Réseaux : IP, Protocoles et Communications

HAI404I : Licence 2 Informatique

Anne-Elisabeth Baert – baert@lirmm.fr

LMD5

Table des matières

Chapitre 0 : Informations pratiques

Informations pratiques

Informations pratiques

Cours 12hh, TD/ TP 16 séances de 1,5 heures.

3 groupes de TD/TP : groupe A, B et C-

Les TDs/TPS et le cours sont obligatoires pour assurer une réussite au module.

Module sur 4 ECTS

MCC : Examen final AUCUN document sauf une feuille A4.

Pré-requis

Système du premier semestre et programmation C/C++. Revoir son cours avant les séances de TDs et TPs ; il *faut* approfondir le cours, on peut même lire des livres !!!!!

Conseils usuels

"Venir " en Cours, TDs, TPs, poser des questions en Cours et en TDs

Revoir son cours avant chaque séance de TDs et TPs ; il *faut* approfondir le cours.

Informations

Toutes les informations sont sur le Moodle.

Objectifs - Programme du module

Le réseau en Informatique

Connaître les principes et mécanismes des réseaux pour un Informaticien. Répondre aux questions

- Qu'est-ce qu'un réseau ? A quoi çà sert ?
- Comment mettre en œuvre et utiliser des applications réseaux ?
- Qu'est-ce qui relève du **travail d'informaticien** dans les réseaux ?

Compétences acquises en HAI404

Compétences

Savoir calculer un débit connaitre la latence,calculer l'efficacité d'un échange, calculer un débit utile

Savoir calculer le temps de transmission total d'un échange

Connaître les couches OSI et savoir décrire l'encapsulation et la désencapsulation

Connaître l'architecture TCP/IP, la taille des en-têtes, l'encapsulation

Connaître la différence entre mode connecté ou non connecté

Connaître l'architecture TCP/IP, le rôle des différents protocoles, la taille des en-têtes

Savoir donner les caractéristiques d'une adresse IP, reconnaître les classes, les adresses privées…

Savoir donner une adresse de diffusion, la première et dernière adresse machine…

Savoir écrire un masque dans la forme décimale et CIDR
Savoir découper un réseau en plusieurs sous-réseaux
Savoir faire et agréger une table de routage
Savoir lire une table de routage et faire le schéma du réseau associé
Savoir décrire les échanges de trames suite à un ping (ARP/ICMP…)

Compétences

avoir donner les caractéristiques d'une adresse IP, reconnaître les classes, les adresses privées…
Savoir donner une adresse de diffusion,l première et dernière adresse machine…
Savoir écrire un masque dans la forme décimale et CIDR
Savoir découper un réseau en plusieurs sous-réseaux
Savoir faire et agréger une table de routage
Savoir lire une table de routage et faire le schéma du réseau associé
Savoir décrire les échanges de trames suite à un ping (ARP/ICMP…)
Comprendre la notion de client et de serveur sur un réseau informatique.
Savoir programmer des applications client/serveur

Programme

Le programme

Introduction générale : éléments de base, architecture, les couches.
Monde Internet : Historique, caractéristiques, adresses., encapsulation
Couche réseaux : configuration de réseaux et sous-réseaux. Problèmes de routage
Couche transport UDP et TCP
Les protocoles d'application, messagerie, transfert de fichiers., DNS.
Programmation réseaux : Types de serveurs., modes de connexion, protocoles sous-jacents.

1 Chapitre 1 : Introduction aux réseaux

1.1 Le problème, les définitions de base

Table des matières

Présentation du problème

Le besoin

Besoins des **utilisateurs** : échanger des données entre eux.

Besoin d'applications qui *communiquent entre elles, échangent des données et partagent des ressources communes.*

Hypothèses :

Les applications sont sur des ordinateurs.

Les ordinateurs fonctionnent de façon autonome.

Les ordinateurs disposent d'un accès au *périphérique réseau ; en terme de système d'exploitation, il y a un contrôleur (une carte réseau) et un pilote permettant de lire et écrire sur le périphérique. La particularité de ce périphérique est qu'il est partagé.*

Les réponses technologiques pour échanger des données

Les liaisons physiques

Exemples : Ondes, Câbles, fibres optiques,…

Chaque support a ses propres caractéristiques, essentiellement une distance liée à un débit, ainsi qu'une distance maximale.

Des protocoles :

Accords sur des règles permettant aux entités communicantes de se comprendre : ce sont les "méthodes" communes.

Des couches :

Comme tout système informatique, en réseau il y a une construction par couches successives, tant **matérielles** que **logicielles**.

1.2 Les protocoles réseaux

Les protocoles

Definition 1. Un protocole est une méthode standard qui permet la communication entre des processus , c'est-à-dire un ensemble de règles et de procédures à respecter pour émettre et recevoir des données sur un réseau.

Le terme *protocole* est associé à des notions très différentes, en fonction du domaine d'utilisation (échanges de fichiers, transmission d'erreur, ...).

Exemples de protocoles
- Http, https
- POP, SMTP,
- TCP/IP, UDP
- DHCP,
- ICMP.

Un protocole simple : Une conversation téléphonique

Le protocole commence lorsque la personne jointe décroche :

1. la personne jointe doit dire quelque chose :
   - *allo*
   - *bonjour*
   - *ne raccrochez pas*
2. à l'une des deux premières réponses, la personne appelante répond pour démarrer la conversation,
3. dans les autres cas, elle continue à patienter (retour au début du protocole),
4. lorsqu'enfin les deux personnes peuvent discuter, le protocole impose de ne pas parler les deux à la fois,
5. pour terminer la conversation une des deux personnes doit l' annoncer,
6. l'autre personne peut refuser mais un accord mutuel est nécessaire,
7. la conversation se termine lorsque l'une des deux a raccroché.

Remarque : Si une personne a raccroché sans avertir, elle n'a pas respecté le protocole.

Exercice

Donner l'algorithme d'une conversation téléphonique.

```
fini=faux ;
(non fini) lireUnMessage() ;
(premierCaractère == 'F') fini=vrai ;
(premierCaractère == 'B') repondreBonjour(nomDemandeur,monNom) ;
dialoguer() ;
expedierErreur() ;
```

et à plusieurs ca donnerait quoi ?

Protocoles : conclusion

Un **protocole** ne s'exprime pas toujours sous forme d'un algorithme : un algorithme, est une description des actions faites par une entité pour se conformer à ce protocole.

Il y a toujours négociation entre **plusieurs** (au moins deux) entités, souvent sous forme :
- de questions : chaîne de caractères reconnue expédiée par un demandeur, test portant sur un élément commun, …
- et de réponses : chaîne de caractères reconnue expédiée par le répondeur, réponse au test, …

permettant de réaliser, retarder ou interdire une action.

1.3 Les liaisons

Vocabulaire en réseau

Definition 2. Une structure de communication désigne *la forme logique sous laquelle les entités communiquent ; c'est la manière dont les données transitent entre les entités.*

On différenciera différentes communications
- le mode *point à point où seules deux entités concernées à la fois,*
- *le mode multipoints où plusieurs entités sont concernées (…), on parle de diffusion ou de multicast par exemple .*

Le vocabulaire réseau

Definition 3. Une topologie de réseau informatique correspond à 'architecture (physique ou logique) de celui-ci, définissant les liaisons entre les équipements du réseau et une hiérarchie éventuelle entre eux.

C'est en général une forme géométrique de la connexion physique : étoile, bus, anneau, arbre, maillage régulier,….

Noter qu'on peut traverser plusieurs réseaux de topologies différentes et on parlera alors d'interconnexion.

Example 4. Anneau à jeton : topologie d'anneau et communication point à point,
Bus ethernet : topologie de bus et communication par diffusion.

Definition 5. Une architecture réseau est un ensemble (empilement, hiérarchie) de protocoles. On parle alors de *modèles en couches.*

Une architecture en couches est définie et délimitée avec les notions de service, de protocole et d'interface.

Definition 6. Un service est *une description abstraite de fonctionnalités à l'aide de primitives (commandes ou événements) telles que demande de connexion ou réception de données.*

Un service réseau est une application exécutée depuis la couche d'application réseau et au-dessus. Il fournit des capacités de stockage, de manipulation, de présentation, de communication ou d'autres services qui sont souvent mises en œuvre en utilisant une architecture Les services réseau se basent sur les protocoles pour fournir, par exemple : des transferts de textes (SMS ?) ; ou de données (Internet ?) .

Definition 7. Un protocole est *un ensemble de messages et de règles d'échanges réalisant un service.*

Il définit les formats des en-têtes et les règles d'échange (syntaxe et sémantique des messages ?) En particulier :
- la délimitation des blocs de données échangés
- le contrôle de l'intégrité des données reçues
- l' organisation et contrôle de l'échange

Definition 8. Une interface (« point d'accès au service » dans la norme) est le moyen concret d'utiliser le service. Dans un programme, c'est typiquement un ensemble de fonctions de bibliothèque ou d'appels systèmes.

1.4 Les catégories des réseaux

Catégories de réseaux

Definition 9. Un domaine est une extension géographique dont les caractéristiques fondamentales sont le débit sur la distance. C'est une aire logique d'un réseau informatique.

4 catégories de réseaux :
- PAN : *Personal Area Network : Réseau Personnel* ,
- LAN : *Local Area Network- Réseaux locaux)* ,
- MAN : *Metropolitan Area Network : Réseaux métropolitains*
- WAN : *Wide Area Network : Réseaux étendus*

PAN (Personal Area Networks)

Definition 10. Un PAN est un réseau personnel qui interconnectent sur quelques mètres les équipements perso. (GSM, portables, montre, lunettes VR ...) d'un même utilisateur. C'est un réseau domestique ou réseau individuel.

Débit, Distance
Réseau de petite taille, oui mais laquelle ? et quel débit est possible ?

Example 11. USB, Firewire, bluetooth (802.15), Infrarouge(IR), Li-Fi, Zigbee ...

LAN : Local Area Networks

Definition 12 (LAN : Local Area Networks)**.** Systèmes de transmission de données à usage privé ou commercial. Bâtiment à câbler sur quelques centaines de mètres comme les réseaux intra-entreprise va permettre le transport de toutes les informations numériques de l'entreprise

Caractéristiques
Leur taille (restreinte), leur technologie de transmission (délai de transmission max connu), leur topologie (bus et anneau).

Débits
Quelques mégabits a une centaine de mégabits voir 10 Gbits/s pour Ethernet (IEEE 802.3)

Example 13. Les entreprises et les universités constituent également des réseaux locaux.

MAN : Metropolitan Area Networks

Definition 14 (MAN)**.** Les MAN (Metropolitan Area Networks)interconnectent plusieurs LAN géographiquement proches (au maximum quelques dizaines de km) à des débits importants .

Caractéristiques des MAN
Les MAN sont formés de **commutateurs** ou de routeurs interconnectés par des liens hauts débits (en général en fibre optique).

Example 15. Infrastructure multiservice téléphone /vidéo-surveillane +parkings dans une ville

WAN : Wide Area Network

Definition 16 (WAN : Wide Area Network)**.** Les WAN sont des réseaux étendus qui couvrent une très grande zone géographique et sont composés de plusieurs sous-réseaux (LAN) hétérogènes.

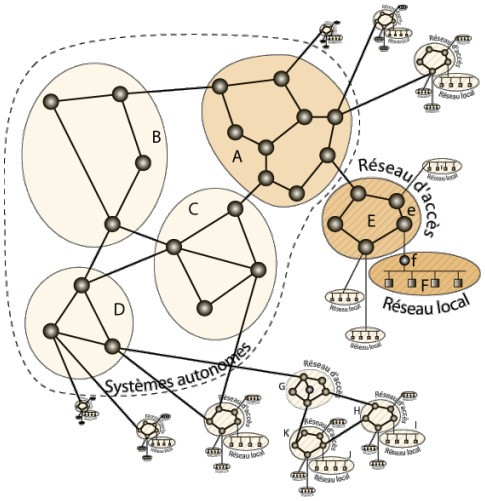
Caractéristiques des WAN
- Assurent la transmission des données numériques sur l'échelle d'un pays, continents.
- Réseau terrestre (grands réseaux de fibre optique)
- Réseau 3G,4G, ...hertzien et satellite

Internet
Le réseau Internet est un WAN.

1.5 Interconnexion des réseaux

Vrai problème
Il faut distinguer les réseaux physiques (média connectant physiquement plusieurs ordinateurs) et les réseaux logiques (*virtuel qui sont les interconnexions de plusieurs réseaux physiques).*

1.6 Le réseau Internet



et Internet !!!Source Interstices

Interconnexion

Definition 17. L'interconnexion-ARCEP- L'interconnexion désigne *le raccordement des différents réseaux de télécommunications entre eux afin de permettre à l'ensemble des utilisateurs de communiquer librement.*

Les problèmes
Sur des réseaux de même type, il suffit de faire passer les paquets d'un réseau à un autre. Sur des réseaux de type différents, il faut modifier les protocoles pour que les paquets puissent passer.

Les solutions
Besoin de modèles spécifiques et besoin de machines spécifiques qui assurent l'interconnexion.

Interconnexion

L'**interconnexion** est assurée par des ordinateurs simples ou des machines spécialisées. Leur rôle est d'assurer la **commutation**, c'est-à-dire , *le transfert de l'information entre un point d'entrée et un point de sortie* . Leur problème essentiel est l'efficacité.

Example 18. — concentrateurs (*hub) connexion entre eux plusieurs hôtes*
- *commutateurs (switch) reliebt divers éléments tout en segmentant le réseau,*
- *ponts (bridges) permettent de relier des réseaux locaux de même type,*
- *routeurs (router) permettent de relier de nombreux LAN pour permettre la circulation optimisée de données d'un LAN à un autre*
- *passerelles (gateway) permettent de relier des LAN de types différents .*

La commutation de circuits

Les circuits

Un circuit est un « tuyau » placé entre 1 émetteur et 1 transmetteur (par ex. Fils métalliques, fibres optiques ou ondes hertziennes), il appartient aux 2 entités qui communiquent.

