent en envoyant un **son permanent** qu'ils modulent (démodulent) autour d'une **porteuse** en fonction des données numériques (analogiques) qu'ils émettent (reçoivent).
Les boitiersxs ADSL (**Box**) ne sont pas des modems !

BSC-BISYNC

Binary **S**ynchronous **C**ommunication

Est le plus ancien des protocoles de niveau physique (1/2 liaison) il est encore employé dans les architectures IBM sans SNA (System Network Architecture).

C'est un protocole synchrone half duplex d'égal à égal qui fonctionne en mode **caractères** qui véhicule deux types de données : des **commandes** et des **données**.



Sur une ligne point à point des messages vides circulent **en permanence** sur la ligne. La liaison s'établit via une commande spéciale qui demande une réponse.

Les protocoles orientés bit

Transportent des données binaires arbitraires. Pour assurer le contrôle, on transfère les données **par paquets** (trames).

Pour se prévenir contre les erreurs il faut délimiter les trames à l'aide d'une **marque de début** et **d'une marque de fin** (fanion), le problème de transparence se pose.

Dans le cas des protocoles orientés caractères on utilise un **caractère spécial** pour "quoter" les caractères de commande. Le gestionnaire de ligne doit alors automatiquement enlever ces caractères "parasites" (BSC : DLE).
Dans les protocoles orientés bits le fanion est une forme binaire précise, pour SDLC ou HDLC c'est l'octet : 01111110 qui sert de fanion.

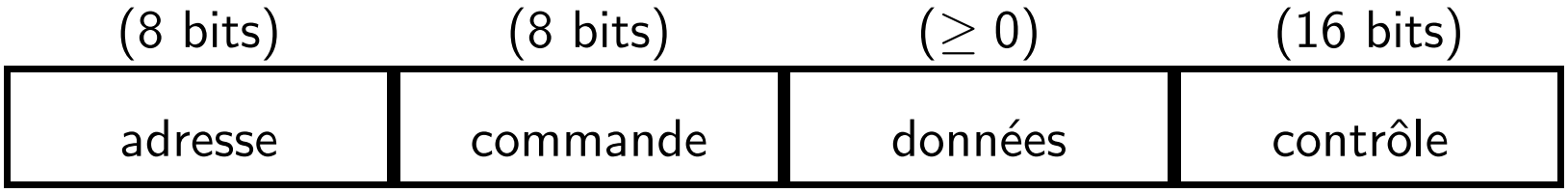
La solution adoptée consiste alors à rajouter un 0 supplémentaire dans les données après toute suite de 5 fois 1.

SDLC-HDLC

Synchronous **D**ata **L**ink **C**ontrol (IBM),

High-level **D**ata **L**ink **C**ontrol (ISO).

Ils sont analogues mais incompatibles :



On a trois types de trames :

Information, avec des numéros d'acquittements (émission réception),

Supervision, avec des commandes pour accuser réception, rejeter . . .. . .

Non numérotées, avec des commandes pour initialiser, terminer . . .

Le contrôle de flux

Est utilisé à différents niveaux :

liaison : pas de bits erronés,

réseau : pas de paquets au mauvais endroit,

transport : pas d'erreurs sur les données.

Le contrôle d'erreur est souvent fait grâce à un Cyclic Redundancy Check (*CRC*) de taille fixe.

On transmet les données en ajoutant au bout le reste modulo un polynôme fixe. À réception le paquet (données + contrôle) doit être divisible par le polynôme.

le Checksum

Se calcule de manière analogue au bit de parité mais en groupant les bits par paquets de taille égaux à la taille du checksum. La somme se fait alors bit à bit.

Ainsi si on veut faire un checksum sur un octet mis au bout d'un groupe de $n+1$ octets on sommera les bits d'indice $8k+i$ pour k variant de 0 à n pour obtenir le $i$ème bit du checksum.

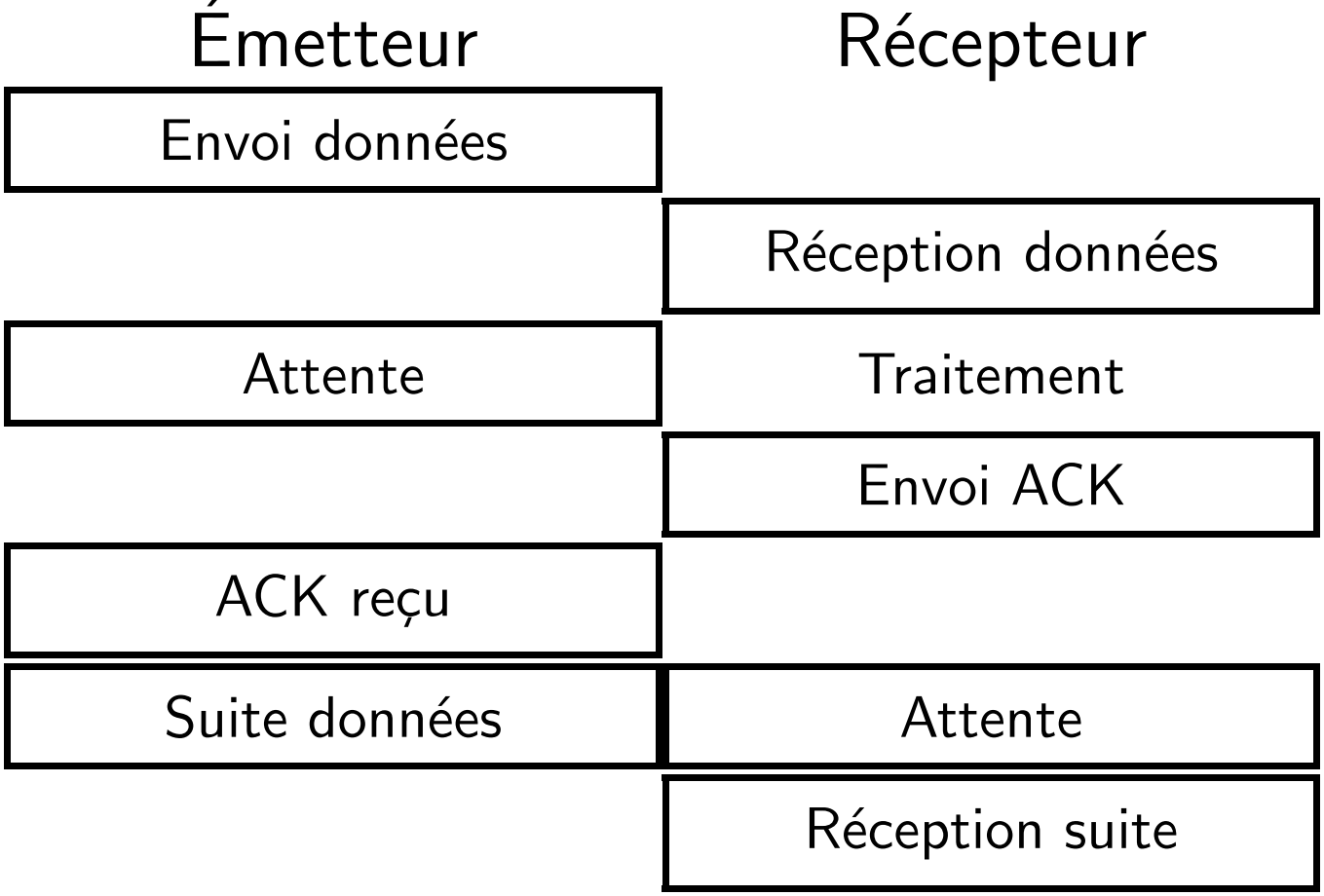
Mécanismes d'anticipation

Dans un mécanisme de contrôle d'erreurs il faut être sûr que les données sont bien arrivées au niveau où elles ont été transmises ; on utilise des "accusés de réception".

Si on attend un accusé de réception à chaque fois qu'on fait une transmission il peut en résulter une attente non négligeable $\Rightarrow$ mauvaise utilisation de la ligne.

Pour une ligne synchrone (BSC) où les échanges se font dans les deux sens en même temps l'envoyeur est obligé d'attendre l'acquittement du receveur avant d'envoyer d'autres données. La transaction suivante est donc nécessairement vide car le receveur doit traiter la donnée envoyée $\Rightarrow$ 50 % de perte de la ligne.

Sans anticipation



On **anticipe** la réponse du récepteur, on n'envoie plus trame par trame mais par groupe, c'est la **fenêtre d'anticipation**.

Elle peut avoir une longueur quelconque, être négociée en cours de

Anticipation

Dans un protocole synchrone on utilise souvent deux caractères d'acquittement : ACK0 (pair) ACK1 (impair).

émetteur	Récepteur	
$données_1$	$\emptyset$	
$données_2$	$\emptyset$	$traitement_1$
$\emptyset$	ACK0	$traitement_2$
$données_3$	ACK1	
$données_4$	ACK0	$traitement_3$
$données_5$	ACK1	$traitement_4$

Un seul caractère NACK suffit et provoque la retransmission de 2 trames, il faut que l'émetteur dispose d'une zone tampon.

X25

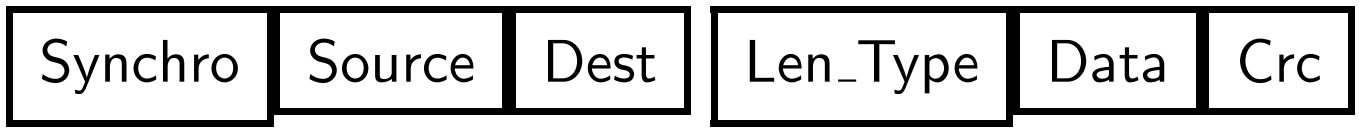
Définit les règles que doivent respecter des machines (DTE) qui se connectent à un DCE du réseau X25 mais pas comment fonctionne le réseau lui même. Il donne les normes aux niveaux :

1. deux interfaces physiques : numérique synchrone (X21) ou asynchrone (V24) via un **PAD** (**P**acket **A**ssembler **D**isassembler).
2. le niveau liaison est géré par un protocole proche de HDLC.
3. le niveau réseau fournit différents types de services et de connexions.

La liaison de deux DTE se fait via l'établissement d'un circuit virtuel temporaire ou permanent qui autorise 4096 voies logiques et des services d'acquittement de messages qui sont délivrés dans l'ordre.

Ethernet, CSMA CD

Une **trame** Ethernet



- Synchro est une suite de bits 101010...11 sur 8 octets,
- Adresses sources et destination sur 6 octets chacune,
- Len_Type est soit la longueur soit le type (la couche de niveau 3).

Toute machine peut accéder au réseau **Mulitple Acces** s'il est libre (**Carrier Sense**), envoie ses données et continue d'écouter (12 octets) pour vérifier que personne n'a émis en même temps **Collision Detection**.

Problématique d'interconnexion de réseaux

La multiplicité des normes de liaison et le besoin d'équipements hétérogènes font qu'il est nécessaire d'avoir des matériels pour connecter entre eux différents réseaux ou étendre ces réseaux.

Raccorder deux réseaux locaux de type Ethernet ? Ce sont des réseaux de type BUS où toute station a droit d'émettre dès que le réseau est libre, elles ne détectent la collision qu'après émission.
Un **répéteur** amplifie le signal et allonge la longueur du réseau.
Un **pont** prend les trames d'un côté et les transmet de l'autre. Des ponts "intelligents" existent même pour éviter de transmettre des trames "inutiles".

Pour raccorder des réseaux différents les problèmes sont multiples :

- les services qu'offre chacun sont différents,
- les formats des données sont différents,
- les vitesses des réseaux sont différentes.

Des normes IEEE existent cependant pour relier entre eux les différents réseaux 802.x (réseaux locaux : Ethernet, Wifi ...).