Definition 19. La commutation de circuits *désigne le mécanisme consistant à rechercher différents circuits élémentaires pour réaliser un circuit plus complexe* . Elle est orientée connexion.

Comment ?

Grâce a la présence de noeuds (commutateurs de circuit) qui permettent de choisir un circuit libre en sortie en le « connectant » au circuit entrant et de mettre en place le circuit nécessaire à la communication entre les 2 entités.


Commutation manuelle avec des opératrices téléphoniques

La commutation de paquets

Definition 20. Utilisation du principe de signalisation : Pour mettre en place un circuit, il faut propager un ordre demandant aux autocommutateurs de mettre bout à bout des circuits élémentaires. Signalisation= Commandes +Propagation Mise en oeuvre de la fermeture, ouverture et maintien des circuits.

Definition 21. Dans la **commutation de paquets**, chaque entité d'interconnexion stocke la donnée (un paquet), détermine le prochain destinataire et lui fait suivre la donnée (*store & forward). Elle est orientée sans connexion*

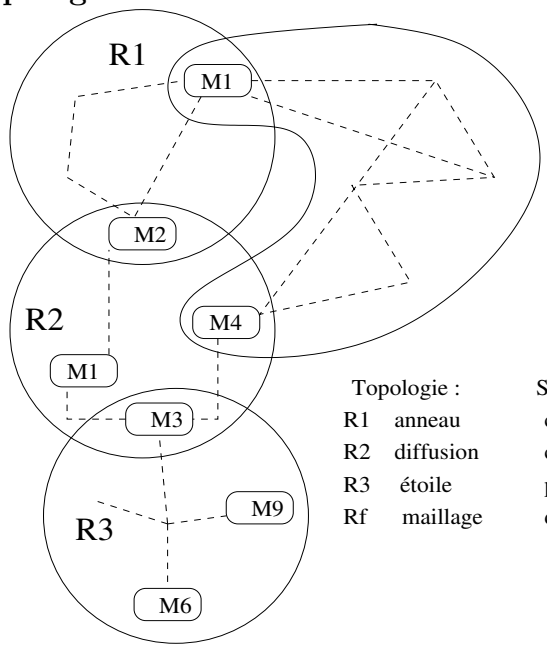
La commutation par paquets permet de fournir des flux de données à débit binaire variable, réalisés sous forme de séquences de paquets.

La commutation de circuits VS de paquets

Transfert de données
Les transferts de données s'opèrent dans un **mode** sans connexion ou connecté (on parle alors de circuit virtuel pour TCP/IP).

ARPANET.pngARPANETl'ancêtre d'Internet

Structure et topologie



Topologie :	Structure :
R1 anneau	coaxial, anneau jeton
R2 diffusion	coaxial ethernet
R3 étoile	paire torsadée ethernet
Rf maillage	divers, X25

Tout est normalisé ....

Plein de normes existent

normalisation	v24 v28 v35 v90	modems
	x21 x25 x29 x400	ccitt
	802.2 802.3 802.4	ieee
	802.11 802.11g	ieee, sans fil
	8802/2 8802/3(Ethernet) 8802/4	iso
	RFC 791 (Internet Protocol)	IETF/IRTF

Internet

Modèle OSI de l'ISO (cf. architecture, dans la suite de ce chapitre)

Internet ; est un ensemble de protocoles et d'applications.

1.7 Les caractéristiques des réseaux

Caractéristiques des supports

Unités
Unité utilisée : X bits par seconde. Notation : X bit/s.

Example 22. 55 Kbit/s était un "bon" débit lorsqu'on utilisait une ligne téléphonique avec un modem classique (appareil permettant la connexion d'un ordinateur personnel avec un réseau d'un opérateur, utilisant sa propre ligne téléphonique).

256 Kbit/s à 10 Mbit/s pour un modem ADSL. 100Mbit/s à 8 Gbit/s pour la fibre/

Caractéristiques des données

Le paquet
Unité fondamentale en réseau , le **paquet** qui est une suite d'octets. Autres noms en fonction de la spécificité du réseau : trame, cellule.

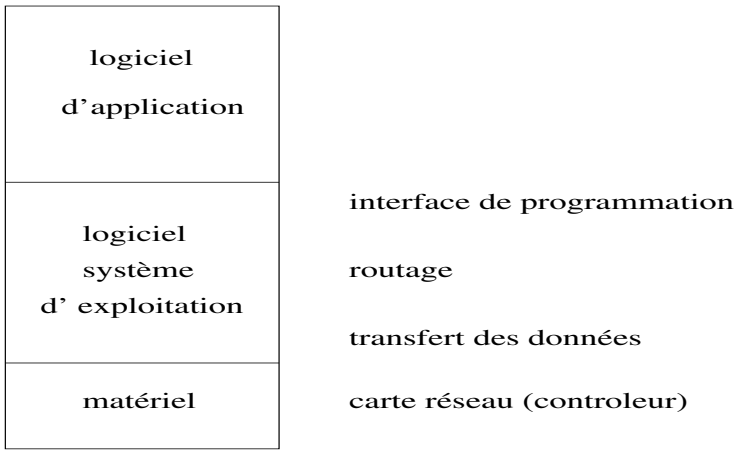
Format d'un paquet
Chaque paquet contient *les données à transmettre, une en-tête (contenant des adresses permettant d'acheminer le paquet) et une partie de contrôle (permettant de contrôler la validité des données)*

entête	donnée	contrôle

Question :
quelle destination ? locale ? suivante ? finale ?

Caractéristiques des logiciels

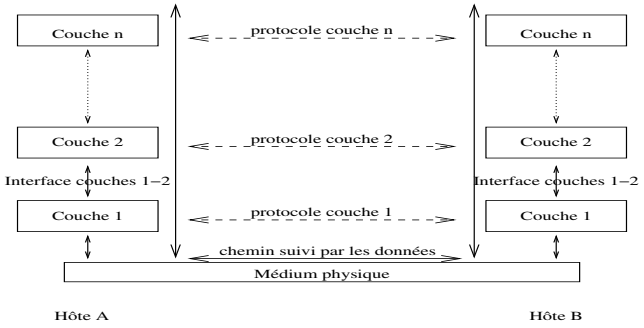
Les Logiciels et le réseau
Offrir des services aux applications : stockage, acheminement, accès au matériel, interface de programmation ...



2 Chapitre 2 : Les couches Réseaux

2.1 Les couches : architecture pour les réseaux

Architecture : construction en couches



Definition 23. Protocole : *règles et conventions utilisées entre couches homologues (hôtes différents).*

Definition 24. Interface : *règles et conventions utilisées entre couches voisines (même hôte).*

Encapsulation - Un Début

Principes

Chaque couche a ses propres impératifs liés à

- l'adressage (forme et codage de l'adresse),
- la taille (maximale et minimale) du paquet,
- la méthode de réalisation du contrôle.

Encapsulation

Elle est donc amenée lors de l'**expédition** à envelopper (on dit encapsuler) le paquet transmis par la couche au dessus dans la partie *donnée de son propre paquet,*

Encapsulation - Un Début

	entête n | donnée couche n | fin n	
	$\Downarrow$	
entête $n-1$	donnée couche $n-1$	fin $n-1$

Architecture : problème de couches

Historique

1977 :début d'une réflexion sur une architecture de réseau en couches, 1983 : définition du modèle OSI **O**pen : systèmes ouverts à la communication avec d'autres systèmes **S**ystems : ensemble des moyens informatiques (matériel et logiciel) contribuant au traitement et au

transfert de l'information. ISO = International Organization for Standardization en français Organisation Internationale de Normalisation **I**nterconnection

On s'intéresse dans cette partie surtout à la **diversité des problèmes**.

Definition 25. Le modèle OSI est un modèle *d'architecture de réseau qui propose une norme pour le nombre, le nom et la fonction de chaque couche* .

Que fait ce modèle ?
Il garantit que 2 systèmes hétérogènes pourront communiquer si :

- même ensemble de fonctions de communication,
- fonctions organisées dans le même ensemble de couches,
- les couches paires partagent le même protocole.

OSI-CISCO.pngModèle OSICISCO
OSI-CISCO.pngModèle TCP / IP

2.1.1 La couche physique

La couche physique

Definition 26 (La couche 1 : couche physique)**.** Elle permet la *transmission d'éléments binaires, transmet un flot de bits sans en connaître la signification ou la structure.*

Fonctions :

- Fournir les moyens mécaniques, électriques, fonctionnels, procéduraux pour l'activation, le maintien et la désactivation physiques destinées à la transmission des éléments binaires entre entités de liaisons.
- principalement les capacités électroniques
- **unité** traitée : un bit, au mieux un octet.

La couche physique

Matériels
Les modems, multiplexeurs.

Les caractéristiques
Ses caractéristiques induisent des performances en termes de débit (on dit aussi *bande passante).*

Ce sont plutôt des problèmes d'électronique.

2.1.2 La couche liaison de données

La couche 2 : la couche liaison de données

But de cette couche :
Transformer un moyen brut de transmission en une liaison de données qui paraît exempte d'erreur de transmission à la couche supérieure.

Caractéristiques et fonctions :
- **Unité** traitée : un paquet ;
- achemine les données reçues *de la couche supérieure en les organisant en blocs de transmission,*
- elle transfert des paquets *de source à destination* ; on parle de *trame*, de *cellule*...selon les propriétés et données contenues dans les paquets.
- correction d'erreurs, règles de partage du support, qualité de service.

La couche liaison de données

Remarque :
Jusque là il s'agit d'une liaison directe entre deux hôtes, sans changement de support physique.

Ethernet
Ensemble (matériel et logiciel) permettant de réaliser les impératifs de cette couche. Il est aujourd'hui intégré dans les cartes réseau *ethernet*. La partie matérielle d'*ethernet* permet de détecter si le support est libre ou occupé mais aussi les collisions.

Schéma algorithmique d'accès et partage du support d'*ethernet :*

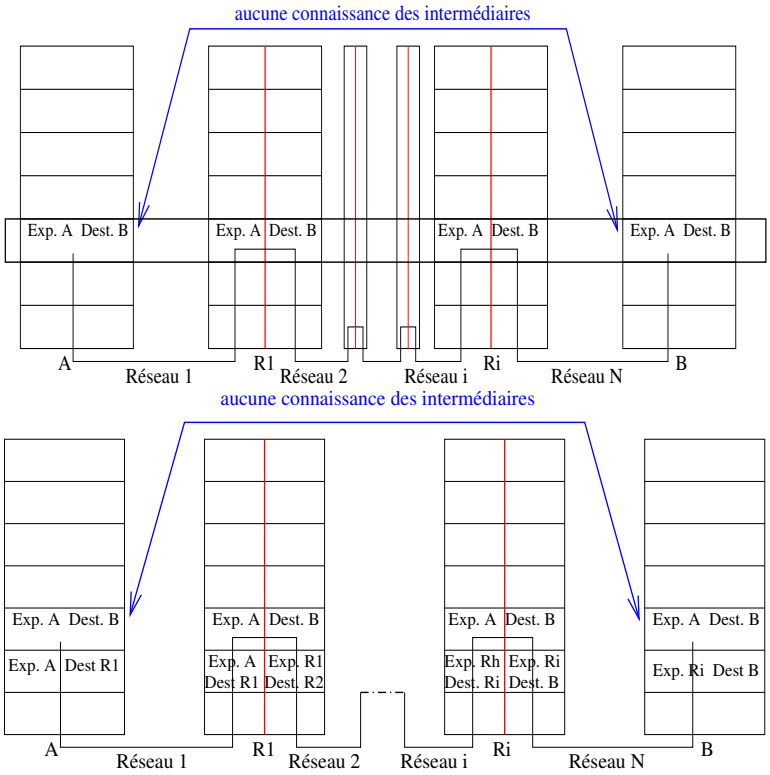
paquetExpédié = faux ;
(non paquetExpédié) (non supportLibre)attendre ;
expédier paquet ;
collision tirer délai aléatoire ;
paquetExpédié = vrai ;

2.1.3 La couche réseau

La Couche réseau

Changement important :
- Le destinataire final peut se situer dans un réseau distant, différent par la structure et la topologie de celui de l'expéditeur.
- Il faut passer par des intermédiaires (routeurs).
- Il faut interconnecter des matériels et des réseaux hétérogènes.

Problèmes d'interconnexion
Interconnexion de réseaux en vue de la transmission de bout en bout : source et destination du message.



Caractéristiques

- contrôle de flux : *baisser, augmenter la cadence en fonction de l'état des espaces tampon* ,
- routage : *trouver un chemin adéquat ou au moins le prochain nœud,*
- adressage : *quelle forme, comment passer de l'adresse réseau à l'adresse physique (adresse physique et adresse liaison sont souvent utilisés comme synonymes)*
- mode connecté (exemples : X25, certains réseaux publics) ou
- mode sans connexion (exemple : IP protocole de l'*Internet*)

Attention

La taille des paquets est $\neq$ de la taille des trames ; donc découpage possible et besoin de réassembler les morceaux.

2.1.4 La couche transport

La Couche transport

Definition 27. La couche transport est *responsable du bon acheminement des messages complets au destinataire.*

Rôle de la couche transport

Son rôle est de prendre les messages de la couche session, de les découper (s'il le faut) en unités plus petites et de les passer à la couche réseau, tout en s'assurant que les morceaux

arrivent correctement de l'autre côté. Elle effectue donc aussi le réassemblage du message à la réception des morceaux.

Les paquets vs les messages
On passe du niveau d'un paquet à celui d'une suite de paquets appelé message.

Les adresses ?
à nouveau, des adresses source et destination seront associées, internes à chaque hôte.

Optimisation grâce aux types de connexions
La couche transport sert à *optimiser les ressources du réseau.*

Elle crée une connexion réseau par connexion de transport requise par la couche session.

Elle est aussi capable de créer plusieurs connexions réseau par processus de la couche session pour répartir les données, par exemple pour améliorer le débit.

Tout est transparent pour la couche session ....