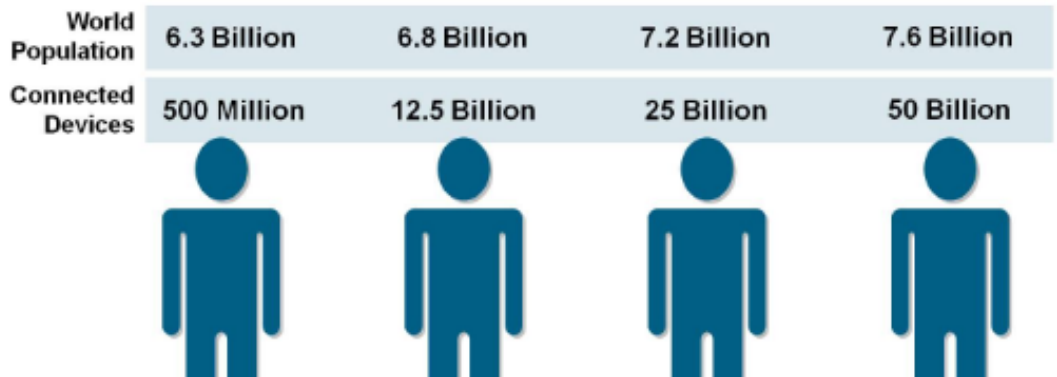
Les réseaux de type Internet

Internet n'est pas à proprement parler un réseau, mais une **famille de réseaux** connectés entre eux.

Ces liaisons permettent de relier, non pas simplement deux machines entre elles, mais deux *réseaux* entre eux.

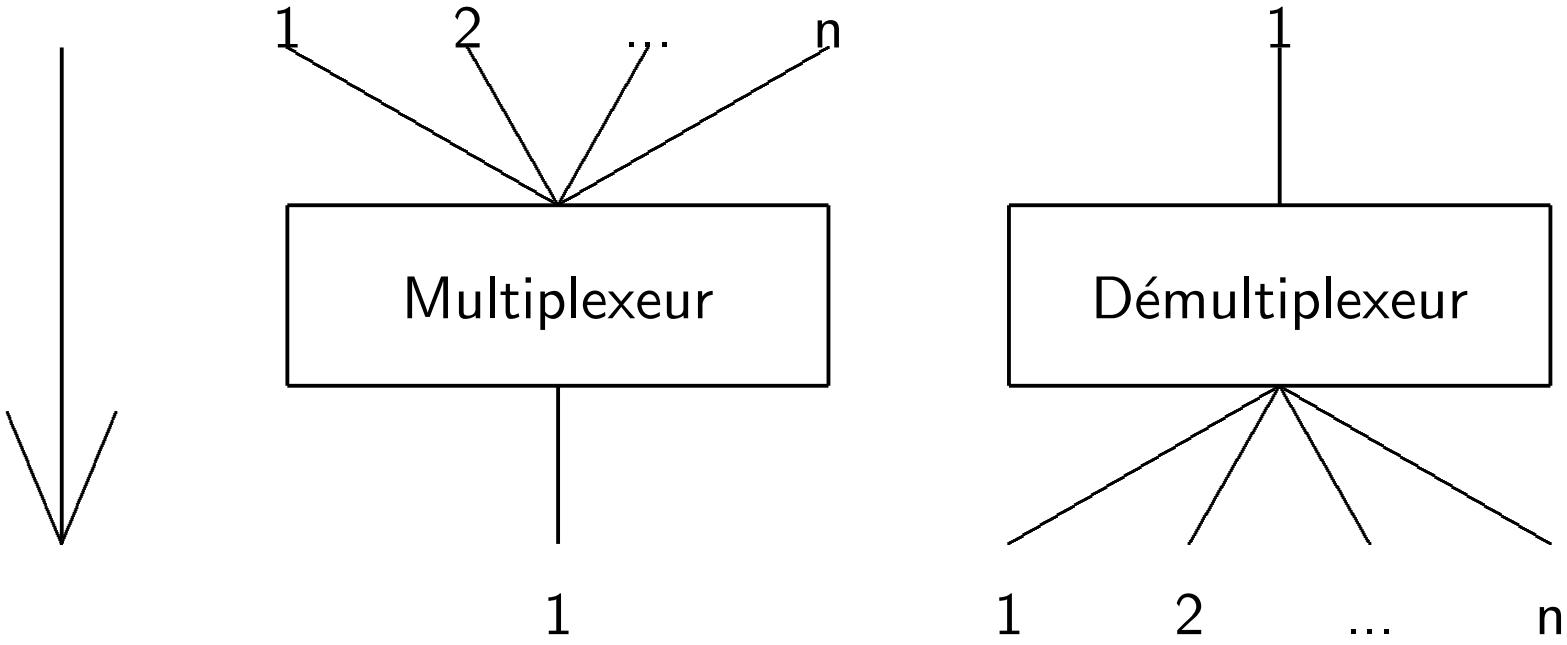
Internet est donc une **Interconnexion de réseaux** (INTERNETwork).

L'utilisateur accède à une machine distante de la même façon qu'à une machine sur son réseau local.

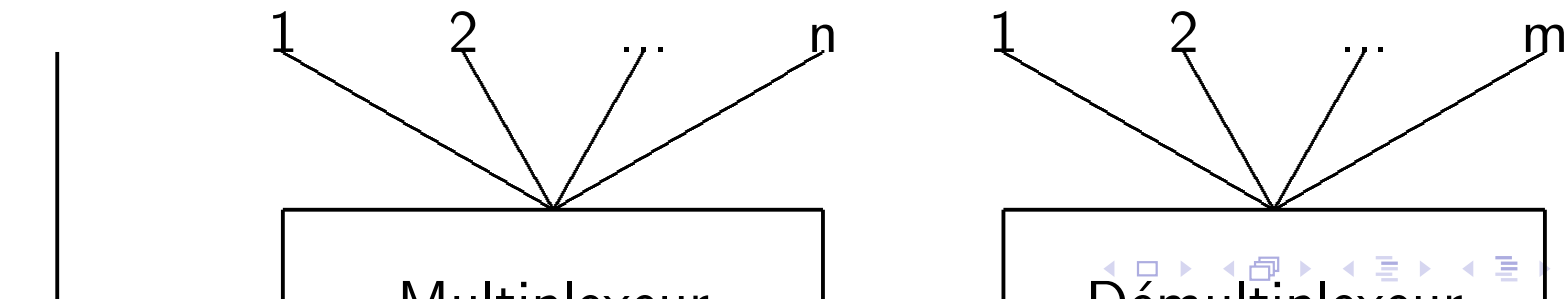


Multiplexage

Multiplexeur 1 n (**TCP**, **UDP**)



Multiplexeur m n (**IP**)



Pourquoi un niveau routage IP ?

Le niveau **IP** permet de rendre les applications réseaux *indépendantes* de la couche physique. On crée ainsi un **réseau logique**.

Les adresses physiques sont liées au matériel (Ethernet), on peut ainsi changer une carte physique, sans affecter le réseau logique. On peut également changer une adresse IP sans changer de numéro physique.

Pour un réseau local la couche IP ajoute des opérations, mais lorsqu'il y a beaucoup de machines, le routage et la bonne adaptation au support physique deviennent des opérations compliquées.

Des protocoles spécialisés ont été développés pour assurer un routage **dynamique** qui s'adapte à n'importe quelle configuration.

Structure d'une adresse IP

Une adresse **IP** est formée de 4 octets, on la note en général de façon décimale en séparant les octets par des points : <oc1>.<oc2>.<oc3>.<oc4>, <oc$i$> est le nombre en base 10 représenté par l'octet. Par exemple :

132.227.66.4

désigne la suite de 4 octets en binaire :

10000100 11100011 01000010 00000100

Pour simplifier le routage, et l'attribution des **numéros IP**, une adresse est divisée en deux parties :

<réseau> , <machine>

- <réseau> désigne un groupe de machines,
- <machine> désigne une machine dans son réseau.

Un numéro <réseau> est attribué à une institution qui gère les numéros <machine>.

Réseaux

On distingue trois types de réseaux **Internet** :

- les réseaux de **classe A** : la partie <réseau> occupe 1 octet, et la partie <machine> en occupe 3. Leur numéro de réseau est entre 1 et 126 (premier bit à 0),
- les réseaux de **classe B** : la partie <réseau> occupe 2 octets, et la partie <machine> en occupe 2. Leur numéro de réseau est entre 128.1 et 191.254 (premiers bits à 10),
- les réseaux de **classe C** : la partie <réseau> occupe 3 octets, et la partie <machine> en occupe 1. Leur numéro de réseau est entre 192.1.1 et 223.254.254 (premiers bits à 110).

Les numéros 0 et 255 sont spéciaux, ceux au-delà de 223 sont pour d'autres classes d'adresses (adressage multiple . . .).

Adressage privé

L'adresse 127.0.0.1 est virtuelle et désigne la machine locale (loopback).

Les numéros de réseaux

- 10.0.0.0 (classe A),
- 172.16.0.0 à 172.31.0.0 (classe B) et
- 192.168.0.0 à 192.168.254.0 (classe C)

sont réservés pour des usages internes à une société et permettent de palier au manque d'adresses IP.

Lorsque tous les bits de la partie machine d'une adresse sont à 1 on a une adresse de diffusion (broadcast) qui désigne l'ensemble des machines du réseau.

Sous réseaux (RFC 950)

Une institution dispose d'une *plage d'adresses* qu'elle doit répartir. Elle peut diviser le champ <machine> dont elle dispose en deux parties :

<sous-réseau> , <sous-machine>,

une adresse **IP** a alors la structure :

<réseau> , <sous-réseau> , <sous-machine>.

Par exemple le réseau de classe B 132.227 dispose d'un champ <machine> de 16 bits, il décide de diviser ce champ en une partie <sous-réseau> de 8 bits et une partie <sous-machine> de 8 autres bits.

Maintenant 132.227.66 est un sous-réseau du réseau 132.227, tout se passe comme si 132.227.66 était une adresse de classe C. Le réseau 132.227 a un **masque de réseau** (netmask). La machine 132.227.66.4 a les champs :

<132.227> , <66> , <4>.

Les adresses physiques (Ethernet)

Une adresse Ethernet est constituée de 48 bits (6 octets) que l'on note en hexadécimal :

08 :00 :20 :03 :2F :40,

ces numéros, uniques pour chaque carte sont attribués par le constructeur.

Ils ne sont pas configurables.

L'association entre un numéro IP et une carte Ethernet est faite par l'intermédiaire d'une table dynamique.

Un protocole (**ARP**) de niveau physique (Ethernet) a été mis en place pour permettre à une machine de connaître le numéro de carte Ethernet à partir d'un numéro Internet. On peut même attribuer un numéro IP dynamiquement à partir d'un numéro de carte Ethernet ; c'est le protocole **RARP** (Reverse Address Resolution Protocol).

ARP - RARP

Lorsque IP décide d'envoyer un paquet sur un réseau Ethernet, il doit connaître l'adresse physique. S'il ne la possède pas dans sa table, il demande à **ARP** de la lui trouver.

La machine **A** a besoin de l'adresse **EthB** d'une machine **B** (**IPB**), **ARP** construit une trame Ethernet spéciale destinée à l'ensemble des machines du réseau local (broadcast). Chaque machine regarde la trame, l'envoie à son module **ARP**, l'une des machines reconnaît son adresse IP dans le paquet. Elle renvoie donc directement une trame réponse à l'adresse **EthA**.

A peut mettre à jour sa table, IP peut maintenant la consulter, et envoyer une trame sur le réseau Ethernet.

IP - ICMP

IP se charge de l'acheminement des paquets. Il fait de son mieux pour qu'un paquet arrive à destination, mais il n'assure pas sa délivrance finale. Une fois un paquet envoyé il est oublié.