Le multiplexage
Elle est capable d'utiliser une seule connexion réseau pour transporter plusieurs messages à la fois grâce au *multiplexage.*

Mode connecté ou sans connexion
Elle permet d'être en mode connecté ou sans connexion (dépend du service offert avec ou sans garantie de délivrance, ...)

Qos
Elle gère le contrôle de flux et donc la qualité de service ; notion importante, dépendante du service rendu par les trois premières, mais difficile à exprimer.

Couche transport dans Internet

Les protocoles TCP, UDP, …dans le monde Internet.

Internet – protocoles transport

Services offerts par les protocoles de transport sous-jacents :

TCP	UDP
fiable	non
ordre garanti	non garanti
duplication impossible	possible
mode connecté	sans connexion
orienté flot	orienté message

Signification

Fiable : retourne un résultat à l'application, éventuellement négatif !

Ordre garanti : s'il y a désordre dans l'arrivée des paquets, le protocole prend en charge la remise en ordre et l'application ne s'en aperçoit pas.

Duplication impossible : s'il y a eu une double réception, le protocole la traite et l'application ne s'en aperçoit pas.

Signification - ça se complique

Mode connecté : la boîte réseau est utilisée pour communiquer de façon exclusive avec une seule autre boîte réseau ; on parle alors de *circuit virtuel* établi entre les deux applications ; analogie : le téléphone (mode connecté) et le courrier postal (mode sans connexion).

Orienté flot : le contenu expédié est vu comme un flot ; il peut être reçu en plusieurs morceaux ; de même, plusieurs expéditions peuvent être délivrées en une seule réception. m lectures $\leftrightarrow$ n écritures, $m \neq n$.

Orienté message : un message est expédié comme un bloc et reçu entièrement (ou non reçu si le protocole n'est pas fiable) ; vu de l'application, il n'est pas découpé. 1 lecture $\leftrightarrow$ 1 écriture.

2.1.5 La couche Session

La Couche Session

La couche 5 : la couche session

Elle organise et *synchronise les échanges entre tâches distantes.*

Adresses ?

Elle réalise le lien entre les adresses *logiques et les adresses physiques* des tâches réparties

Gestion de jetons

Elle établit également une liaison entre deux programmes d'application devant coopérer et commande leur dialogue (qui doit parler, qui parle...).

Points de reprise

Elle permet aussi d'insérer des points de reprise dans le flot de données de manière à pouvoir reprendre le dialogue après une panne.

2.1.6 La couche présentation

6 – Couche Présentation

Rôle de la couche présentation

Elle s'intéresse à *la syntaxe et la sémantique des données* . Elle peut convertir, reformater, crypter compresser les donnée.

	octet1	octet2	octet3	octet4
gros boutiste	poids fort 2^{31} ... 2^{24}	2^{23} ... 2^{16}	2^{15} ... 2^{8}	poids faible 2^{7} ... 2^{0}
petit boutiste	poids faible 2^{7} ... 2^{0}	2^{15} ... 2^{8}	2^{23} ... 2^{16}	poids fort 2^{31} ... 2^{24}

Mais il n'y a pas que des entiers à transmettre...

— ASCII
— Unicode

2.1.7 La couche application

La couche Application

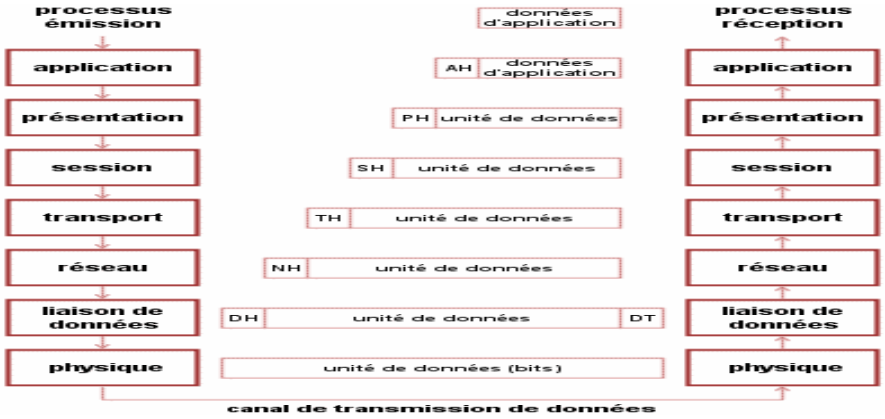
Son rôle :
C' est le point de contact entre l'utilisateur et le réseau.
Elle va apporter à l'utilisateur les services de base offerts par le réseau.

Ce n'est pas un fourre-tout pour autant. Penser aux
— protocoles de messagerie (acheminement, transcription des adresses),
— protocoles de la toile : http, https,
— protocoles de transfert de fichiers (multi-fichier, compression) :ftp ...
— codage des images (type de codage, compression),
— synchronisation d'horloges...

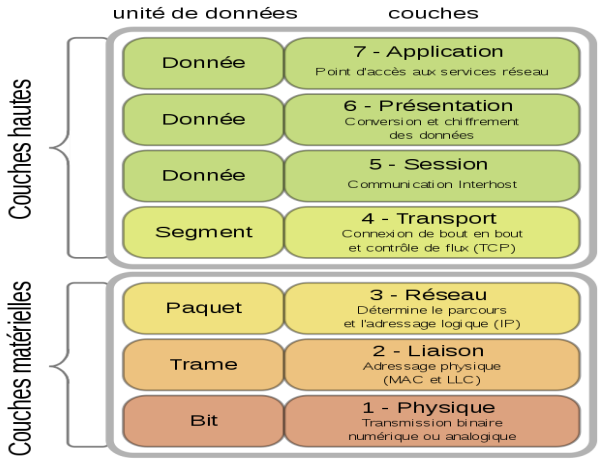
En quelque sorte, le début des problèmes lorsque la partie transmission sur les réseaux *fonctionne* (?)
Prévoir un protocole par nouvelle application.

2.2 Du passage des données entre les couches

Les transmissions de données



Les transmissions de données

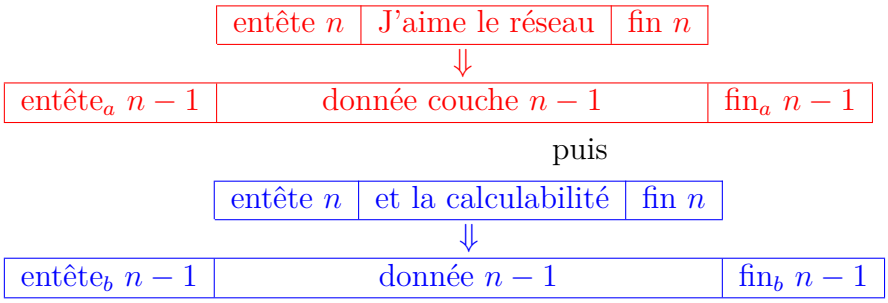


Encapsulation et Découpage

L'encapsulation

Pour tenir compte de ses propres caractéristiques, chaque couche peut être amenée, lors de l'encapsulation à découper ce paquet en tranches et transmettre alors chaque tranche dans

un paquet.



Exemple d'encapsulation

Encapsulation Ethernet
Un paquet de la couche réseau, dans le protocole IP encapsulé dans un paquet ethernet :

				ent. IP	donnée IP	
entête eth				donnée eth		CRC
preamb 64 bits	D eth 48 bits	S eth 48 bits	type 16 bits	368 à 12000 bits		32 bits

longueur paquet IP $\leqslant$ 65536 octets
64 octets $\leqslant$ longueur totale trame ethernet $\leqslant$ 1518 octets

ATM
Pour citer au moins une autre technologie : une trame ATM est de longueur fixe, 53 octets, dont 48 pour la donnée et 5 pour l'entête.

Désencapsulation

Désencapsulation

Chaque couche est amenée lors de la **réception** à :
- détecter une éventuelle anomalie en recalculant le code de contrôle,
- décapsuler le paquet : enlever entête et contrôle et transmettre au voisin.

Remarque
Le code de contrôle n'est pas systématiquement présent dans toutes les couches ; il peut aussi être effectué sur une partie du paquet seulement et être incorporé dans l'entête.

Question 1

Est-ce que le découpage à l'encapsulation peut intervenir à une position quelconque dans le paquet découpé ?

La réponse est oui (petit arrondi possible), mais la justification ?

Question 2

Lorsqu'il y a eu découpage, qui (quelle couche, sur quel hôte) doit faire le réassemblage ?

Réponse vaste à garder au chaud.

Interconnexion de réseaux – la base

Couche/Matériel

niveau	outil
physique	répéteur, concentrateur, *hub*
liaison	pont, commutateur, *switch*
réseau	routeur, *router*
plus haut	passerelle de …, *gateway*

— Il existe des produits intermédiaires : pont–routeur, …
— à chaque niveau, la machine réalisant l'interconnexion est capable de traiter le paquet correspondant (sauf pour le niveau physique). Elle reconnaît et peut séparer tous les éléments de l'entête ou contrôler la validité du paquet.
— Le problème : performance ⇒ machines spécialisées.

3 Chapitre 3 :Des noms et des adresses

3.1 Caractérisitiques d'Internet

Caractéristiques d'Internet

- Un ensemble de réseaux physiques disparates, interconnectés
- utilisant un ensemble de protocoles commun, regroupés dans l'appellation *TCP/IP*
- services connus : messagerie, transfert de fichiers, connexion à distance, serveurs de noms, partage de ressources.
- propositions et normes de facto : RFC (Request For Comment)

Les RFC sont des documents de référence, parfois bien lisibles, parfois non, avec un index riche, contenant un marquage d'obsolescence. **Recommandation** : Consulter au moins les plus connus, protocoles de messagerie, ceux de la toile, des protocoles communs de l'Internet.

3.2 Présentation du problème

Identifier pour s'y retrouver ...

Le problème des adresses
On a un ensemble de machines qui ont des identifiants (différents ou pas) et on veut qu'elle communique entre elles. Il faut donc identifier ces machines intelligemment par rapport au réseau (en général) et par rapport à sa carte réseau en particulier.

De l'ordre avec le nom de domaine
Nommage hiérarchisé par domaines : *on a un domaine racine (=domaine de premier niveau) , puis un sous-domaines (= de second niveau) , etc, jusqu'à l'hôte.*

Représentation
nom-hôte **.** sous-dom **.** …. domaine **.** dom-racine
Exemple : courses.carrefour.fr

3.3 Un nom, c'est bien, une adresse, c'est mieux ....

Il faut nommer les choses

Problème
Les noms sont un bon moyen pour désigner les hôtes et un très mauvais moyen pour acheminer des paquets.

On va associer une adresse aux hôtes : un **entier**. Dans la version 4 du protocole IP (version en cours) c'est un entier de 32 bits. Cette version s'appelle IPV4.

Adresse/Paquets
Cet entier sera l'adresse de l'hôte et figurera ainsi dans tous les paquets de la couche IP.

3.4 Historique : l'adressage par classes

Adressage par classes sur IP

Historique

Dans la préhistoire de l'Internet (1980-1990), les adresses étaient attribuées par classe, selon l'importance du réseau à administrer. Retour aux classes de services grâce à IPV6. - représente un bit affecté pour l'adresse du *réseau* ; x représente un bit affecté pour l'adresse de l'*hôte*.

classe	octet1	octet2	octet3	octet4
A	0--- ----	xxxx xxxx	xxxx xxxx	xxxx xxxx
B	10-- ----	---- ----	xxxx xxxx	xxxx xxxx
C	110- ----	---- ----	---- ----	xxxx xxxx
D	1110	multiadressage		
E	1111	futur !!!!!!		

Adressage par classes

Mais aujourd'hui : adressage sans classes sur IPV4. On y remédie en utilisant les masques le sous-adressage et le sur-adressage.

Exercice

Combien de réseaux respectivement de classe A,B,C sont possibles dans ce monde ?
Combien d'hôtes sont possible dans chacun des cas ?
Calculer les bornes dans chaque cas.

Adresses spécifiques

Adresse du réseau

Lorsque la partie allouée à l'hôte est toute à zéro, cette adresse le réseau. Elle sert dans les algorithmes de routage (voir ci-après).

Adresse tous

Tous dans un réseau : lorsque la partie allouée à l'hôte est toute à 1 binaire, cette adresse désigne **tous** les hôtes du réseau. Elle sert lorsqu'on veut expédier un paquet à l'ensemble des hôtes.

Adresse du réseau

- 198.211.18.47 désigne un hôte déterminé,
- 198.211.18.0 désigne l'adresse du réseau et cette adresse n'est utilisée que dans l'algorithme de routage ; elle ne figurera jamais dans un paquet,
- 198.211.18.255 désigne **tous** les hôtes du réseau ci-dessus ; elle peut figurer dans un paquet.

Remarque :
on verra que les adresses *réseau* et *tous* n'ont pas forcément les suffixes respectifs 0 et 255.

3.5 Adressage CIDR

Adressage CIDR

Classless Inter-Domain Routing
L'adressage sans classes=adressage CIDR est un système de gestion et d'allocation d'adresses IP le plus utilisé aujourd'hui. Ce système a été conçu pour remplacer l'adressage par classes (RFC 1518 et 1519).

Definition 28. Une adresse IP avec CIDR ressemble à une adresse IP normale, à la différence près qu'elle se termine par une barre oblique suivie d'un nombre, appelé préfixe du réseau IP.

Les adresses CIDR réduisent la taille des tables de routage et rendent plus d'adresses IP disponibles au sein des organisations.

3.6 Et les adresse Ips, c'est quoi ?

Protocole IP

Definition 29. On appelle **datagramme** *un paquet vu de la couche réseau.*

Ce que garantit *IP* :
- *Acheminement de datagrammes sans connexion ;*
- décision selon l'adresse réseau du destinataire ;
- décision *à chaque datagramme indépendamment du passé ;*
- *redécoupage possible ;*
- boucles possibles ;
- acheminement **au mieux** (*best effort*), *donc pas de garantie de livraison : un paquet peut être perdu, supprimé…*

Format du Paquet IP

octet 1	octet 2	octet 3	octet4
Vers. | lg. ent.	type service	lg. paquet	
Identification		drapeaux	place frag.
dur e vie	proto. suiv.	contr le ent te	
adresse IP source			
adresse IP destination			
options …			
…			bourrage
Donn es …			

L'entête classique, sans options, fait 20 octets.