Pour qu'un réseau TCP/IP fonctionne correctement le taux d'erreur sur le réseau physique doit être inférieur à 5%, la taille des trames envoyées (**MTU** Maximum Transmission Unit) doit être adaptée au support.

IP peut donc **fragmenter** les datagrammes qu'il envoie, il doit alors les ré-assembler à l'arrivée. C'est la **fragmentation défragmentation**.

ICMP (Internet Control Message Protocol) permet à des machines de s'envoyer des informations sur le routage. Il utilise **IP** comme transport, mais fait partie intégrante du niveau logique IP.

Noms de machines

Les numéros **IP** sont utilisés dans toutes les connexions, mais ils ne sont pas très mnémoniques. Les applications **TCP/IP** autorisent des envois vers des **noms de machines**.

Par exemple la machine 193.54.22.132 s'appelle hermes. Pour les petits réseaux une table statique (**hosts**) d'associations <numéro-IP> , <nom-machine>, est suffisante.

Avec 3 milliards machines connectées c'est illusoire, on utilise alors des **noms complets** du type :

<nom-machine>.<nom-organisme>.<nom-zone>,

et des **serveurs de noms** (nameservers) qui font dynamiquement l'association et la machine 193.54.22.132 s'appelle hermes.ensiie..fr.

TCP - UDP

TCP assure le transfert fiable d'un flot de données entre deux machines. Il s'assure que chaque datagramme qu'il construit arrive sans détérioration grâce à un mécanisme de *fenêtre glissante* (sliding window). **TCP** demande un accusé de réception pour chaque octet qu'il envoie, l'envoi est recommencé après un délai d'expiration (time-out).

UDP (User Datagram Protocol) est un mécanisme de contrôle en mode **non-connecté** (la délivrance des données n'est pas assurée). Il ne fait que le multiplexage sur les ports source et cible, c'est l'application (**NFS** par exemple) qui détermine la taille des datagrammes qui doivent tenir dans un paquet IP.

Applications

Les applications réseau utilisent des ports **TCP** ou **UDP**, elles doivent :

- envoyer et recevoir des données (messages),
- ouvrir ou fermer une connexion,
- accepter ou attendre des messages.

Ces opérations sont réalisées par l'intermédiaire de **sockets** (UNIX). On travaille sur une **socket** un peu comme sur un fichier (read, write, open, close) mais avec des opérations en plus qui permettent de gérer un flot de données (accept, wait).

Par exemple la commande Telnet B depuis la machine A va :

- demander un port **TCP** libre pour émettre (1234),
- créer une socket TCP/1234,A,TCP/23,B

Clients-Serveurs

Quand on exécute la commande Telnet B depuis A, il faut que sur B un programme accepte les connexions sur le port 23.

Ce programme (in.telnetd) peut encore une fois jouer un rôle de multiplexeur, et accepter plusieurs connexions concurrentes. C'est un **serveur** de terminaux.

Lorsqu'on exécute Telnet on est **client** du serveur.

Le mécanisme **client/serveur** est très utilisé dans le monde Internet. Un serveur attend une ou plusieurs connexions sur un port donné (port standard), il suppose que toute connexion arrivant sur ce port connaît son protocole ; les messages sont des **requêtes** d'un client.

Les protocoles au dessus de **TCP** et **UDP**

Les applications qui utilisent **UDP** comme le serveur de noms n'ont en général pas besoin d'une grande fiabilité, ils font eux-mêmes leurs demandes de re-transmissions. Lorsqu'on demande le numéro **IP** d'une machine nommée **A** la question et la réponse sont de "petite taille", il est inutile de démarrer un protocole compliqué et lourd comme **TCP**. Ces protocoles utilisent en général leur propre format de données.

Les applications qui utilisent **TCP** ont en général besoin de transférer beaucoup de données de manière fiable. Les protocoles utilisés sont en général des protocoles "en clair", c'est à dire que les ordres du protocole transitent directement sans passer par un format interprétatif pour les données.

Applications UDP

Les principales sont :

- **Internet Name Server** le système d'association {nom machine} $\leftrightarrow$ {numéro **IP**}
- **Trivial File Transfer Protocol** (TFTP) qui implémente un système simple de transfert de fichiers entre deux machines.
- **NFS** (Network File System) qui permet de voir un système de fichiers d'une autre machine comme un système de fichiers local.
  Il utilise le mécanisme **RPC** (Remote Procedure Call). Le système d'exploitation de la machine client communique avec le programme nfsd sur la machine serveur pour obtenir des informations sur des "fichiers virtuels". Il y a peu de différence entre un vrai **filesystem** et un filesystem NFS.

Le service de nommage DNS

Les noms de machines sont hiérarchisés en zones délimitées par des points (. et un nom à la forme machine.zone.

Une zone est formée d'un nom et d'une liste de zones qui sont ses fils dans un arbre. Le nom vide correspond à la racine de l'arbre. Une zone dont la liste des fils est vide est un **domaine**.

Chaque zone posséde un **serveur de noms**. Pour un domaine le serveur de noms connait des noms de machine et leur adresse IP ainsi que l'adresse IP du serveur de noms de la zone à laquelle il appartient et à qui il transmet les demaine qui ne concernent pas sa zone.

Les serveurs de noms de zone conaisse l'adresse IP d'un serveur pour chacune des zones qui sont ses fils.

Une zone qui n'est pas la racines connait l'adresse IP du serveur de noms de son père.

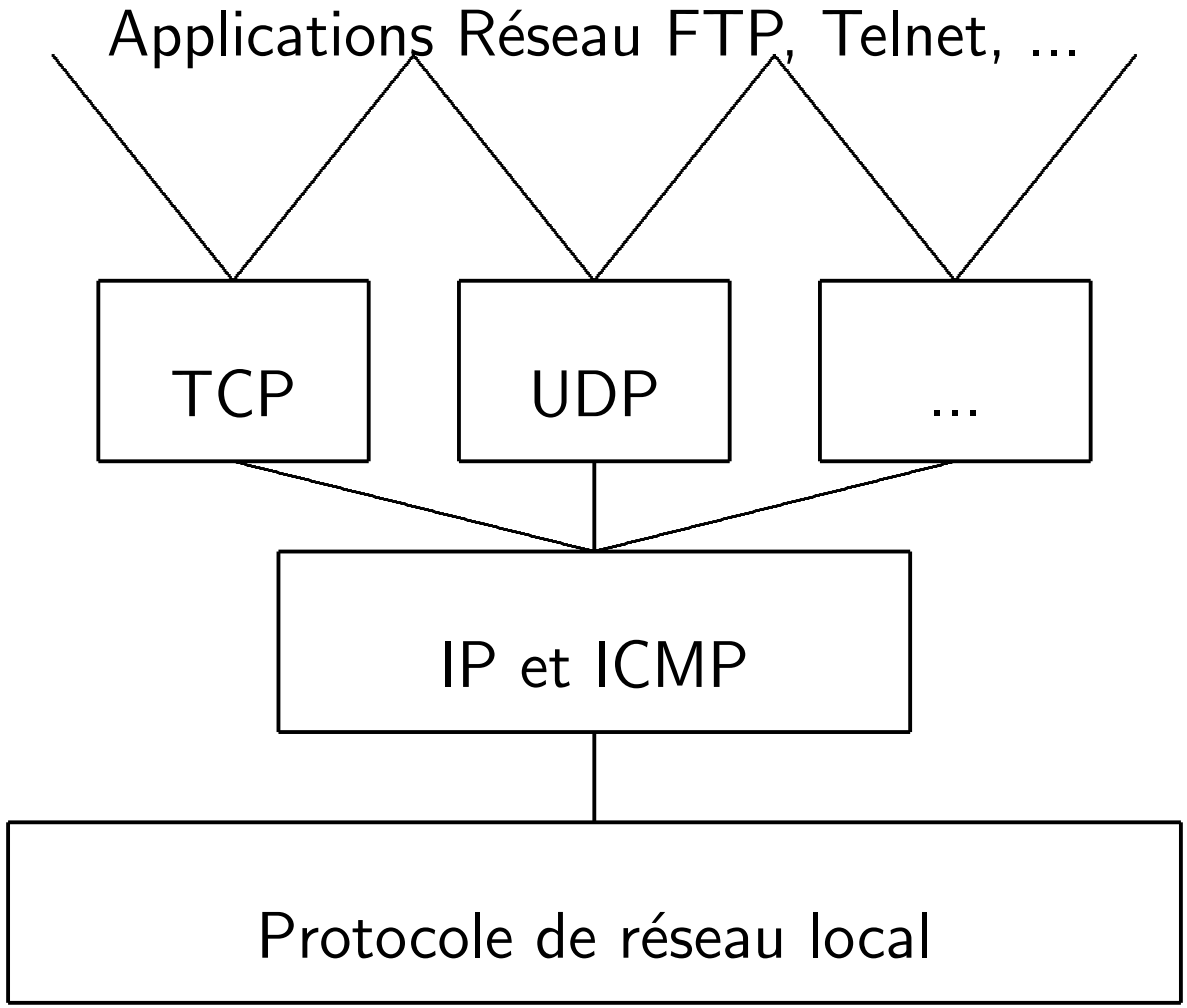
Applications TCP

FTP : utilise deux connexions, une pour le contrôle (port TCP 20 du serveur), une pour les données (port TCP 21 du serveur). Il permet de transférer des fichiers entre des machines.

SMTP(Simple Mail Transfert Protocol) : est un protocole qui permet d'échanger du courrier électronique entre deux machines (port TCP 25 du serveur).

Ssh(Secure shell) : permet d'exécuter une commande sur une autre machine (port TCP 22 du serveur). Cette commande est utilisée par la commande **scp** qui copie un fichier de façon plus transparente que ftp.

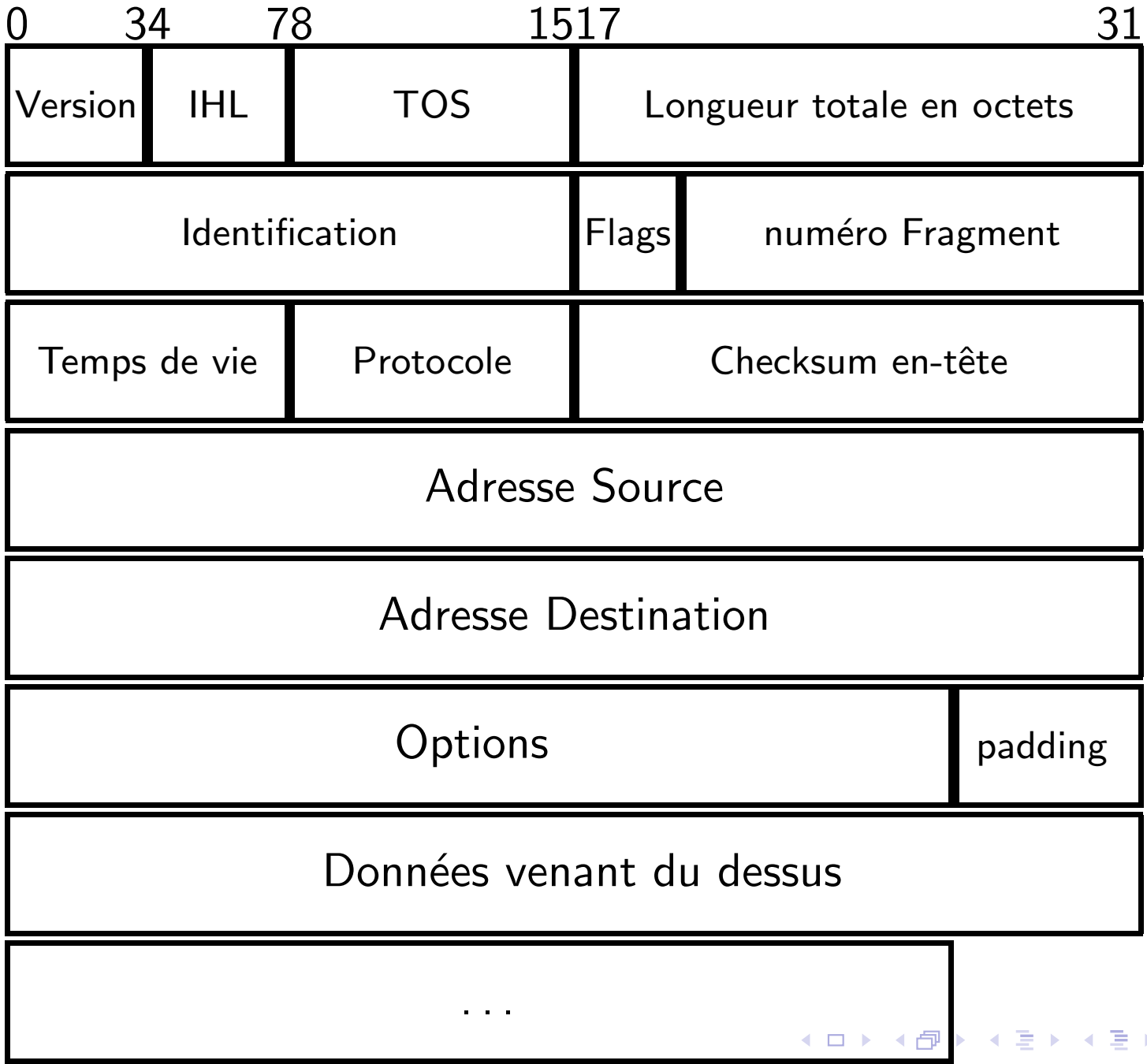
IP et les autres protocoles



IP doit :

- acheminer les paquets provenant des modules de plus haut niveau (**TCP, UDP**) vers leur destination finale,

Les datagrammes **IP**

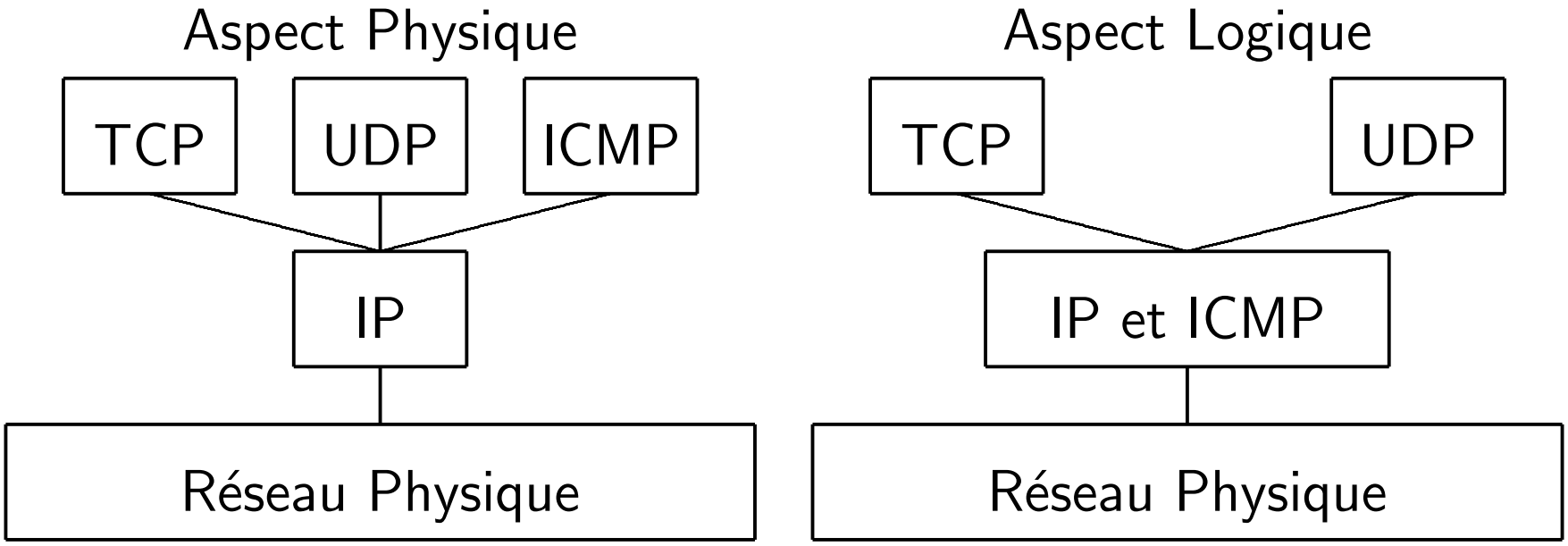


- Version : la version actuelle est **4**.
- IHL : (Internal Header Length), c'est le nombre de mots de 32 bits qu'il y a dans l'en-tête.
- TOS : (Type Of Service) détermine la qualité du service pour le paquet (rapide, normal ...)
- Longueur : c'est la longueur **totale** du paquet en octets.
- Identification : c'est un numéro d'ordre pour le paquet.
- Flags : (3 bits) indiquent si on peut fragmenter ce paquet ...
- Numéro : c'est le numéro de fragment du paquet.
- Temps de vie : est le temps en secondes pendant lequel ce paquet reste actif.
- Protocole : la couche supérieure à qui on doit délivrer le paquet.
- Checksum : pour détecter les erreurs dans **l'en-tête** du paquet, (faux $\rightarrow$ poubelle).
- Adresse source : l'émetteur du paquet.
- Adresse destination : le destinataire.
- Options : permettent de préciser certaines choses dans les paquets (niveau de sécurité, routage spécial pour un paquet ...)

ICMP

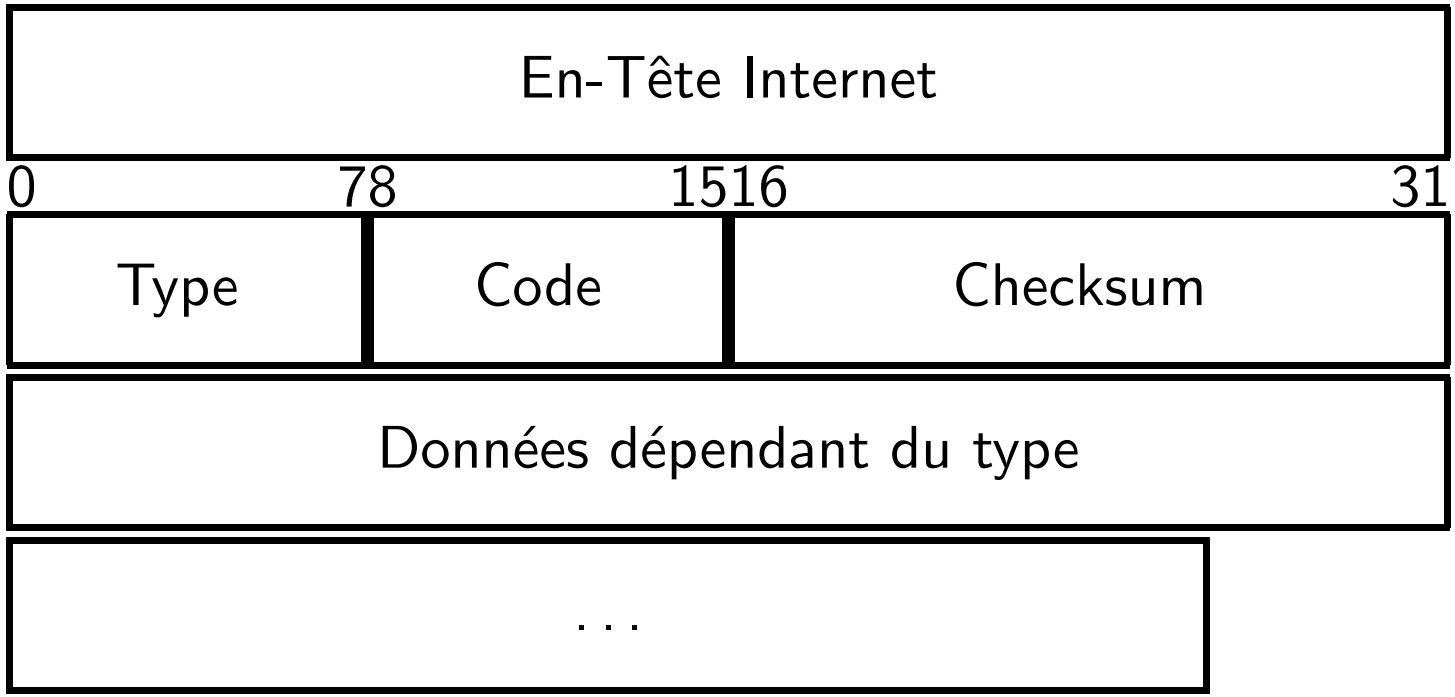
Internet Control Message Protocol sert de couche de contrôle pour les deux fonctions qu'assure **IP**. Il gère les routes et la fragmentation de messages.

Il utilise **IP** comme transport, mais fait partie du niveau logique **IP**.



Un message **ICMP** est accompagné de son en-tête IP.

Un message **ICMP**



- Type : donne la nature du message **ICMP**.
- Code :
- Checksum : sur la totalité du message.
- Données : interprétées en fonction du type.

Le niveau **IP**

IP fait voir plusieurs *réseaux physiques* comme un seul *réseau logique*, ses tâches sont :

- prendre les données des niveaux supérieurs **TCP, UDP**, il joue un rôle de **multiplexeur**,
- choisir une “passerelle” en fonction de la destination finale, c’est le rôle du **routage**,
- trouver une interface physique pour acheminer les données vers la **passerelle**, il joue un rôle de **démultiplexeur**,
- suivant l’interface physique choisie il peut éventuellement **fragmenter** les paquets, en fonction du **MTU** de l’interface physique,
- encapsuler les paquets pour une interface physique donnée.

Le routage **IP**

On distingue deux types de machines, les machines **simples** et les **passerelles** qui ont plusieurs interfaces physiques. Le routage le plus simple est celui d'une machine simple où le choix d'une interface physique ne se pose pas.

Le routage peut être **direct** si la destination finale est sur le même réseau physique, ou **indirect** si la destination finale est sur un autre réseau physique. Il faut alors choisir une passerelle en fonction de la partie réseau de l'adresse finale. Lorsque les masques de réseaux sont en fonction c'est la partie sous-réseau de l'adresse qui sert pour le routage.

La fonction de routage **IP** est assurée par une table d'association du type :

numéro-de-réseau-ou-de-machine *passerelle-sur-le-même-réseau*

Prenons l'exemple du réseau de classe B `132.227.0.0` où les

Machine Simple

la table de routage de la machine `132.227.2.1` contient :

132.227.2.0	via	132.227.2.1
132.227.1.0	via	132.227.2.3
132.227.3.0	via	132.227.2.2
132.227.4.0	via	132.227.2.2

Si le reste du monde **Internet** est relié via une machine sur un des sous réseaux 3 ou 4 on peut remplacer les deux dernières lignes par :

default	via	132.227.2.2

Si le reste du monde **Internet** est relié via une machine sur le sous réseau 1 on peut remplacer la deuxième ligne par :

default	via	132.227.2.2

Si il existe plusieurs interconnexions possibles entre des réseaux ou des sous réseaux, une machine doit choisir une des possibilités.

Les messages **ICMP Redirect** permettent aux passerelles de rediriger dynamiquement les paquets. Ils sont de deux types **Host redirect** ou **Net redirect**, les premiers indiquent à une machine que les paquets à destination d'une machine donnée sont à envoyer à une autre passerelle.