3.7 Des adresses IP pour le routage

Routage - Généralités

Le routage
Les routeurs font du routage selon l'adresse du **réseau** destinataire (et non l'hôte destinataire).

Les algorithmes
Algorithmes plus ou moins sophistiqués (parcours dans un graphe dynamique).

Acheminement
Tous les hôtes impliqués dans l'acheminement doivent résoudre le problème du routage pour chaque paquet à expédier : à qui envoyer ce paquet ? Ici, *tous les hôtes impliqués* sont l'hôte expéditeur et tous les routeurs intermédiaires, jusqu'au dernier routeur localisé sur le même réseau que l'hôte destinataire.

Remarques :
- Ne pas oublier que chaque acheminement hors du réseau local implique un acheminement local (le routeur local).
- La commande `netstat -r` permet de visualiser la *table de routage*.
- Cette table peut être statique ou dynamique (cf. chapitre Routage).

Table de Routage - Exemple

Tables de routage
Un hôte dans un réseau local de technologie ethernet, avec un seul routeur vers le monde extérieur aura une table de routage de cette forme :

Destination	Contact	Interface
201.202.203.0	direct	eth0
autre	201.202.203.1	eth0

Contact ≡ Passerelle dans l'affichage des tables.

Réseau local
Pour contacter tout hôte du réseau local, pas besoin d'un intermédiaire ; on envoie les paquets directement au destinataire, en les expédiant sur la carte réseau dont l'adresse est $eth0$.

Autre réseau
Pour contacter tout autre hôte, expédier le paquet vers la machine dont l'adresse réseau est 201.202.203.1, toujours par la carte réseau d'adresse $eth0$.

Des Problèmes en perspective

Les problèmes
La table de routage donne l'adresse réseau du contact. Or l'adresse du destinataire dans le paquet de la couche réseau **doit** être celle du destinataire final, de bout en bout !

Il faudra disposer de l'adresse physique du routeur, c'est-à-dire l'adresse physique correspondant à l'adresse réseau 201.202.203.1.

Même dans le cas d'un contact direct, on ne dispose que de l'adresse réseau du destinataire.

Les solutions
Dans tous les cas, à la fin de l'algorithme de routage, on obtient comme résultat une adresse réseau (routeur ou destinataire final).

Il faudra obtenir l'adresse physique si on veut acheminer ce paquet.

3.8 Les adresses physiques

Adresses MAC

Definition 30. Une adresse MAC est un identifiant *de 12 chiffres hexadécimaux attribué à chaque carte réseau : c'est une adresse physique*

Par convention on place des : tous les 2 chiffres

EXEMPLE :
12 :34 :56 :78 :91 :BC

Dans un même sous réseau, la communication entre machines est possible avec des adresse MAC.

Les adresses MAC ne sont pas routables, on utilise donc les adresses IPs.

3.9 Les serveurs de noms : du nom aux adresses

Serveurs de Noms

C'est une des premières applications importantes quoique invisible dans les réseaux.

Le problème
Connaissant le nom d'un hôte, trouver son adresse. Elle est indispensable si l'on veut construire un paquet qui lui est destiné.

Principe
Une base de données distribuée, où chaque administrateur mettra à jour les données relatives à son réseau. Il mettra en place une application appelée *serveur de noms* (DNS) qui répondra à chaque requête contenant un nom, par l'adresse correspondante.

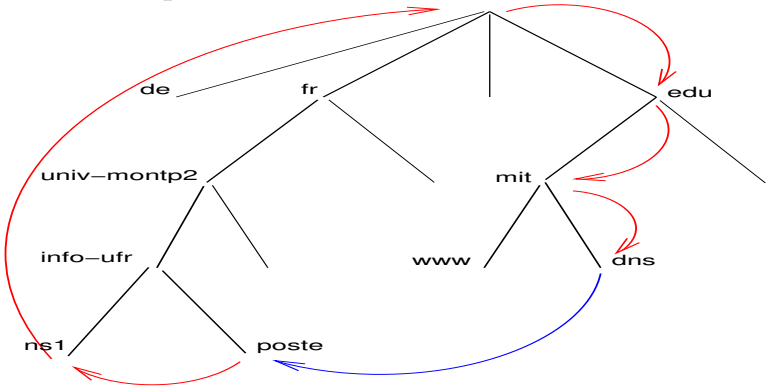
Algorithme de recherche

Généralisation du problème
Connaissant une caractéristique d'un hôte, trouver toutes les informations enregistrées à son sujet.

Algorithme de recherche
Basée sur une recherche en arbre plutôt originale.

Principe de l'algorithme
Un hôte demande à son serveur de noms local les coordonnées globales pour un nom d'hôte. Pour toute réponse concernant un hôte non local, le serveur doit disposer d'au moins une adresse d'un serveur extérieur qui donnera la réponse ou pourra faire suivre la requête.

Serveurs de noms - Exemple



Solution simple
Disposer d'au moins une adresse d'un serveur racine.

Exemple simple
Un utilisateur navigant sur `g12@info-ufr.univ-montp2.fr` veut contacter `www.mit.edu`.

L'application locale va adresser la requête au serveur de noms local (`ns1` sur la figure).

`ns1` ne dispose que de la base locale (tous les noms et adresses des machines que l'administrateur veut rendre visibles à l'extérieur), et d'une adresse d'un serveur racine.

Recherche en arbre
Le serveur racine connaît uniquement les serveurs de noms de premier niveau. Dans cet exemple, celui du domaine edu. Celui-ci connaît à son tour les serveurs de noms des domaines d'un niveau en dessous et ainsi de suite.

Chemin inverse
Le serveur du domaine `mit` répondra, sans avoir à suivre le chemin inverse. **Question** : pourquoi ?

L'adresse source est dans le paquet. .

4 Chapitre 4 – Configuration de Réseaux et Adressage Sans Classes

4.1 Le Besoin

Problème

On peut imaginer un réseau de Classe C sans répartition en sous-réseaux, sans trop de difficultés. Quoique, si l'on en a besoin, serait-ce possible à réaliser ?

Il est absurde de construire un réseau de classe B ou (pire) A sans le répartir en sous-réseaux.

Mais est-ce que la seule solution serait de répartir un réseau de classe A en sous-réseaux de classe B et un (sous-)réseau de classe B en sous-réseaux de classe C ?

On devrait pouvoir plutôt adapter l'organisation du réseau aux services demandés.

Organisation d'un réseau

Partager un réseau en sous-réseaux permet :

- de faire correspondre l'organisation du réseau avec l'organisation administrative en services :
  - les personnes d'un même service S_0 ont besoin de correspondre entre eux plus souvent qu'avec d'autres services (est-ce vrai ?) ;
  - ils ont alors besoin de ***leur*** sous-réseau ;
  - bien sûr, ceci ne doit pas empêcher le communications entre différents services, donc entre les sous-réseaux.
- d'améliorer le fonctionnement global du réseau :
  - lorsque tous les hôtes d'un réseau sont sur une seule liaison physique, alors toute communication entre deux hôtes bloque la ressource réseau globale (pas de parallélisme possible) ;
  - la séparation en sous-réseaux permettra de n'affecter qu'un sous-réseau lorsque deux hôtes d'un même sous-réseau communiquent entre eux ; le parallélisme devient possible : deux hôtes H_1 et H_2 peuvent communiquer sur leur sous-réseau SR_1 sans perturber la communication entre H_3 et H_4 sur SR_2.

4.2 Retour sur l'Adressage

Principe de l'Adresse Réseau

Une adresse réseau est de la forme :

partie réseau	partie hôte

La partie *hôte* est à disposition de l'administrateur local. Qui peut en profiter pour créer des sous-réseaux.

réseau	sous-réseau	hôte

La longueur attribuée à la partie *sous-réseau* va déterminer le nombre de sous-réseaux possibles et par conséquent le nombre d'hôtes dans ce sous-réseau.

Exemple de Partage

Deux bits de sous-réseaux permettent de configurer au plus 4 sous-réseaux, avec 64 hôtes au plus par sous-réseau, sans oublier que deux adresses d'hôte sont réservées : celle désignant *le réseau* (adresse hôte entière à 0 binaire) et celle désignant *tous* (adresse hôte entière à 1 binaire).

Supposons que l'adresse 192.36.125.0 ait été attribuée à une institution. Si l'administrateur doit faire 4 sous-réseaux, on aura la répartition suivante, écrite volontairement en binaire :

réseau	sous-réseau	hôte
11000000 00100100 01111101	00	000000 à 111111
11000000 00100100 01111101	01	000000 à 111111
11000000 00100100 01111101	10	000000 à 111111
11000000 00100100 01111101	11	000000 à 111111

En Décimal - Surprise

On peut maintenant écrire l'ensemble de la distribution des adresses en décimal, ainsi :

SR n^o	adresse réseau	adresse tous	adresses hôtes
1	192.36.125.0	192.36.125.63	192.36.125.1 à 192.36.125.62
2	192.36.125.64	192.36.125.127	192.36.125.65 à 192.36.125.126
3	192.36.125.128	192.36.125.191	192.36.125.129 à 192.36.125.190
4	192.36.125.192	192.36.125.255	192.36.125.193 à 192.36.125.254

On voit bien ici que les suffixes 0 et 255 ne sont pas les seuls représentants des adresses *réseau* et *tous* respectivement.

Remarques et Exercices

Remarques

— En affectant 2 bits aux sous-réseaux, on pourrait aussi construire 1 sous-réseau de 128 adresses d'hôtes et 2 sous-réseaux de 64.

— Évidemment, ce n'est pas parce que le total fait 256 qu'on peut faire n'importe quelle combinaison, même si le nombre total d'hôtes est une puissance de 2.

Exercices

— Écrire la répartition des adresses relatives à la première remarque.
— Comment répartir un réseau de classe B en sous-réseaux de 256 adresses chacun (i.e. de type classe C chacun) ?
— Comment répartir un réseau de classe B en sous-réseaux de 128 adresses chacun ?

4.3 Notion de masque

Généralités sur les Masques

Definition 31. Un masque est une donnée numérique (binaire), permettant d'extraire une partie d'une donnée numérique par une opération logique.

Ici c'est un *et* logique qui sera utilisé. Cette opération est nettement plus rapide qu'une suite de décalages.

Généralités sur les Masques

Example 32. On prend un réseau de classe C, sans sous-réseaux, par exemple 192.34.38.0. Le masque 255.255.255.0 permet d'extraire l'adresse réseau à partir de l'adresse de tout hôte. Soit un hôte H d'adresse 192.34.38.212 ;

	192	34	38	212
et	255	255	255	0
s'écrit	11000000	00100010	00100110	11010100
et	11111111	11111111	11111111	00000000
résultat	11000000	00100010	00100110	00000000
soit	192	34	38	0

Remarques, Exercices

— Un masque n'est pas nécessairement constitué d'une suite consécutive de 1, suivie d'une liste de 0. Voir par exemple le masque relatif aux droits de création de fichiers pour s'en convaincre.
— En fait, dans la configuration des réseaux il est très commode d'utiliser des masques constitués d'une suite de 1 suivie d'une suite de 0, parce que les parties réseaux et sous-réseaux sont « à gauche ».

Example 33.
— Quel est le masque nécessaire pour extraire la partie réseau seule de l'adresse d'un hôte quelconque, dans un réseau de classe C qui a quatre sous-réseaux ?
— Quel est le masque nécessaire pour extraire les deux parties, réseau et sous-réseau dans ce même réseau ?

Pourquoi le Routage a Besoin de Masque ?

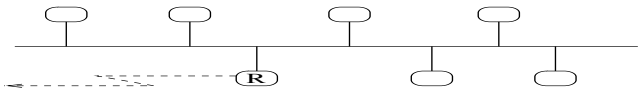
On utilise des masques dans l'algorithme de routage (cf. couche réseau) pour répondre lors du traitement d'un paquet à la

Question ?
Est-ce que le destinataire du paquet est sur le même (sous-)réseau que moi-même ?

On verra qu'en fait la question est un peu différente, mais elle se généralise facilement.

Cas d'un routage avec masque

Considérons un réseau de classe C, par exemple 192.34.38.0 sans sous-réseaux, connecté au monde extérieur par un routeur R. Le schéma suivant représente le cas d'un réseau à diffusion (par exemple, ethernet).



Étudions le routage dans ce cas.

La table de routage classique, simplifiée, d'un hôte quelconque H_0 se présente ainsi :

Destination	Contact	Interface
192.34.38.0	direct	eth0
autre (*défaut*)	192.34.38.1	eth0

où `eth0` désigne le périphérique « carte réseau » et 192.34.38.1 est l'adresse réseau du routeur. Cette table dit que :

— pour tout paquet destiné à un hôte local, H_1 par exemple, il faut expédier le paquet directement à H_1 ; ceci veut dire que la couche liaison de H_0 mettra dans l'adresse de destination l'adresse liaison (dite aussi adresse physique) de H_1 ;

— pour tout paquet destiné à un hôte **non** local, H_{ext}, il faut expédier le paquet à 192.34.38.1, ici le routeur ; ceci veut dire que la couche liaison de H_0 mettra dans l'adresse de destination l'adresse liaison du routeur.

Routage avec Masque

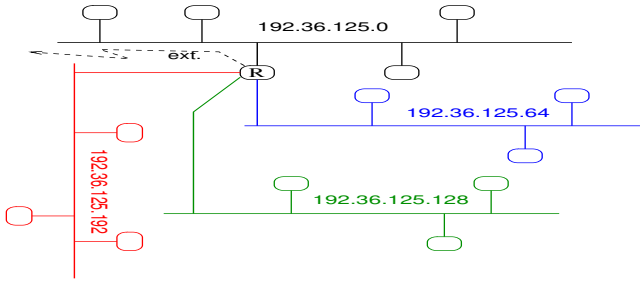
Question : Comment peut-on savoir qu'une adresse de destination fait partie du réseau local ou non ?

Réponse : en utilisant un masque appliqué aux adresses source et destination. Si le résultat est identique, alors les deux hôtes sont sur le même réseau.

Question : Quel masque faut-il appliquer pour que le routage se passe correctement dans tous les cas, quelle que soit la répartition en sous-réseaux ? C'est l'objet de la suite.

Masques de Sous-Réseaux

On reprend l'exemple du réseau 192.36.125 divisé en quatre sous-réseaux. Supposons que les quatre sous-réseaux créés SR_1, SR_2, SR_3, SR_4 soient interconnectés par un routeur R.



Problème : Un routage correct doit permettre à tout hôte d'acheminer directement un paquet destiné au même sous-réseau et de passer par le routeur pour toute autre adresse, extérieure ou appartenant à un des autre sous-réseaux. Le routeur doit pouvoir distinguer les divers sous-réseaux.

Solution

On ajoute un masque pour chaque destination dans la table de routage :

Sur un hôte quelconque, dans le sous-réseau 192.36.125.0

Destination	Contact	Masque	Interface
192.36.125.0	direct	255.255.255.192	eth0
autre (*défaut*)	192.36.125.1	???	eth0

Sur un hôte quelconque, dans le sous-réseau 192.36.125.64

Destination	Contact	Masque	Interface
192.36.125.64	direct	255.255.255.192	eth0
autre (*défaut*)	192.36.125.65	???	eth0

Remarque : On s'occupera bien plus tard des ???

Question :À quoi correspondent les adresses 192.36.125.1, 192.36.125.65 ?

Exercice : écrire la table de routage d'un hôte quelconque dans les deux autre sous-réseaux.

Table du Routeur

Destination	Contact	Masque	Interface
192.36.125.0	direct	255.255.255.192	xxx0
192.36.125.64	direct	255.255.255.192	xxx1
192.36.125.128	direct	255.255.255.192	xxx2
192.36.125.192	direct	255.255.255.192	xxx3
autre (*défaut*)	$x.y.z.t$	???	xxx4

Questions :
— À quoi correspond $x.y.z.t$?
— Que représentent les interfaces xxx1 à xxx4 ?
Exercices :
— Prendre un paquet partant d'un hôte à destination d'un autre dans un autre sous-réseau local et montrer que cette table est correcte.
— Prendre d'autres valeurs de masques, par exemple 255.255.255.0 puis 255.255.255.128 et analyser ce qui se passe (attention, cette dernière valeur est *traitre*.

Notation

On peut constater qu'une adresse IP est insuffisante pour déterminer la taille du réseau correspondant. Par exemple, 192.36.125.0 ne dit pas s'il s'agit d'un réseau découpé ou non.

Afin d'éviter toute ambiguïté, on associe aux adresses de réseau le masque correspondant, par la notation :

adresse/masque

où *masque* désigne la longueur de la chaîne de bits à 1.

Exemple : 192.36.125.0/26 désigne le réseau d'adresse 192.36.125.0 avec un masque contenant 26 bits à 1, c'est-à-dire le masque 255.255.255.192.

On pourra constater que toutes les valeurs de masque sont possibles, de /1 à /32.

4.4 Généralisations

Réseaux de Taille Intermédiaire

Le problème : Que doit faire une organisation ayant besoin d'un réseau de plus de 254 hôtes, tout en ne justifiant pas d'un réseau de classe B ?

Ce problème est d'autant plus important que la classe B est saturée et qu'il y a actuellement peu de chances d'obtenir une telle adresse.

Solution : Se faire attribuer plusieurs réseaux de classe C et jouer sur les masques et le routage afin de rendre cette attribution acceptable.

Attention : s'il s'agit de partager chacune de ces adresses en sous-réseaux, le problème est simple. Par contre, si l'on veut gérer l'ensemble des adresses comme un bloc découpé en unités de tailles diverses, il faudra obtenir des adresses de classe C ayant une partie commune maximale sur un sous-ensemble des 24 premiers bits !

Sur-adressage

On vient de voir comment découper un réseau en sous-réseaux. Mais parfois on a besoin de faire l'opération réciproque : **associer plusieurs adresses obtenues en un seul réseau**. On parle alors de *sur-réseau*.

Dans ce cas, il faudra obtenir des adresses *compatibles*, c'est-à-dire ayant une partie commune **sans trous**.

Exemples :

— 192.34.38.0 et 192.34.39.0 peuvent être associées avec un masque de 23 bits ; on dira que 192.34.38.0/23 et 192.34.39.0/23 sont compatibles.
— 192.34.38.0 et 211.56.72.0 ne sont pas compatibles : on ne pourra pas créer un réseau homogène avec ces deux adresses, tout en ayant un routage correct, à moins de créer une table de routage contenant autant de lignes que de hôtes dans le réseau.
— 192.34.38.0 et 192.34.37.0 ne sont **pas** compatibles, à moins d'avoir obtenu **aussi** 192.34.36.0 **et** 192.34.39.0 !

5 Chapitre 5- Grande Traversée des Paquets

5.1 Retour à l'Enfer des Couches

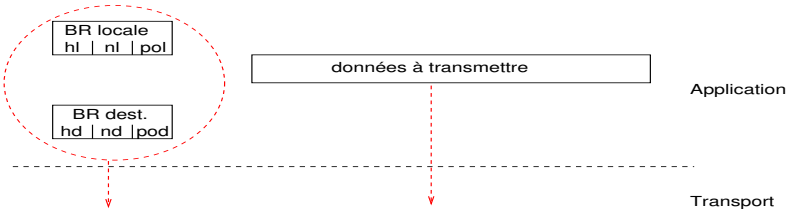
5.2 Encore un problème de couches ?

Rôle de l'application

Expédition
Lors d'une expédition, l'application expéditrice prépare et fournit à la couche en dessous (ici le transport) :

— le contenu du message (le *paquet vu par l'application*) à expédier
— les triplets des adresses des boîtes réseau *source* et *destination*.

Analyse dans l"application avant l'expédition (`send()` ou `sendto()`) : l'adresse de la BR de destination est déterminée.

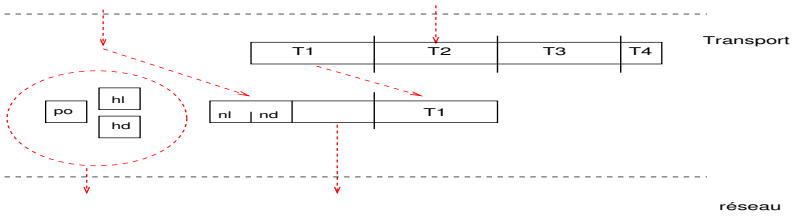


Rôle du Transport

Principe :
Chaque couche construit son paquet ; c'est ce qu'elle sait faire. Elle utilise ce qui lui est nécessaire et transmet à la suivante les éléments non utilisés jusque là.

La couche transport

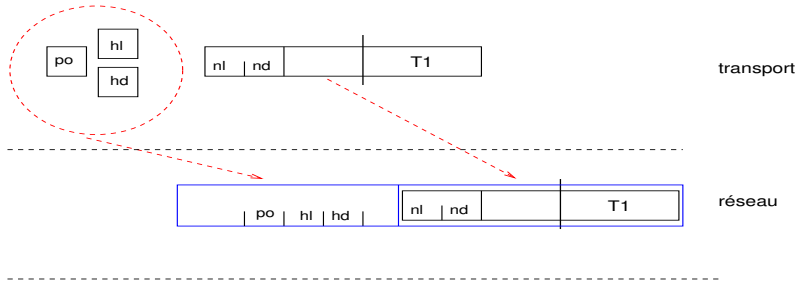
— utilise les numéros de BR inclus dans les adresses et seulement les numéros,
— découpe la données si nécessaire : déjà vu dans l'encapsulation.



Rôle de la Couche Réseau

La couche réseau

— utilise les adresses réseau (les numéros IP dans notre cas),
— redécoupe la donnée si nécessaire (penser aussi aux routeurs qui relient des réseaux de caractéristiques différentes)

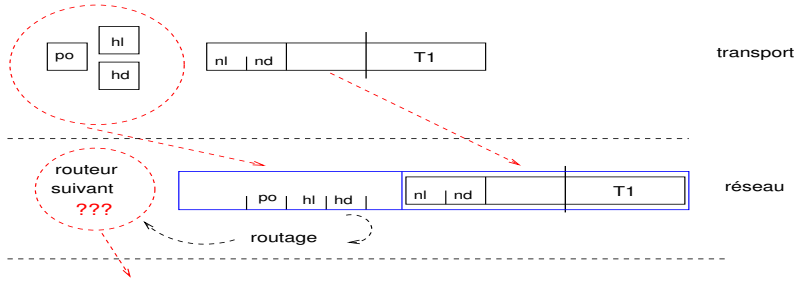


Rôle du Routage

Le paquet ?:

Le paquet de *bout en bout* est constitué, mais à qui le faire suivre ?

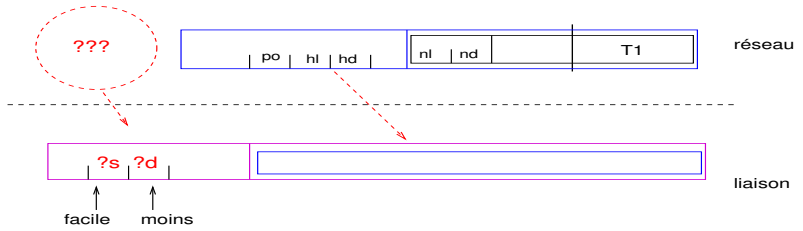
— La couche réseau résoud le problème du routage ; elle trouve donc l'adresse **réseau** du destinataire suivant.



Rôle de la Couche Liaison

La couche Liaison

Le **problème** : le voisin d'en dessous aura besoin de l'adresse du niveau liaison du destinataire local pour acheminer la donnée. Connaissant l'adresse réseau, comment obtenir l'adresse liaison ?



Solution

Si le problème ci-dessus est résolu, la couche liaison pourra utiliser son propre protocole pour acheminer le paquet au destinataire suivant.

5.3 Recherche de l'Adresse Physique

Solution Statique

Correspondance d'adresse

Une solution possible consiste à avoir une table de correspondance pour l'ensemble des hôtes du réseau local, par exemple dans un réseau local type *ethernet* :

adresse réseau	*adresse physique*
201.202.203.1	$8:0A:B2:84:7F:04$
201.202.203.2	$0:12:34:8F:EE:AA$

Problèmes
Une telle solution résout le problème, mais présente tous les défauts d'une table statique dès qu'une mise à jour doit être effectuée : toutes les machines doivent être mises à jour de façon coordonnée.

Ces mises à jour peuvent devenir fréquentes dans le cas d'affectation d'adresses de réseau dynamiquement (voir *dhcp*).

Solution Dynamique

Definition 34. La solution proposée actuellement est de construire la table précédente dynamiquement. Le protocole ***ARP*** (Address Resolution Protocol) est utilisé pour cette construction.

Principes
- Diffuser à tout le réseau local l'adresse réseau du destinataire (local) en demandant à celui qui possède cette adresse de répondre en donnant son adresse physique.
- Chaque hôte va maintenir sa propre table de correspondance dite table *ARP*, comme dans l'exemple précédent.
- Une durée de vie sera associée aux données, permettant de ne pas ignorer un hôte dont une des adresses a été modifiée. On parle de cache ARP .

Paquets ARP

Format des paquets ARP :

entête	type opération	adresse φ expéditeur
adresse réseau expéditeur	adresse φ cible	adresse réseau cible

type opération deux types sont possibles, requête (question) et réponse.

adresse φ adresse physique. Dans une requête ARP, l'adresse physique de la cible est évidemment absente.

Remarques :
- Ce même format de paquet peut être utilisé pour obtenir une adresse réseau à partir d'une adresse physique.
- La cible remplit le champ manquant, inverse expéditeur et cible, change le type de *requête* en *réponse* et renvoie le paquet.

6 Chapitre 6 – Gestion des Erreurs sur IP

Présentation du Problème

Constat : l'acheminement de datagrammes dans l'Internet se fait **au mieux**, sans garantie de livraison.

Action : Si un routeur ne peut acheminer un datagramme alors il tente d'en avertir l'hôte expéditeur.

ICMP (*Internet Control Message Protocol*)est le protocole d'annonce d'erreurs.

Il est utilisé par le logiciel de la couche réseau (IP), non seulement dans le sens *routeur* → *hôte*, mais aussi par des hôtes ou routeurs pour des utilisations *détournées* comme par exemple des tests d'accessibilité.

Remarque : noter qu'un routeur ne peut annoncer l'erreur qu'à l'hôte source (seule adresse figurant dans le paquet IP). C'est le logiciel de la couche réseau sur l'hôte source qui traite l'erreur ou la fait suivre à l'application correspondante.

Types d'Erreurs

Les exemples suivants permettent de voir l'étendue des dégâts et de constater qu'annoncer une erreur à la source n'est pas toujours la bonne solution.

Un routeur peut se trouver dans une situation désagréable comme :

— pas de chemin vers l'adresse destination dans sa table de routage,
— l'hôte de destination n'existe pas (détection par le dernier routeur),
— le réseau par lequel il veut acheminer est en panne ou congestionné,
— obligation de détruire le datagramme, par exemple, suite à une erreur du code de contrôle, ou à une durée de vie dépassée.

ICMP intègre aussi la possibilité d'obtenir diverses informations entre routeurs, entre hôtes ou les deux. Une des utilisations les plus connues est

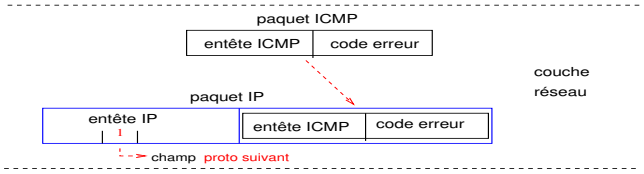
- la demande d'écho et
- la réponse associée à cette demande,

par le logiciel `ping`.