Par exemple si la machine 132.227.2.1 n'avait à la base qu'une route par défaut vers 132.227.2.3 si elle envoie un paquet à destination de la machine 132.227.4.1, le paquet sera éliminé par 132.227.2.3 et il enverra à 132.227.2.1 le message **ICMP Host redirect** 132.227.4.1 via 132.227.2.2. Toute machine qui reçoit un tel message doit le traiter et mettre à jour sa table de routage.

Encapsulation Physique

Supposons qu'une machine **A** d'adresse **IP** IPA, et d'adresse Ethernet EtA envoie un paquet à **B** (IPB). Le paquet fabriqué par **IP** sur **A** contient dans son en-tête **IP** :

- IPA comme adresse source
- IPB comme adresse destination.

Si **B** (EtB) est sur le même réseau, la trame Ethernet contient :

- EtA comme adresse source
- EtB comme adresse destination.

Si **B** est sur un autre réseau, **C** (IPC EtC) est passerelle, et la trame Ethernet contient :

- EtA comme adresse source
- EtC comme adresse destination.

Machine Passerelle

La fonction d'une passerelle est de connecter plusieurs réseaux physiques, tout en interdisant le trafic "parasite". Elle ne doit pas transmettre sur un réseau les paquets qui ne lui sont pas destinés.

Son rôle est aussi d'informer les autres machines (simples ou passerelles) sur les erreurs de routage par l'intermédiaire des messages **ICMP**.

Afin d'optimiser le routage, elle doit être capable d'échanger des informations sur les différents réseaux, et sur la manière d'y accéder avec d'autres passerelles. Elle doit implémenter un protocole de type **IGP** (Internal Gateway Protocol) pour échanger ces informations.

Une passerelle doit également implémenter une notion d'accessibilité d'un réseau et éventuellement rerouter les paquets vers une autre passerelle.

Routage pour une passerelle

Le routage est plus compliqué pour une passerelle que pour une machine simple. Elle doit implémenter une notion de “distance” vers un réseau ou un sous-réseau et choisir le chemin le plus court. La distance vers un réseau est comptée en nombre de passerelles qu'il faut traverser pour atteindre un réseau donné.

Une passerelle doit également s'assurer que les autres passerelles peuvent être atteintes sans tester en permanence leur bon fonctionnement. Elle utilise les paquets **IP** qui transitent pour déterminer les réseaux accessibles (et donc les passerelles qui fonctionnent).

Lorsqu'une passerelle a le choix entre plusieurs routes possibles elle utilise le chemin le plus court ou la dernière passerelle qui fonctionnait.

Une passerelle doit éviter d'utiliser les messages **ICMP** "echo request" pour savoir si un réseau est accessible ; elle doit utiliser un protocole qui permet de gérer les routes. Le protocole le plus utilisé est **RIP** (Route Information Protocol).

RIP comme **ARP**, utilise les broadcast (au niveau IP). Chaque passerelle envoie régulièrement un broadcast donnant sa table de routage. L'ensemble des passerelles sur le même réseau écoutent ces messages, mettent à jour leur table de routage, et peuvent ensuite retransmettre ces informations sur un autre réseau.

Une passerelle peut être **active** ou **passive** pour **RIP** suivant qu'elle envoie des informations régulièrement ou se contente d'écouter celles des autres. Afin d'éviter des paquets inutiles ce protocole n'est utilisé qu'à l'intérieur d'un même réseau **IP** découpé en sous-réseaux locaux.

Lorsqu'une passerelle détermine qu'une machine est inaccessible (en panne . . .) elle envoie le message **ICMP** "host unreachable", et le message "net unreachable" si elle détermine que l'ensemble d'un réseau **IP** ne peut être atteint.

Ce type de message n'indique pas que le chemin est incorrect, mais simplement qu'on ne peut acheminer un paquet vers la machine ou le réseau considéré. Ce n'est pas une erreur **IP** il est simplement interprété par d'autres passerelles pour éviter d'envoyer de nouveaux paquets.

Supposons qu'une passerelle **A** envoie le message **réseau R inaccessible** à une passerelle **B**, et que **B** sache trouver un chemin vers (router) **R** via une autre passerelle **C**. Une fois que **B** a reçu le message elle utilise **C** comme intermédiaire jusqu'à ce qu'un paquet provenant de **R** arrive via **A** (ce qui est interprété comme une indication de fonctionnement).

Agglomérer plusieurs réseaux (RFC 1519, 1812)

La pénurie d'adresses de réseau conduit autoriser le groupage de plusieurs adresses de réseau. Le masque a alors la forme $/n$ où n est le nombre de bits à 1 au début de l'adresse.

La notation **CIDR** (Classless Inter Domain Routing) permet de donner une adresse IP accompagnée de son masque.

Ainsi 132.227.1.1/15 désigne les deux réseaux de classe B 132.226.0.0 et 132.227.0.0 puisque $226 = 11100010$ et $227 = 111100011$ qui ont les 7 premiers bits en commun.

Plus généralement un masque est une suite quelconque de 4 octets.

Pour les passerelles le routage est simplifié puisqu'elles peuvent agréger les routes pour des réseaux ayant le même préfixe.

La fragmentation

Une passerelle peut être amenée à fragmenter un paquet en plusieurs morceaux pour s'adapter au **MTU** d'une ligne donnée.

Le deuxième champ de l'en-tête **IP** sert à la fragmentation :

- le numéro d'identification du paquet est donné par la machine émettrice du paquet, il sert à retrouver les morceaux d'un même paquet.
- la partie **flags** sert à dire si on peut fragmenter le paquet, s'il fait partie d'un groupe, si ce paquet est le dernier du groupe.
- la partie numéro d'ordre donne l'endroit en mots de 8 bits dans le paquet d'origine où se situe ce fragment. Ceci permet de placer n'importe quel morceau "à sa place" même si les morceaux qui précèdent ne sont pas arrivés.

La méthode de fragmentation est simple :

- si la taille du paquet est inférieure au **MTU** de la ligne, on l'envoie tel quel,
- sinon on coupe le paquet en bouts multiples de 8 octets inférieurs au MTU et l'en-tête IP de la ligne,
- pour chaque bout on construit un en-tête **IP** avec le numéro d'identification originel du paquet et la position de ce morceau dans le paquet d'origine. Ainsi chaque fragment peut suivre son chemin puisque il a un en-tête IP complet.

Un émetteur peut décider qu'un paquet ne doit pas être fragmenté, il positionne alors un **flag** ("don't fragment"). Si en franchissant des passerelles ce paquet doit être coupé, la passerelle refuse le paquet et envoie un message **ICMP** prévenant que le paquet est trop gros. L'émetteur doit alors ré-émettre un paquet plus petit, ou autoriser la fragmentation.

Pour ré-assembler un paquet identifié par les adresses sources, destination et le numéro d'identification, il suffit de :

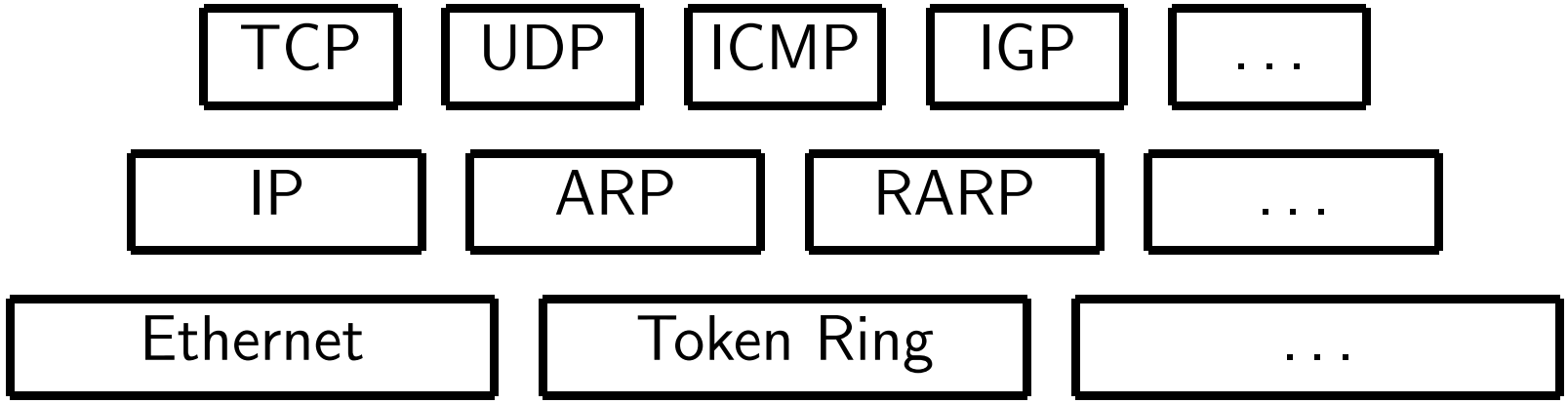
- examiner les **flags** pour savoir s'il a été coupé,
- examiner les numéros d'ordre pour savoir à quel endroit du paquet d'origine se trouve ce morceau.

Lorsqu'une machine reçoit un paquet fragmenté, elle démarre une temporisation qui expire en fonction du **Time to live** (TTL) de chacun des morceaux, si le paquet n'a pas été complètement reformé à expiration de ce temps, il est détruit.

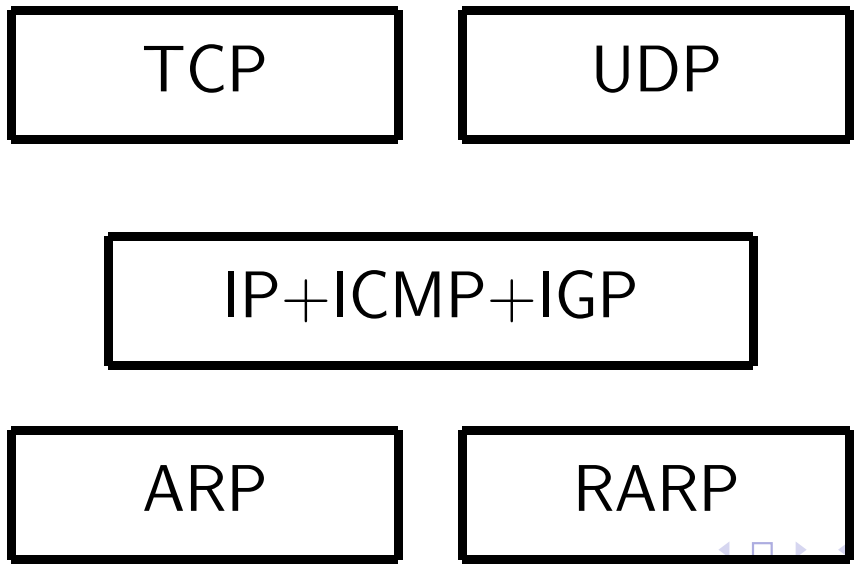
Une passerelle n'est pas obligée de reformer un paquet entier avant de le retransmettre, en le re-fragmentant si nécessaire. Les **box** ATM reconstituent le paquet IP en entier.

Physique-Logique

Aspect physique des protocoles



Aspect logique des protocoles



Gérer son espace d'addressage DHCP

Avoir une adresse IP, connaître une passerelle sur son réseau physique et pouvoir résoudre les noms de machines.
Dynamic Hub Control Protocol est un protocole de niveau physique pour obtenir ces informations. Une machine envoie une requête **DHCP Query** en diffusion sur le réseau en donnant son adresse MAC (physique).

Sur le réseau local un serveur DHCP est à l'écoute et possède

- une plage d'adresses dynamiques attribuables aux stations et
- une liste d'association adresses MAC/adresses IP.