Où Traiter ?

ICMP fait partie de IP. C'est-à-dire que dans la couche *réseau* il y a du logiciel et des paquets ICMP au même titre que IP.

Les paquets ICMP sont acheminés dans des datagrammes IP. On en déduit l'encapsulation suivante :



Noter que le champ *protocole suivant* dans l'entête IP est utilisé pour désigner le *suivant*, soit dans la couche transport (tcp, udp, autre), soit ICMP, avec une valeur différente bien sûr.

Noter aussi qu'une erreur dans l'adresse source du datagramme va aboutir à la perte de l'annonce d'erreur.

Rappel du Paquet IP

Un rappel de la forme d'un paquet IP

octet 1	octet 2	octet 3	octet4
Vers. | lg. ent.	type service	lg. paquet	
Identification		drapeaux	place frag.
dur e vie	proto. suiv.	contr le ent te	
adresse IP source			
adresse IP destination			
options ...			
...			bourrage
Donn es ...			

Paquet ICMP

•L'entête de tout paquet ICMP est de la forme :

type	code	contrôle
8bits	8bits	16 bits

•Le champ *type* désigne le type d'erreur. **Exemples** :

8	demande d'écho
0	réponse écho

3	destination inaccessible
4	congestion
11	dépassement durée de vie

•Le champ *code* comporte une information complétant le type d'erreur. **Exemples** :

0	réseau inaccessible

1	hôte inaccessible
6	réseau inconnu

•Dans tous les cas d'erreur, ICMP ajoute dans la donnée les 64 premiers bits du datagramme ayant provoqué l'erreur. Plus généralement, la donnée permet de compléter plus explicitement les indications de l'entête.

6.1 Erreurs Liées au Routage

Destination Inaccessible

• Lorsqu'un routeur ne peut pas délivrer ou faire suivre un datagramme, il construit un message d'erreur ICMP, avec dans le champ *type* la valeur 3, dans le champ *code* une valeur de 0 à 12, calcule la somme de contrôle et ajoute au paquet ICMP les 64 premiers bits du datagramme, extrait l'adresse de l'hôte source Hs puis détruit ce datagramme non routable.

Ce paquet est encapsulé dans un datagramme IP, contenant en source le routeur expéditeur et en destinataire Hs, avec dans le champ *protocole suivant* le code 1, désignant ICMP.

L'hôte source peut ainsi analyser *plus sérieusement* la cause du rejet et faire suivre à l'application un retour d'erreur.

•Noter qu'un routeur peut faire suivre des datagrammes **sans se rendre compte** que la destination est inaccessible.

Exercice : Donner deux exemples démontrant ce phénomène, l'un concernant un hôte destinataire (penser à ethernet par exemple pour répondre), l'autre concernant un routeur destinataire.

Dépassement de Durée de Vie

Associer une durée de vie au datagramme IP permet de faire en sorte qu'un datagramme ne puisse circuler indéfiniment dans l'Internet sans arriver à destination.

Est-ce possible ? Oui, pour des erreurs de routage provoquant des aller-retours d'un datagramme entre deux routeurs, chacun ayant malheureusement une interprétation erronée des informations de routage, ou pire, une boucle de routage entre plusieurs routeurs (voir le chapitre sur le routage).

Solution : le champ *durée de vie* contient dans sa forme la plus simple (l'actuelle, dans IPV4), le nombre maximum de routeurs que le datagramme peut traverser. Chaque datagramme IP se voit appliquer le principe suivant :

Algorithme TTL

Appelons TTL le champ *durée de vie* du datagramme IP.

L'hôte source du datagramme initialise ce champ à une valeur déterminée, dans le logiciel de la couche réseau.

Chaque routeur applique ensuite l'algorithme suivant :

```
TTL - - ;
(TTL == 0) expédier message ICMP (dépas
détruire datagramme ;
```

Les Échos

La demande d'écho dans ICMP permet aux routeurs de savoir si les routeurs voisins sont actifs ou non. Lorsqu'un routeur reçoit un message ICMP de *demande d'écho*, il doit répondre par un message ICMP de *réponse écho*.

Cette caractéristique est utilisée non seulement entre routeurs, mais aussi entre hôtes pour tester leurs présences, comme nous l'avons déjà vu pour le logiciel `ping`.

Noter que ping visualise la valeur du champ *Durée de Vie* et affiche aussi le temps d'aller-retour du datagramme.

Exercice : Pour quelles raisons est-ce que la durée d'aller-retour du premier datagramme dans `ping` est souvent supérieure aux suivants ?

Et si ICMP Provoquait une Erreur ?

Remarque Importante : Tout paquet ICMP est encapsulé puis routé dans un datagramme IP. Dès lors, ce datagramme peut subir les mêmes avatars que tout datagramme IP, perte, congestion, abandon.

Les pertes et erreurs engendrent des pertes et des erreurs (d'après Rez O.)...

Dans leur sagesse, les concepteurs ont décidé qu'on ne devait construire un message ICMP relatif à un datagramme contenant déjà un message ICMP...

Conséquence : voici encore une raison pour laquelle des protocoles comme `TCP` doivent inclure des garanties, ajouter des délais, tenir actifs les circuits virtuels, et alerter les applications avec des moyens complémentaires.

6.2 Utilisation Détournée

Détournemesnt de TTL

Le comportement des routeurs relativement au champ *Durée de Vie*, permet d'en faire une utilisation détournée, afin de déterminer le chemin d'accès à un hôte.

La commande `traceroute` applique un algorithme dont le principe est :

HdestNonAtteint = vrai ;
TTL=0 ;
(HdestNonAtteint) TTL + -
expédier (datagramme, Hdes
(expéditeur erreur ICMP) ;
(réponse de Hdest) HdesNon
Alg

Question : Est-ce vraiment un chemin correct ?

Analyse de Traceroute

Exercice : prendre le schéma de réseau suivant et montrer que l'algorithme précédent peut afficher des chemins faux ou pire, inexistants. On suppose que S cherche un chemin vers D et que les Rx représentent des routeurs.

On peut définir
— *faux* par : le résultat donné ne sera pas un chemin suivi par un paquet,
— *inexistant* par : le chemin affiché contient au moins un arc (ou un sommet) inexistant.

6.3 Compléments Datagramme IP

Fragmentation

Un datagramme IP peut être *fragmenté*, c'est-à-dire découpé en morceaux, sur un ou même plusieurs routeurs, en fonction des caratéristiques des réseaux que le routeur interconnecte.

Chaque fragment circule comme un datagramme indépendant, donc peut suivre un chemin différent d'un autre fragment.

Conséquences :

— le réassemblage ne peut se faire que sur le hôte destinataire final,

— dans la couche IP qui doit attendre la réception de tous les fragments, tout en acceptant entre temps d'autres datagrammes,

— chaque fragment doit contenir les informations nécessaires à l'identification du datagramme d'origine et à l'insertion correcte du fragment dans ce datagramme.

7 Chapitre 7 – Routage

7.1 Introduction au Routage

Le Problème du Routage

Déterminer en fonction de l'adresse réseau du destinataire final d'un datagramme le prochain destinataire.

On peut compléter très légèrement le tableau déjà vu dans le cas d'un hôte quelconque sur un réseau local.

Destination	Contact	Masque	Interface
201.202.203.0	direct	255.255.255.192	eth0
autre	201.202.203.1	0.0.0.0	eth0

Contact indique soit le prochain routeur, soit une destination sur le même réseau.

Problèmes

— Cette table peut prendre des dimensions gigantesques dans le cas d'un routeur censé connaître l'ensemble des destinations de l'Internet ou d'un sous-ensemble.

— Comment construire cette table ?

Routage Statique

Le routage statique constitue une solution simple à la construction de la table : elle est figée et modifiée uniquement par une intervention d'un administrateur.

Cette solution est parfaitement bien adaptée à un réseau local avec un seul routeur assurant la connectivité vers le monde extérieur.

L'utilisation d'une route par défaut permet de passer rapidement le relai d'un hôte à un routeur, d'un routeur à un autre routeur. On comprend mieux pourquoi des incohérences sont possibles. Et il y a pire, par exemple des boucles...

Routage Dynamique

Dans le cas d'un routeur reliant plusieurs réseaux, un routage adaptatif ou dynamique permettra de tenir compte de :

— l'infrastructure des réseaux connectés,

— l'arrivée et la réparation de pannes, la création de nouveaux liens,

— la charge des réseaux (congestions, oscillations),

— de la qualité de service requise, etc.

Une distribution *intelligente* des adresses de réseaux permettrait de réduire la taille des tables de routage (voir routage hiérarchique). Hélas, ce n'est pas le cas, du moins ceci n'a pas été fait systématiquement, dans l'Internet.

Connaissance Partielle et Erreurs

Constat : Dans tous les cas, le résultat de l'algorithme de routage est l'adresse du *suivant*.

On espère que les routeurs sont cohérents entre eux, c'est-à-dire que le *suivant* peut continuer à acheminer correctement le paquet. Sinon, on aura des **erreurs de routage**.

Ceci reste vrai même si un routeur connaît le chemin complet, car on ne peut pas **forcer** une décision sur un **autre** routeur ; sauf cas spécifiques de tests de chemins, ce serait néfaste de le forcer.

Traitement des Erreurs - Un Début

On peut empêcher un paquet de vivre indéfiniment dans l'Internet :

Principe :

— Un champ *durée de vie (ttl)* est attaché à chaque paquet (cf. entête du paquet IP) ; il est initialisée par l'hôte source du paquet ;
— Chaque routeur décrémente la durée de vie ;
— Le routeur qui arrive à une durée de vie nulle ou négative détruit le paquet.

Actuellement, la durée de vie est mesurée en *nombre de routeurs traversés*, dit aussi *nombre de sauts*.

ttl- - ;
(ttl $>$ 0)router paquet ;
expédier (erreur routage) à hôte source ;

Classes d'Algorithmes

Plusieurs classes d'algorithmes dynamiques existent en fonction de l'étendue des réseaux reliés.

Globalement, on a besoin de trouver des chemins dans un graphe dynamique. Il faut se poser les questions de

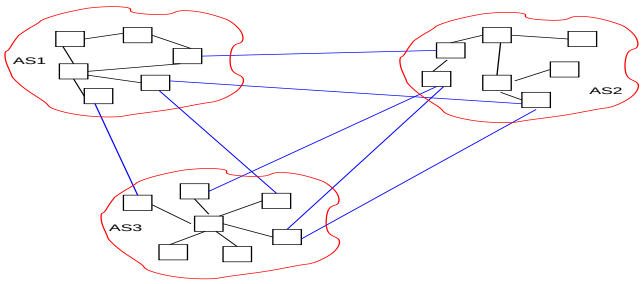
— l'efficacités des algorithmes distribués ou centralisés ;
— convergence, stabilité et cohérence des algorithmes.

Pourquoi plusieurs classes d'algorithmes ? Parce qu'on ne peut plus se contenter d'une organisation centralisée.

Organisation en Systèmes Autonomes de l'Internet

Aujourd'hui, l'Internet est organisé en *Systèmes Autonomes* (AS ci-après), vastes réseaux (en général), administrés chacun par une entité unique.

Chaque AS possède des routeurs *intérieurs* reliant les sous-réseaux entre eux, et des routeurs *extérieurs* reliés à des routeurs extérieurs d'autres AS.



Organisation du Routage dans l'Internet

Chaque AS organise son propre routage interne librement. Plusieurs algorithmes sont connus : RIP, OSPF.

Pour la partie externe, il faut un protocole commun.

Les AS communiquent entre eux par un seul algorithme lié à un seul protocole : aujourd'hui, BGP.

Les routeurs *extérieurs* d'un AS doivent connaître **toutes** les adresses des autres AS afin de constituer un routage cohérent. On insiste sur **toutes**, pas seulement celles des AS et routeurs adjacents.

On pourra voir qu'il est difficile de concilier le fonctionnement extérieur, visible, avec les choix politiques internes. C'est un sujet de recherches actuellement.

7.2 Algorithmes à Vecteurs de Distances

Algorithme à Vecteur de Distance RIP

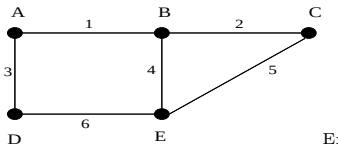
Principes :

- Diffusion d'informations sur le routage à base de la **distance** ; quelle métrique pour exprimer une distance ? le plus fréquent : *nombre de sauts* ;
- Chaque routeur dispose d'une table contenant des triplets $(destination, numéro\_de\_liaison, coût)$ c'est-à-dire pour telle destination, envoyer sur telle liaison pour tel coût ;
- La table est mise à jour dynamiquement ; il y a diffusion périodique d'informations $(destination, coût)$
- Chaque routeur qui reçoit une information la compare au contenu courant de sa table ;
- Si l'information est *meilleure* il la prend ;
- Sinon, il y a des cas où l'on est obligé d'accepter une information fut-elle *moins bonne*, d'autres où on la rejettera.

Algorithme RIP

table de routage ; des doublets $(dest, cout)$ reçus sur une liaison l_{recue} table de routage
toutes les données arrivant $dest_{recue}$ trouvée dans table $l_{recue} == l_{table} c_{table} = c_{recue} + 1$;
$c_{table} > (c_{recue} + 1) l_{table} = l_{recue}$;
$c_{table} = c_{recue} + 1$;
ajouter $(d_{recue}, l_{recue}, c_{recue} + 1)$ dans table

Vecteurs de Distances - Exemple



Extrait de *Le routage dans l'Internet.*

Initialisation ;
A diffuse $(A, 0)$ sur les liaisons 1 et 3 ;
B diffuse $(A, 1), (B, 0)$ sur 1, 2 et 4 ;
information perdue sur 4 ;
C diffuse $(C, 0), (A, 2), (B, 1)$ sur 2 et 5 ;
D diffuse $(D, 0), (A, 1)$ sur 3 et 6.