Le serveur peut attribuer une adresse IP en fonction de l'adresse MAC et donner à la station l'adresse IP d'un routeur et celle d'un serveur DNS (en fait deux) (**DHCP Reply**).

L'expolsion de l'Internet

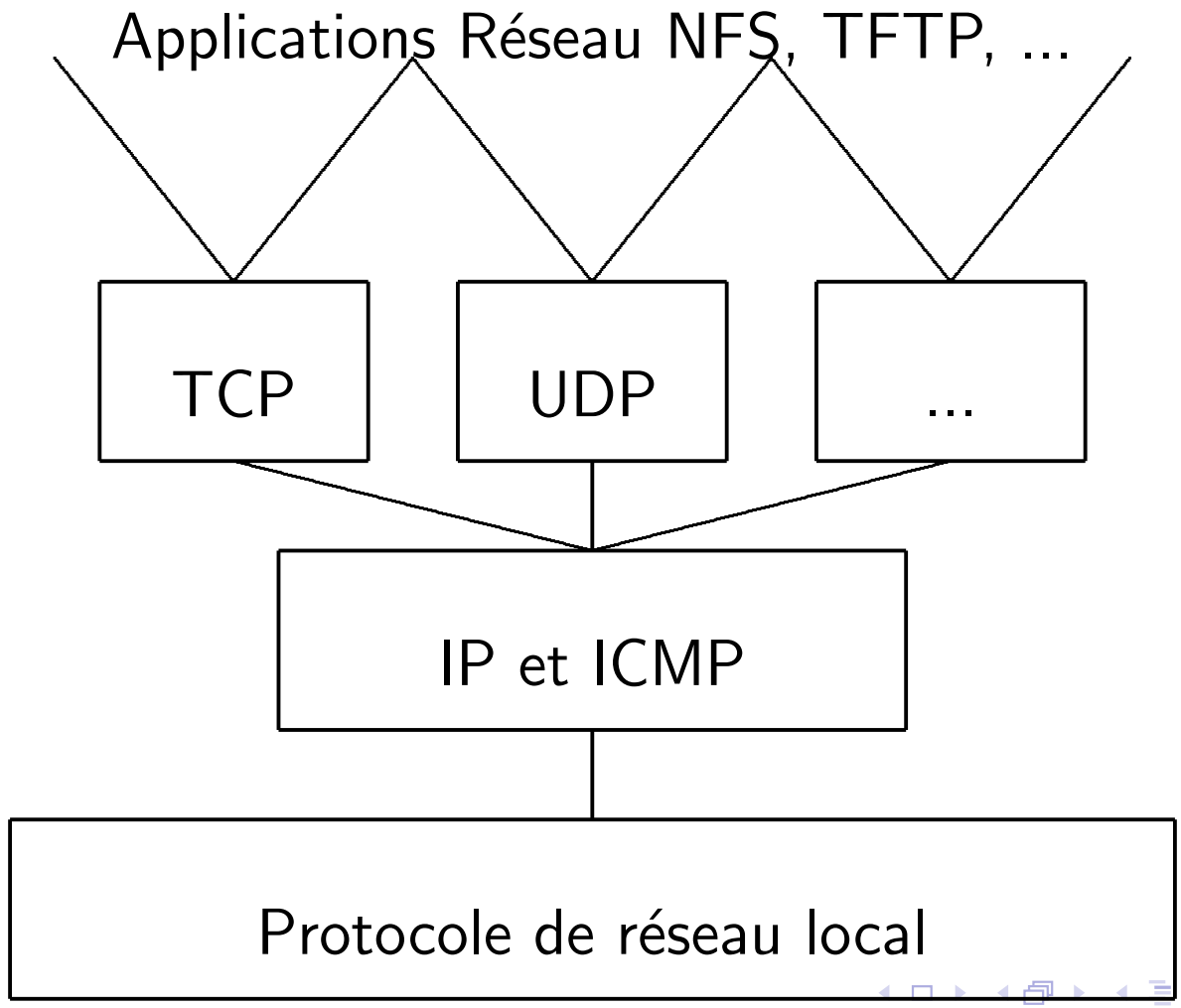
Une adresse publique IP est partagée par tout un réseau privé. Les **NAT** (Network Adress Translation) permettent d'accéder à certains services Internet.
Un Routeur NAT supporte des protocoles de niveau transport (UDP, TCP) il prend à son compte la demande d'un client ayant une adresse privée et tient à jour une table d'assocation entre le port qu'il créé, le port du serveur et l'adresse IP privée ainsi que le port du client.
Ainsi C fait une demande de connexion TCP sur le port 80 du serveur S, il obtient le port 1234. La connexion
TCP :C-1234 :S :80 est relayée par le routeur R qui fait une demande de connexion TCP sur le port 80 de S et il obtient le port 12345. De l'extérieur S voit que R est connecté via le port 12345. Lorsqu'il reçoit des données sur le port 12345, R les renvoie à C sur le port 1234. Il a mémorisé TCP.12345 vers TCP.C-1234.

Offrir des services sur un Réseau privé

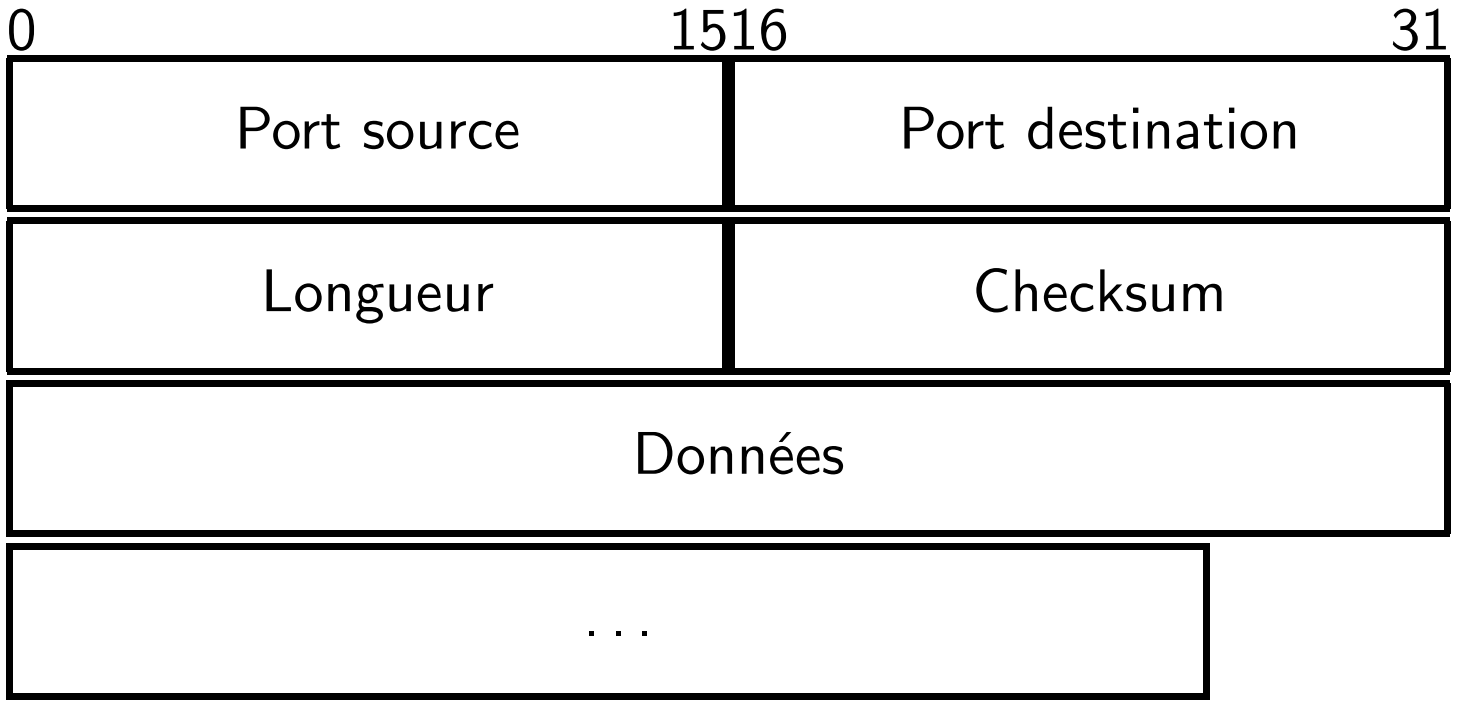
Seule l'adresse IP d'un routeur est publique, il doit transmettre les connexion vers un port donné sur le réseau privé.
Ainsi on indique au routeur R que les connexions vers le port TCP 80 seront redirigées vers le port TCP 80 de la machine S ayant une adresse IP privée.
On doit identifier tous les services que l'on veut offrir !
Les NAT ne sont pas transparents pour IP !

UDP

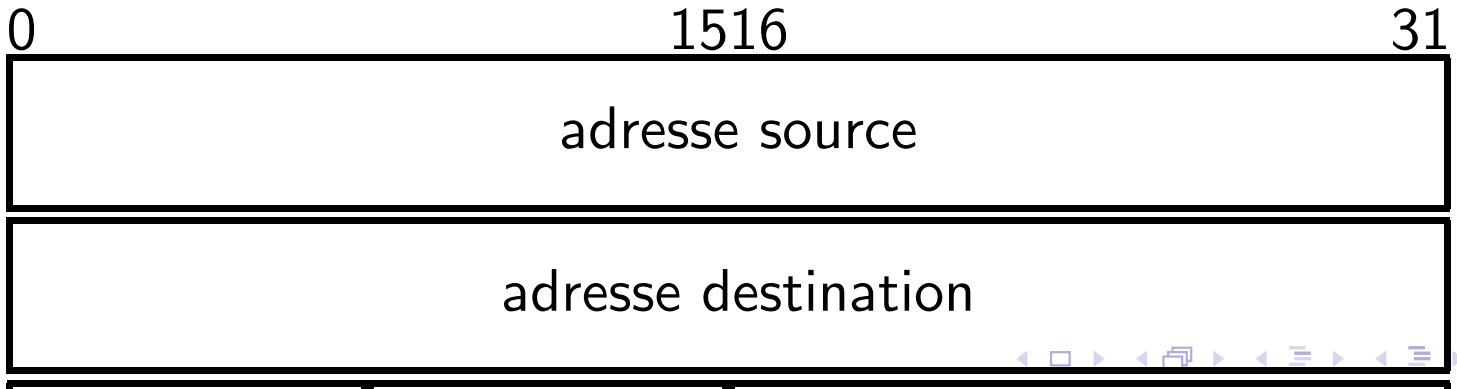
User Datagram Protocol est le plus simple des protocoles construits par dessus **IP**. Il offre en mode **non-connecté** un transport simple de paquets en laissant l'application réseau faire le travail de contrôle.



Un datagramme UDP



On ajoute en plus un **pseudo-header** :



Obtenir un port UDP : STUN

UDP ne contrôle pas une ligne entre deux machines mais simplement un port sur lequel des données sont reçues et transmises à l'application.
Certains protocoles (jeux vidéo, téléphonie...) nécessitent l'ouverture de ports serveurs sur des adresses privées.
Le protocole STUN (Simple Traversal of UDP over NAT, RFC 3489) permet des découvrir le port UDP alloué par un routeur NAT. Un serveur STUN sur un réseau public (port UDP 190) reçoit une demande et il transmet en réponse l'adresse IP ainsi que le port qui l'ont contacté. Un client STUN peut ainsi apprendre l'adresse IP publique et le port que le routeur utilise.
N'importe quelle application peut alors envoyer des données sur ce port et cette adresse publics, elles seront transmises à l'application qui peut l'utiliser comme un port serveur.