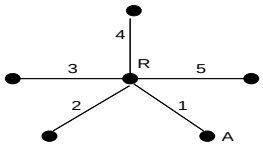
A			B			C			D			E		
→	l	c	→	l	c	→	l	c	→	l	c	→	l	c
A	loc	0	B	loc	0	C	loc	0	D	loc	0	E	loc	0
			A	1	1				A	3	1			
B	1	1				A	2	2						
						B	2	1						
			C	2	1							C	5	1
												A	5	3
												A	6	2

Justifications

Traitement global : Chaque routeur diffuse périodiquement sa table, ensemble de couples $(destination, coût)$ vers ses voisins adjacents.

Important : Si une nouvelle information sur une destination arrive par la **même** liaison que celle par laquelle on route, alors il faut la prendre, qu'elle soit bonne ou mauvaise.

Pourquoi ? Considérer le point de vue d'un routeur.



Si A annonce à R qu'une destination X est atteinte pour un coût 10 et que le même, A annonce ensuite un coût différent (envisager 7 puis 12) pour la même destination, R doit accepter ce nouveau coût, sauf si entre temps il a un meilleur chemin.

Remarques

Attention : Nous avons vu **une** simulation possible de la diffusion. L'ordre de diffusion peut être totalement différent, avec des résultats différents sur les tables.

On peut démontrer la convergence de cet algorithme si aucune modification (panne, apparition de nouvelles liaisons) n'arrive. Mais les pannes et modifications sont **fréquentes**. On peut montrer que pour toute panne ou apparition de liaison, l'algorithme converge si un nouvel incident n'a pas lieu avant l'aboutissement de la convergence.

Le défaut de fonctionnement reproché à RIP est résumé ainsi :
Les bonnes nouvelles se propagent vite, les mauvaises se propagent doucement.

Traitement d'une Panne dans RIP

Principe du traitement d'une panne : annoncer un coût ∞ pour chaque voisin en panne. Cette détection peut se faire par un outil comme `ping` permettant de tester l'existence d'un hôte.

Exemple : Supposons que la liaison 1 tombe en panne. B corrige sa table avec le triplet $(A, 1, \infty)$ et la diffuse.

Selon l'ordre de propagation de l'information, on peut constater une convergence rapide, ou des phénomènes connus sous le nom de *comtage à l'infini* :

Supposons que C diffuse sa table, **avant** que B ne diffuse la sienne. C diffuse entre autres informations $(A, 2)$, qui lors du traitement sur B va engendrer le triplet $(A, 2, 3)$!!!

D'autres défauts de fonctionnement de cet algorithme ont été répertoriés (voir bibliographie), accompagnés de solutions plus ou moins heureuses. Elles font l'objet de cours ultérieurs.

7.3 Algorithmes à états de liens-OSPF

Algorithmes à États de Liens - Principes

- Chaque routeur possède la topologie complète du réseau ;
- Deux tâches sont accomplies pour arriver à connaître cette topologie :
  - test d'activité de tous les routeurs adjacents : échanges courts (type `ping`)
  - diffusion périodique de l'état des liens : c'est un compte-rendu des communications possibles. L'état est rediffusée *autant que nécessaire* à tous les routeurs participants, avec horodatage (numéro de séquence) des messages.
- Chaque routeur calcule un plus court chemin vers chacun des autres routeurs.

Exemple : *Open Shortest Path First* (OSPF), est actuellement un algorithme à état de liaisons très utilisé à l'intérieur des systèmes autonomes de l'Internet.

Algorithmes à états de liens - Algo

ensembles $(voisins, couts, séquence)$; N : ensemble de nœuds dont le pcc est connu ;
$D(v)$: coût parcours vers v nouvelle topologie et arbre des plus courts chemins
//Initialisation N={A} ; D(v) infini (tous) ;
tous les nœuds v v adjacentD(v)=coût(A,v) ;
//mise à jour
tous les nœuds v explorés trouver w extérieur à N tel que D(w) min ;
ajouter w à N ;
tout v adjacent à w D(v)=min(D(v),D(w)+c(W,v)) ;

8 Chapitre – Mise en Œuvre d'Applications

8.1 Principes

Vision des Applications

- communication par des boites réseaux (boîtes où l'acheminement est pris en charge par *des réseaux*, comme des boîtes postales où le transport est effectué par *des postes*) ;
- directement au dessus du transport : l'interface de programmation offrira un accès au niveau transport ;
- une application peut obtenir plusieurs boîtes ;
- chaque boîte avec deux cases : *départ, arrivée* ; on dépose ce qu'on veut expédier et on lit ce qui est arrivé ;
- chaque BR a une adresse composée du triplet :
  - adresse hôte (numéro IP)
  - numéro de boîte
  - protocole

Allocation des BR

[height=12em, width=20em]prog1.pdf

Attention : chaque protocole de transport fait sa propre numérotation des boîtes. Donc un numéro de boîte n'indique pas de quel protocole il s'agit.

??? : pas de BR sans application associée.

Quel Modèle ?

Questions lancinantes : Ce modèle de programmation s'appuie sur combien de couches ? Par rapport au modèle à 7 couches de l'OSI, peut-on établir une correspondance ?

On ne pourra répondre qu'en connaissant les services offerts et en détectant les services non offerts.

8.2 Protocoles sous-jacents

Internet – protocoles transport

Services offerts par les protocoles de transport sous-jacents :

TCP	UDP
fiable	non
ordre garanti	non garanti
duplication impossible	possible
mode connecté	sans connexion
orienté flot	orienté message

Signification

Fiable : retourne un résultat à l'application, éventuellement négatif !

Ordre garanti : s'il y a désordre dans l'arrivée des paquets, le protocole prend en charge la remise en ordre et l'application ne s'en aperçoit pas.

Duplication impossible : s'il y a eu une double réception, le protocole la traite et l'application ne s'en aperçoit pas.

Signification - ça se complique

Mode connecté : la boîte réseau est utilisée pour communiquer de façon exclusive avec une seule autre boîte réseau ; on parle alors de *circuit virtuel* établi entre les deux applications ; analogie : le téléphone (mode connecté) et le courrier postal (mode sans connexion).

Orienté flot : le contenu expédié est vu comme un flot ; il peut être reçu en plusieurs morceaux ; de même, plusieurs expéditions peuvent être délivrées en une seule réception. m lectures $\leftrightarrow$ n écritures, $m \neq n$.

Orienté message : un message est expédié comme un bloc et reçu entièrement (ou non reçu si le protocole n'est pas fiable) ; vu de l'application, il n'est pas découpé. 1 lecture $\leftrightarrow$ 1 écriture.

Client – Serveur

[Client] : application qui envoie des requêtes à l'application dite *serveur*, attend une réponse indiquant leurs réalisations et les résultats éventuels. : application qui attend des requêtes provenant d'applications clientes, réalise ces requêtes et rend les résultats.

requête : suite d'instructions, commandes, ou simple chaîne de caractères, obéissant à un langage, un accord ou une structure préalables connus des deux acolytes (protocole d'appplication).

Fonctionnement des processus

Client **Serveur** Un processus "serveur" assure un service : il tourne en permanence en attendant des requêtes de clients. Il dispose d'une BR *publique*.

- Les clients arrivent à exprimer des requêtes parce qu'ils connaissent l'existence du service et l'adresse de cette BR *publique* du serveur.
- Les clients ne savent pas si le serveur est actif (ils l'espèrent actif).
- Les requêtes arrivent dans une file d'attente.
- En général le serveur doit minimiser le temps d'attente dans la file →traitement rapide ou délégation.
- **Attention à cette terminologie dans les systèmes de fenêtrage**

8.3 Interface de programmation

Interface Socket

Un ensemble de primitives permettant de réaliser des applications communiquant sur un réseau, fournissant au programmeur l'accès (l'interface) à la couche transport.

Principes :

- mode **sans connexion** : on peut communiquer dans les deux sens avec plusieurs BR.
  - se faire allouer une BR locale
  - identifier le distant
  - envoyer des messages / consulter les messages entrants
  - savoir rendre (fermer) la BR si plus nécessaire
- mode **connecté** : principes identiques, mais la communication est dédiée exclusivement à deux BR déterminées (voir *circuit virtuel*.

8.4 Programmation - Sans Connexion

Sans Connexion - Deux Modèles

On peut proposer deux modèles de programmation :

- Un modèle symétrique : chaque application désigne son acolyte, c'est-à-dire la BR qu'elle veut joindre.
- Un modèle asymétrique : une des application, A, désigne l'acolyte B. B prend connaissance de l'adresse de la BR expéditrice lors de la réception du message de A.

Sans Connexion, Symétrique

Appli a	**Appli b**	
demander BR locale BR_a	demander BR locale BR_b	*ma BR*
désigner distant(BR_b)	désigner distant(BR_a)	*BR dest.*
sendto()		*synchronisation*
recvfrom()		*par le transport*
shutdown()		
close()		

Exemple sans connexion - les BR

```
/* demande de BR locale */
Sock breLoc(SOCK_DGRAM, (short)31470,0) ;
int descbreLoc ;
/* on recupere le descripteur */
if (breLoc.good()) descbreLoc=BreLoc.getsDesc() ;
else {cout<<"pb BR locale"<<endl ;
        exit(1) ;}
/* désignation BR distante */
SockDist saBr("hote.teslunettes.fr",(short)31469) ;
sockaddr_in *adrsaBr= saBr.getAdrDist() ;
```

sockaddr_in est une structure contenant le triplet désignant une adresse de BR dans le monde Internet.

Sans Connexion - le Dialogue

```
/* on expédie */
int retourSend=sendto(descbreLoc, msg, sizeof(msg),0,(sockaddr *)adr-
saBr, lgsaBr) ;
/* et on reçoit*/
int retourRecv=recvfrom(descbreLoc, tamponReception,lgReception, 0,NULL,
NULL) ;
```

Remarque importante : il n'est pas nécessaire d'être en réception afin de recevoir le message. `recvfrom()` est bloquant : s'il n'y a pas de message, l'exécution est bloquée, s'il y en a, la réception est effectuée conformément aux paramètres indiqués.

Sans Connexion - Asymétrique

On attend une première réception, pour détecter qui est l'expéditeur.

Après cette réception, on peut récupérer l'adresse de la BR expéditrice.

On peut alors réfléchir aux fonctions d'un serveur répondant à des clients différents.

SockDist expInconnu ;
socklen_t lgInconnu=expInconnu.getsLen() ;
sockaddr_in *adrexpInconnu=expInconnu.getAdrDist() ;
int retourRecv=recvfrom(descbreLoc, tamponReception,lgReception, 0,(sockaddr *)adrexpInconnu,&lgInconnu) ;

Sans les Classes Fournies

On peut ne pas trouver les classes fournies à son goût. Se plonger alors dans les détails :

- Pour l'allocation des BR, voir les appels système
  - `socket()`
  - `bind()`
  - `gethostbyname()` et associés
  - `getservbyname()` et associés.
- Pour le dialogue, les appels sont ceux utilisés ci-avant ;
- Pour le reste, consulter les classes proposées ; attention à la syntaxe, peu encourageante.

De la Réservation des Ports

Question : Est-il logique de réserver les numéros de BR (numéros de ports) comme dans les exemples précédents ?

Réponses :

oui c'est un moyen rapide permettant de construire des exemples,

non pour plusieurs raisons :

- les numéros peuvent être utilisés par d'autres applications,
- pire, ils peuvent être utilisés par des applications courantes, dites *bien connues*,
- comment connaître les numéros alloués à distance ?
- sans parler des *plantages* entraînant des délais d'attente lors de la mise au point des programmes (voir les erreurs liées à `bind()`.

Réservation des Ports dans l'Internet

Principes :

- Un client peut demander l'attribution d'une BR sans se soucier du numéro ; une allocation par le système d'un numéro quelconque, libre est une bonne solution ;
- Seuls les serveurs ont nécessairement besoin d'être indentifiés ; ils identifient les clients lors de la première réception ;
- Dans le monde Internet, chaque application connue (donc le serveur correspondant) va se voir attribuer un numéro **publique** connu de tous les hôtes.

Exemples de Réservation

- tous les serveurs `sshd` vont utiliser strictement le port numéro 22 ;
- tous les serveurs `httpd` vont utiliser *par défaut* le port numéro 80.

Ainsi, tous les clients pourront localiser les serveurs, dès lors que ces numéros réservés sont enregistrés dans un fichier local. Noter que le contenu est universel, du moins pour les applications publiquement connues (voir sous Unix `/etc/services`).

Les numéros jusqu'à 1024 sont officiellement réservés pour ce type d'applications et ne peuvent être demandés par une application d'utilisateur non administrateur (`root`).

8.5 Programmation - Mode Connecté

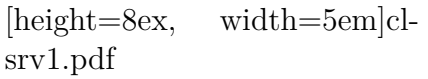
Mode Connecté - Principes de Fonctionnement

- **avec connexion** :

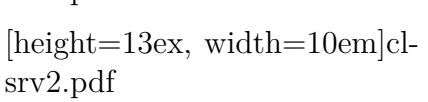
Serveur	**Client**
se faire allouer une BR publique	demander une BR privée
	identifier le serveur
attendre des requêtes	
	demander une connexion
répondre (accepter de)	attendre réponse
dialoguer	
savoir arrêter	savoir accepter l'arrêt
ou	ou
savoir accepter l'arrêt	savoir arrêter

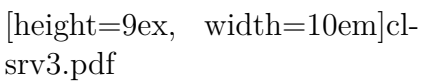
Connecté – Schéma de principe

Le serveur crée une file d'attente pour recevoir les demandes de connexions sur une BR publique



les clients font une demande de connexion à partir d'une BR privée vers la BR publique du serveur





le serveur accepte une connexion ; il y a création d'un circuit virtuel avec deux BR privées dédiées à cette communication

Protocole TCP un retour

- protocole orienté connexion avec création d'un circuit virtuel
- bidirectionnel
- fiable : arrivée garantie, pas de duplication
- ordre des paquets garanti
- message vu comme un flot de caractères : pour une écriture on peut avoir besoin de plusieurs lectures et réciproquement.