Négocier l'ouverture d'un port : UPNP

Pour du transfert de fichier d'égal à égal (peer to peer) on doit avoir un port TCP ouvert sur chaque hôte. Un routeur peut supporter l'Universal Plug and Play (**UPnP**) qui permet à un client de demander l'ouverture d'un port sur une adresse publique.
Les protocoles IGD (Internet Gateway Device, RFC 6970) et PCP (Port Control Protocol, RFC 6887) officialisent l'ouverture dynamique de ports sur un routeur.

TCP

TCP est un protocole de contrôle en **mode connecté**, il :

- transfère des données,
- s'assure de la fiabilité du transfert,
- utilise un mécanisme de contrôle de flux,
- gère plusieurs applications (multiplexage),
- ouvre et ferme des connexions.

Les niveaux supérieurs ne sont pas avertis de la taille des données effectivement transférées.

Comme le niveau inférieur (**IP**) travaille indépendamment de **TCP**, la ligne est de type **full duplex**.

Le transfert de données

Les données que l'on transfère viennent d'un **flot de données** qui est coupé en morceaux de **taille arbitraire**. L'application peut si elle le désire donner une taille maximale aux datagrammes, ceci se fait à l'établissement de la liaison.

C'est **TCP** qui décide en général quand il doit passer un paquet de données à **IP** pour être acheminées. L'application peut toutefois demander que le flot de données soit transmis jusqu'à un certain point. **TCP** offre une fonction **push**.

TCP offre une notion de **données urgentes** (urgent data) pour l'application, il ne les traite pas spécialement, l'application qui reçoit les données est simplement informée de leur "urgence" et les traite.

L'établissement de la liaison

TCP définit la notion de **ports standards** (well known ports) pour des applications spécifiques (Telnet 23). On distingue deux modes d'ouverture :

- **passif** (serveur) : on accepte des connexions venant de n'importe où,
- **actif** (client) : on crée les paramètres nécessaires à l'établissement de la ligne.

Chaque connexion utilise un **TCB** (Transmission Control Block) pour garder les paramètres nécessaires à son fonctionnement.

La liaison entre deux **TCP** est établie après une étape de **synchronisation** entre les deux machines. Une connexion peut être utilisée plusieurs fois à la suite, on a alors différentes **incarnations** d'une connexion.

La fiabilité

Un flot de données est transmis de manière **fiable** et **dans l'ordre** initial au receveur. La fiabilité est assurée par l'utilisation de **numéros de séquence** et d'**accusés de réceptions**.

Chaque octet de données a un numéro d'ordre, on transmet l'ordre du premier octet de données dans un paquet en même temps que ce paquet.

Les segments ont également un numéro d'accusé de réception qui est le numéro d'ordre du prochain octet que l'on attend.

Lorsque **TCP** envoie un segment, il le met dans une **pile de retransmission** (retransmission queue). Un **temporisateur** (timer) est démarré, et si un accusé de réception arrive pour les octets qu'il contient, on l'enlève de la pile ; le paquet est retransmis après un **temps mort** (time-out).

Les fenêtres

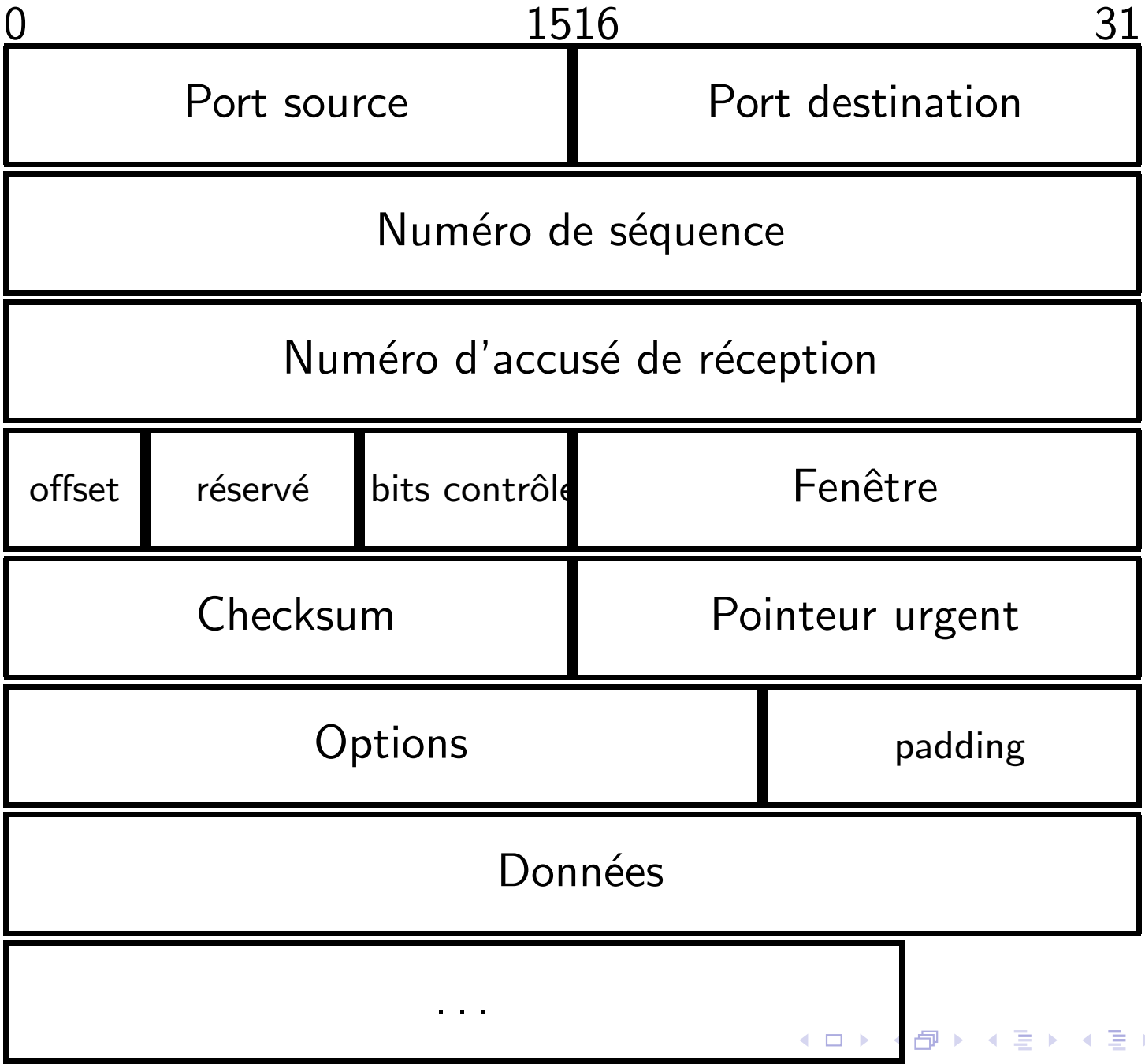
Pour utiliser au mieux la ligne **full-duplex** le mécanisme de **contrôle de flux** (flow control) utilise des **fenêtres**.

Une fenêtre est le nombre d'octets à partir du numéro d'accusé de réception du paquet que **TCP** peut recevoir.

Dans **l'espace des numéros d'ordre** d'émission on a une **fenêtre d'émission**, et une fenêtre de réception dans l'espace des numéros d'ordre de réception (accusé de réception).

On peut ainsi envoyer des données "en avance" sur un accusé de réception. Ce qui permet d'utiliser beaucoup mieux la ligne qu'un mécanisme plus simple qui attend un accusé de réception après chaque segment envoyé.

Les segments TCP



Comme dans **UDP** un **pseudo-header** permet de déterminer :

- l'adresse source
- l'adresse destination
- le protocole (TCP)
- la longueur TCP.

Le **checksum** est calculé sur le segment et sur le pseudo-header.

Les options permettent en début de connexion (phase de synchronisation) de donner une taille maximale aux segments envoyés.

Comme la longueur de l'en-tête est variable, le champ **offset** donne le nombre de mots de 32 bits que comporte l'en-tête **TCP**.

Le champ **bits contrôle** (6 bits) indique :

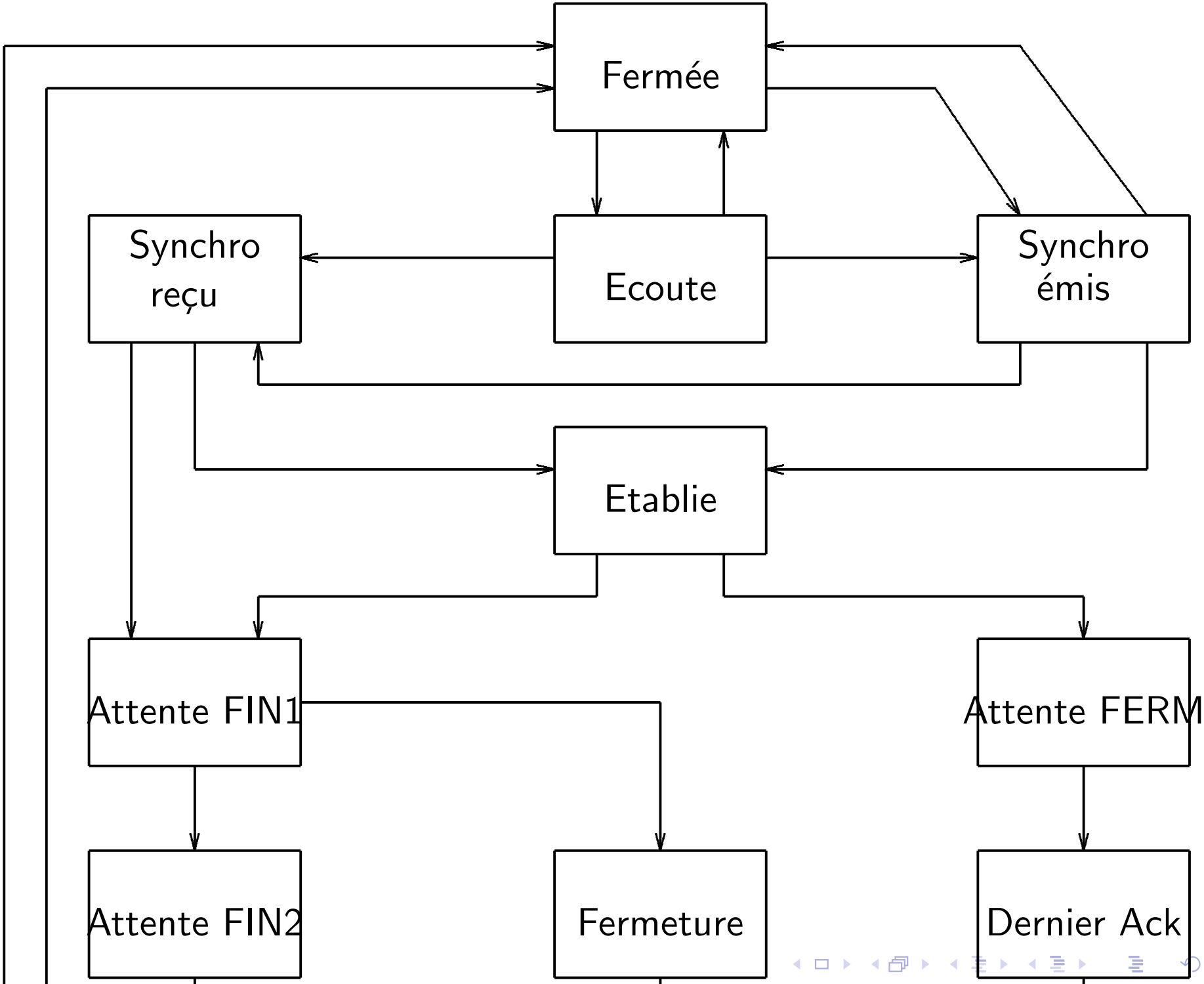
- si le pointeur **données urgentes** est significatif, il donne alors la fin des données urgentes dans le segment,
- si le champ **accusé de réception** est significatif, il donne alors le **numéro de séquence** que l'envoyeur attend ensuite,
- si la fonction **push** a été utilisée (il n'y a pas de relation entre un push et une fin de segment),
- si la connexion doit être réinitialisée,
- si l'on est en phase de **synchronisation**,
- si l'on est en phase de **clôture** de la ligne.

Le deuxième octet du paquet est soit le champ numéro de séquence du premier octet de données, ou un **numéro de synchronisation** en phase de synchronisation.

Dans le **TCB** il y a des variables qui donnent les **ports** source et cible, des pointeurs vers les buffers émission et réception, vers la pile de retransmission, et vers le segment courant.

Les espaces **d'émission** et de **réception** sont **finis** (32 bits), **TCP** ne limite pas la taille des données, et autorise plusieurs incarnations d'une même connexion, il faut donc faire attention à la numérotation des octets (modulo 2^{32}) pour ne pas confondre d'anciennes données provenant d'une connection fermée avec des données de la connexion en cours.

La phase de synchronisation permet de régler les numéros de séquence et les numéros d'accusé de réception.


Fermée
Synchro reçu
Ecoute
Synchro émis
Etablie
Attente FIN1
Attente FERM
Attente FIN2
Fermeture
Dernier Ack

À l'initialisation de la ligne il faut que les deux **TCP** qui communiquent se "mettent en phase". Pour éviter de confondre les données que l'on va envoyer avec d'autres (provenant d'anciennes connexions) les deux **TCP** vont échanger leur **ISN** (Initial Sequence Numbers). L'**ISS** (Initial Send Sequence) et l'**IRS** (Initial Receive Sequence) sont échangés.

Ces nombres sont régis par une horloge qui incrémente un compteur toutes les 4 millisecondes, les **ISN** font 32 bits et cyclent toutes les 5 heures. Si la durée de vie d'un paquet (**MSL**) est inférieure à ce délai les **ISN** seront uniques pour une connexion.

On envoie une synchronisation lorsque le bit de contrôle **SYN** est à 1 ; un accusé de réception lorsque le bit **ACK** est à 1.

Initialisation de la connexion

La connexion entre **A** et **B** se fait en échangeant les numéros initiaux (le 3 way handshake). Soit schématiquement :

- **A** envoie à **B** “bonjour” mon numéro est X
- **B** envoie à **A** “OK” ton numéro est X
- **B** envoie à **A** “bonjour” mon numéro est Y
- **A** envoie à **B** “OK” ton numéro est Y

X est le début de **l'espace d'émission pour A** et le début de **l'espace de réception pour B**

Y est le début de **l'espace d'émission pour B** et le début de **l'espace de réception pour A**

A utilise les numéros donnés par **B** pour les accusés de réception qu'il envoie et ses propres numéros pour ceux qu'il reçoit.

Exemple d'initialisation

TCP A		**TCP B**
Fermée		**attente**
Syn envoyé $\rightarrow$	$< SEQ = 100 >< CTL = SYN >$	$\rightarrow$ **Syn reçu**
Etablie $\leftarrow$	$< SEQ = 300 >< ACK = 101 >< CTL = SYN, ACK >$	$\leftarrow$ **Syn reçu**
Etablie $\rightarrow$	$< SEQ = 101 >< ACK = 301 >< CTL = ACK >$	$\rightarrow$**Etablie**

TCP A		**TCP B**
Fermée		**Fermée**
Syn envoyé $\rightarrow$	$< SEQ = 100 >< CTL = SYN >$	...
Syn reçu $\leftarrow$	$< SEQ = 300 >< CTL = SYN >$	$\leftarrow$ **Syn envoyé**
...	$< SEQ = 101 >< CTL = SYN >$	$\rightarrow$ **Syn reçu**
Syn reçu $\rightarrow$	$< SEQ = 100 >< ACK = 301 >< CTL = SYN, ACK >$	...
Etablie $\rightarrow$	$< SEQ = 300 >< ACK = 101 >< CTL = SYN, ACK >$	**Syn reçu**
... $\rightarrow$	$< SEQ = 101 >< ACK = 301 >< CTL = ACK >$	**Etablie**

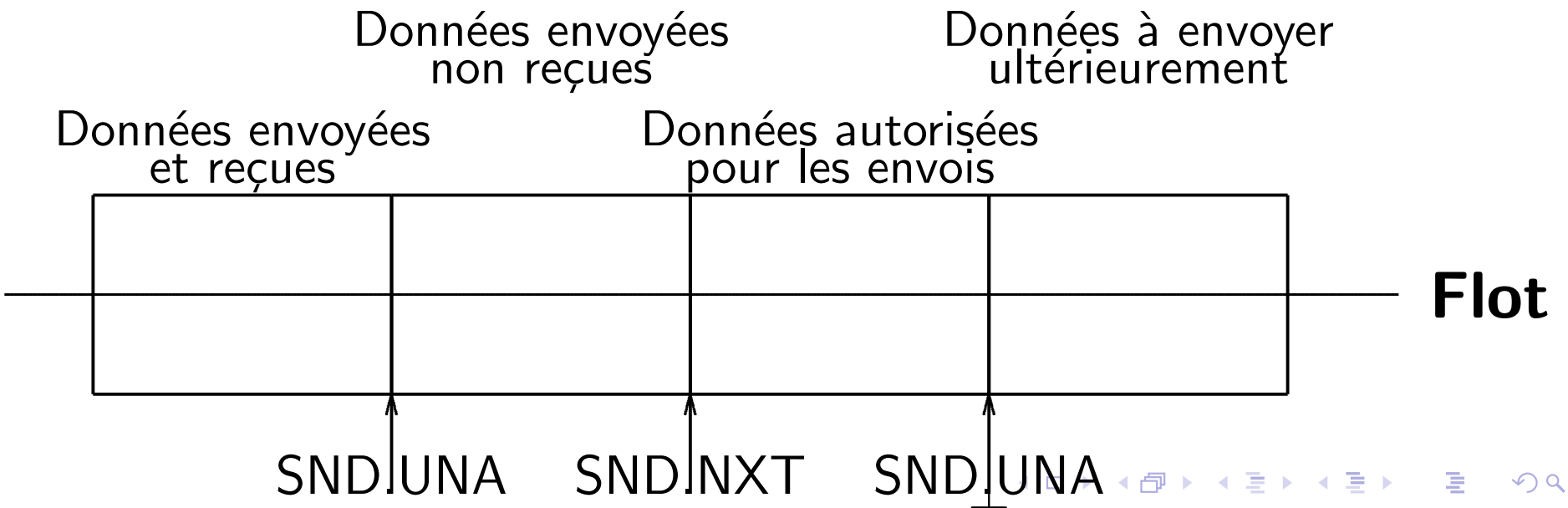
les **ACK** ne consomment pas d'espace de données (émission ou réception).

L'espace d'émission

Le **TCB** contient les variables nécessaires à la gestion de l'émission, dont :

- `SND.UNA` : octets sans accusé de réception,
- `SND.NXT` : octets à envoyer,
- `SND.WND` : fenêtre d'envoi,
- `SND.UP` : pointeur urgent émission,

Soit sur le flot de données :

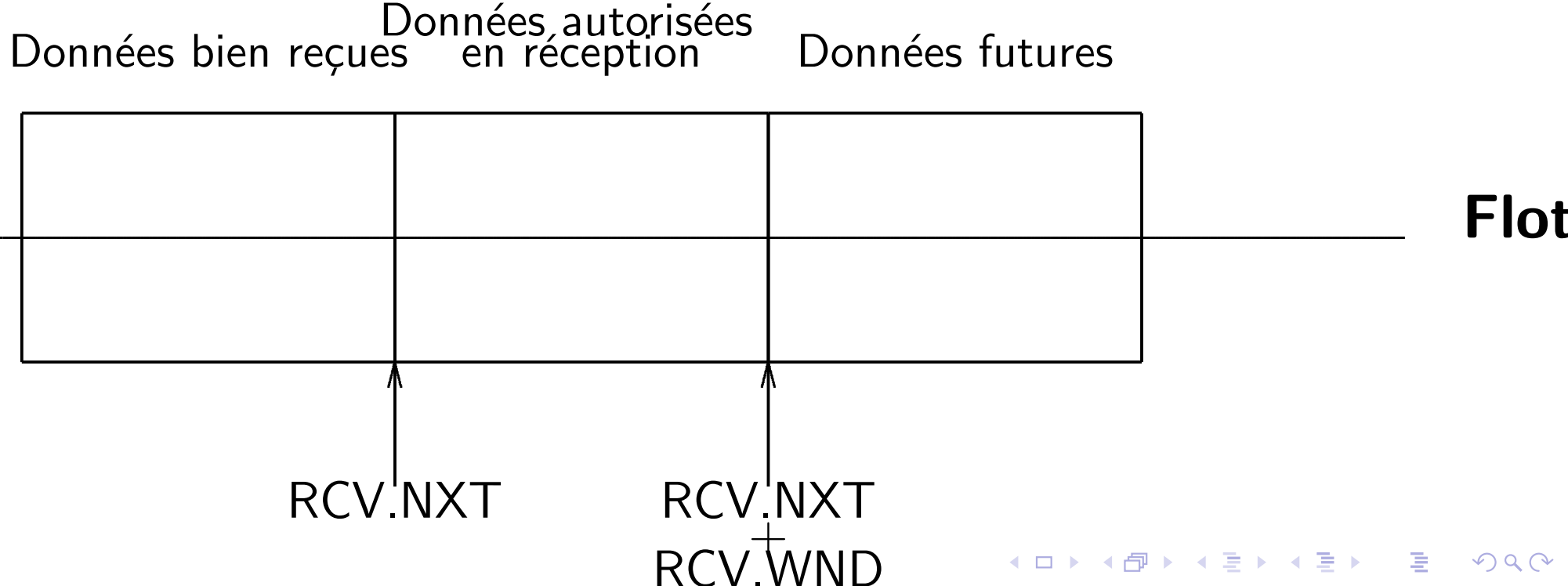


L'espace de réception

Le **TCB** contient les variables nécessaires à la gestion de la réception, dont :

- `RCV.NXT` : données à recevoir,
- `RCV.WND` : fenêtre de réception,
- `RCV.UP` : pointeur urgent réception.

Soit sur le flot de données :



La gestion des données

Il faut pouvoir :

(a) déterminer si un numéro d'accusé de réception se rapporte bien à des octets envoyés non encore bien reçus,

(b) déterminer si tous les numéros de séquence d'un segment donné ont été bien reçus,

(c) déterminer si un segment qui arrive contient des numéros de séquence qui sont bien attendus.

La retransmission se fait au bout d'un **time-out** qui est déterminé dynamiquement par l'émetteur ou après 3 acquitements dupliqués (fast retransmit)
On mesure le RTT (Round Trip Time, le temps entre un envoi de donnée et son acquitement) R moyen de la ligne, son écart type E et on retransmet après $R + 4E$.

Gestion de la fenêtre

La fenêtre d'émission est gérée dynamiquement : le récepteur indique combien de données il peut recevoir. Une "grande" fenêtre indique que l'émetteur peut envoyer beaucoup de données, une petite causera des segments plus petits. Si la fenêtre est trop petite, **TCP** devra attendre entre chaque segment envoyé.

Les données urgentes

Le pointeur d'urgence donne un point dans le flot de données où se **terminent** les données urgentes ; le **TCP** receveur signale à son application qu'elle doit passer en **mode urgent** jusqu'à ce que le pointeur d'urgence soit dépassé (traité par TCP), il lui signale alors de passer en **mode normal**.