Conséquences :

- `Tcp` prend en charge la mise en place du circuit virtuel, donc en fait tout la négociation et les contrôles (accusés réception, flux, …) afin d'assurer le bon fonctionnement du circuit ;
- les processus communiquant doivent se mettre d'accord sur les limites des messages.

Mode Connecté - Mise en Œuvre

Serveur	**Client**	
créer BR publique	...	*adresse connue publiquement*
	créer BR privée	*adresse indéterminée*
listen()	...	*longueur file d'attente*
	connect()	*demande de connexion*
accept()		*acceptation d'une demande*
write() ou send()		
read() ou recv()		*dialogue*
shutdown()		*fin partielle*
close()		*fin*

Exemple - - Côté Serveur

Préparation de la BR publique

- On suppose que la BR demandée par le serveur est publiquement connue. Voir la discussion sur l'allocation des ports.

```
Sock brPub(SOCK_STREAM, (short)(21345), 0) ;
  int descBrPub ;
  if (brPub.good()) descBrPub=brPub.getsDesc() ;
  int estin=listen(descBrPub,5) ;//longueur file        //se mettre en attente
  struct sockaddr_in brCv ;
  socklen_t lgbrCv = sizeof (struct sockaddr_in) ;
  int descBrCv = accept (descBrPub,(struct sockaddr *)&brCv, &lgbrCv) ;
```

Attention : `desBrCv` est un nouveau descripteur, sur la BR privée allouée pour le circuit virtuel.

Exemple - - Côté Client

Préparation de la BR privée et demande de connexion

- Le client peut se faire attribuer une BR quelconque ; il suffit qu'elle ait un type adapté à une telle communication.

```
Sock brCli(SOCK_STREAM, 0) ;
  int descBrCli ;
  if (brCli.good()) descBrCli=brCli.getsDesc() ;
      //désigner le serveur
  SockDist brPub(argv[1], short(21345)) ;
  struct sockaddr_in * adrBrPub=brPub.getAdrDist() ;
  int lgAdrBrPub=sizeof(struct sockaddr_in) ;
      //demander une connexion
  int erlude = connect(descBrCli,(struct sockaddr *)adrBrPub,lgAdrBrPub) ;
```

Attention : le retour de `connect` indique si la requête a été déposée dans la BR publique du serveur.

Syntaxe - - Dialogue

Côté Serveur

```
char lemagne[256] ;
char lot[]="doremifa solasido" ;
int ensif = recv (descBrCv,lemagne,sizeof(lemagne), 0) ;
...
int ox = send (descBrCv,lot,strlen(lot),0) ;
```

Côté Client : Comme pour le serveur, en utilisant la boîte réseau privée locale.

Noter qu'il n'y a plus besoin de spécifier le destinataire : `tcp` a bien fait son travail.

Noter la possibilité d'utiliser `read()` et `write()` à la place de `recv()` et `send()`. Néanmoins, on perd le dernier argument, qui permet de spécifier plus finement les entrées-sorties.

8.6 À Titre Documentaire

À Titre Documentaire - Détails Primitives
**Obtenir un descripteur pour un objet <
>**
`int socket (int famille, int type, int protocole )`

famille : `PF_UNIX, PF_INET, PF_ISO, PF_INET6`

type : `SOCK_DGRAM, SOCK_STREAM, SOCK_RAW`

protocole : 0 par défaut le plus souvent ; voir manuel `protocols` et fichier `/etc/protocols`

retour : descripteur ou -1 (et errno positionné)

À Titre Documentaire - Détails Primitives
Associer un descripteur et une BR déterminée

`int bind (int descripteur, const struct sockaddr *brDem, socklen_t lgDem)`

descripteur provient de socket() ;

brDem doit être initialisée au triplet de la BR dont on demande l'allocation ;

lgDem longueur du triplet désigné ; par exemple `sizeof(struct sockaddr_in)`

retour 0 (OK) ; -1 (+errno positionnée).

Fermeture :
Comme pour les fichiers `int close (int descripteur)` ou `int shutdown (int descripteur, int comment )`

comment `SHUT_RD` arrêt réceptions

`SHUT_WR` arrêt émissions

`SHUT_RDWR` les deux

À Titre Documentaire - Détails Primitives

Dialogue sans connexion :

```
int sendto( int descripteur,         const void *msg, size_t lg,
int flags,         const struct sockaddr *brDest, socklen_t lgDest)
```

msg message à expédier ;

lg longueur du message ;

flags options ;

brDest adresse BR destinatrice ;

lgDest longueur de l'adresse BR dest.

À Titre Documentaire - Détails Primitives

Dialogue sans connexion - encore :

```
int recvfrom (int descripteur,         const void *tamponrec,size_t
lg, int flags,         const struct sockaddr *brExp, socklen_t *lgExp)
```

tamponrec tampon pour le message reçu ;

lg longueur de ce tampon (max. à recevoir) ;

flags options ;

brExp adresse de la BR expéditrice ;

lgExp longueur de l'adresse BR exp.

À Titre Documentaire - Détails Primitives

Connecté - création du CV :

```
int accept (int descripteur,          struct sockaddr *brCv, socklen_t
*lgbrCv)
```

descripteur celui de la BR publique ;

brCv nouvelle BR privée créée ; le CV côté serveur ;

lgbrCv longueur ; attention : paramètre en entrée et résultat ; à réinitialiser avant chaque accept().

retour descripteur sur la BR privée ; -1 (erreur).

file d'attente :

```
int listen (int descripteur, int lgmax)
```

retour 0(OK) ; -1 (erreur).

À Titre Documentaire - Détails Primitives

Connecté - demande de connexion :

```
int connect( int descripteur,          struct sockaddr *brSrv,
socklen_t lgbrSrv)
```

descripteur obtenu par `socket()` ; celui de la BR privée locale ;

brSrv BR publique du serveur ;

lgbrSrv longueur de cette BR serveur ;

retour 0 (OK) ; -1 (erreur).

À Titre Documentaire - Détails Primitives

Dialogue mode connecté :

```
int send (int descripteur, const void *tampon,          size_t
lg, int flags)
```

descripteur côté serveur c'est celui rendu par accept() ;

retour nombre d'octets envoyés ; -1 (erreur).

```
int recv (int descripteur,void *tampon,          size_t lg, int
flags)
```

lg max à recevoir.

retour nombre d'octets reçus ; -1 (erreur).

8.7 Du Blocage des Entrées-Sorties

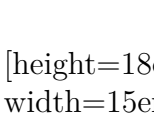
Constat : les entrées-sorties décrites jusque là sont majoritairement bloquantes :

- Les réceptions sont bloquantes de façon visible : si une BR est vide, il y aura attente jusqu'à l'arrivée d'un message ;
- les expéditions le sont aussi, bien que ce soit moins visible ; penser que le tampon d'expédition peut se vider à un rythme lent par rapport au remplissage ;
- les acceptations de connexions le sont de façon évidente ;
- les demandes de connexions le sont aussi, bien que de façon moins visible.

Problème : le schéma établi jusque là en mode connecté n'est valide que lorsqu'il y a un seul client, ou lorsque le traitement d'un client est court, de sorte à ne pas faire patienter la longue file d'attente possible.

Situation Difficile

La situation suivante est pratiquement invivable.

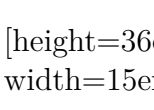


Coté serveur, si on attend une réception sur une des BR et si on n'a pas de réponse, les autres clients patientent lamentablement, quel que soit le point d'attente

Solutions :

- faire des entrées-sorties non bloquantes ; dans la plupart des cas, elles seront d'une inefficacité admirable ;
- déléguer chaque circuit virtuel à un clône ;
- autre ?

Délégation

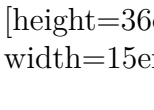


S délègue à Sbis le travail avec C1.

Sbis ferme sa copie de la file d'attente.

S ferme sa copie du circuit virtuel avec C1.

Délégation - il en reste



S délègue à Ster le travail avec C2.

Ster ferme sa copie de la file d'attente.

S ferme sa copie du circuit virtuel avec C2.

8.8 Types de serveurs

Types de serveurs

Itératif

- ne traite qu'une seule demande de connexion à la fois,
- concevable si les requêtes sont (très) courtes et/ou indépendantes les unes des autres,

- en mode conecté l'établissement de la connexion peut devenir le goulet d'étranglement.

Concurrent

- autant de serveurs que de demandes de connexion,
- multiplication du serveur de base (clônes),
- simple et efficace, mais peut devenir encombrant si les requêtes sont courtes et/ou indépendantes et /ou nombreuses.

Exercice : Citer un exemple positif et négatif dans chaque cas.
Plus difficile : citer un exemple d'application dans lequel aucun des deux cas n'est adapté.

9 Chapitre 4– Du Blocage des Entrées-Sorties

9.1 Problèmes, Principes et Solutions

Le Besoin

Problème : La délégation permet de résoudre indirectement une situation de blocage. Mais si on se retrouve avec plusieurs boîtes réseau dans un processus, la question est toujours : comment ne pas rester bloqué, ou comment savoir si un message est disponible dans une BR, sans rentrer dans une situation bloquante.

Deux solutions

1. Faire des entrées-sorties non bloquantes, c'est-à-dire modifier le comportement des entrées-sorties en mode non bloquant. Voir à ce sujet l'appel *fcntl()* pour les fichiers et même *ioctl()* pour les périphériques en général.

   Principe : associer le mode non-bloquant à un descripteur. Toute entrée-sortie reçoit alors un résultat.

   Défaut de ce principe : souvent on fait une boucle sur l'entrée-sortie jusqu'à réception d'un résultat positif, boucle appelée *attente active*.

Solutions - suite

1. Déléguer au système la responsabilité de réveiller le processus lorsqu'un événement est disponible dans un ensemble de descripteurs.

   Principe : donner au système la liste des descripteurs à scruter ; ajouter un délai maximal si nécessaire ; demander au système de réveiller le processus dès qu'un événement est disponible ou que le délai est dépassé.

   Avantage : pas d'attente active, donc pas de mobilisation inutile de ressources, avec un petit inconvénient : rester bloqué (...) jusqu'au réveil...

9.2 Déblocage

Déblocage par Réveil

Un processus peut demander au système de se faire réveiller si :

- un événement a eu lieu sur une BR en réception,
- un événement a eu lieu sur une BR en expédition,
- une exception est intervenue sur une BR,

- ou si un délai maximal a été dépassé, même si aucun événement n'a eu lieu. Ce délai est extensible : immédiatement jusqu'à infini.

Exemple :
me réveiller s'il y a quelque chose à consommer (lire) dans la BR_1, ou si la BR_2 est enfin prête à une expédition, et au pire dans 5 secondes et 234 μsecondes.

Multiplexage

On parle de *multiplexage* des entrées-sorties pour un fonctionnement de ce type, permettant d'attendre plusieurs événements simultanément.

Un appel système, *select()*, permet de réaliser cette attente. Sa syntaxe :

```
 int select( int nbsurv,fd_set *desc_en_lect,          fd_set
*desc_en_ecr,fd_set *desc_en_excp,          struct timeval *taïmeout
)
```

Avec la signification suivante :

nbsurv	→	limite sup. de descripteurs à surveiller
desc_en_lect	→	ensemble à surveiller en lecture
desc_en_ecr	→	ensemble à surveiller en écriture
desc_en_excp	→	ensemble à surveiller en exception
taïmeout	→	intervalle max à attendre

Attention au retour : nombre de descripteurs ou 0 (temps) ou -1 (erreur)

Petite Description des Structures

Qu'est-ce qu'un `fd_set` ?

Un ensemble (tableau) de booléens où l'indice est le numéro de descripteur, la valeur (*vrai/faux*) de chaque élément indiquant s'il faut ou non prendre en compte ce descripteur dans la scrutation.

Exemple :

0	1	2	3	4	5	6	7	8	9	…
f	f	f	v	f	v	f	f	f	f	…

indique qu'on veut scruter les descripteurs 3 et 5 afin d'annoncer au processus demandeur si un événement est disponible, c'est-à-dire s'il y a quelque chose à lire, écrire, ou en exception, selon la demande faite par le processus.

Mise en Œuvre

La mise en œuvre consiste à :

- declarer les ensembles,
- positionner les descripteurs : mettre à *vrai* les cases correspondantes,

- lancer la scrutation avec *select()*,
- lors du réveil tester le résultat et agir en conséquence.

Attention : le résultat indique le nombre de descripteurs à traiter ou la cause du réveil (délai, erreur), mais le traitement reste à faire !

Un ensemble de macro-définitions permet de faire des opérations sur des ensembles `fd_set`.

FD_ZERO(fd_set *ens_desc) ; →initialiser à faux un ensemble
FD_SET(int desc, fd_set *ens_desc) ; →positionner à vrai
FD_CLR(int desc, fd_set *ens_desc) ; →repositionner à faux
FD_ISSET(int desc, fd_set *ens_desc) ;→test de l'état

Exemple

```
//obtenir les descripteurs des BR et fichiers concernés ... //déclaration
fd_set detennis ;  ...boucle de traitement //indispensable de réinitialiser
FD_ZERO(&detennis) ; FD_SET(descBravo, &detennis) ; //idem pour les autres
descripteurs, de fichiers ou BR nbsurv= .... ; int erstice=select(nbsurv,
&detennis, NULL,NULL,NULL) ; //Debout, il y a eu déblocage ; il s'est
passé quelque chose //selon résultat erstice : si 0 délai dépassé ;  //
si positif où (quel descripteur) ? if FD_ISSET(descBravo, &detennis){
recv(descBravo,.....) ; //traitement réception....} //et ainsi de suite
pour tous les descripteurs ...fin boucle traitement
```

Compléments Exemple

Comment déterminer un délai ?

En affectant des valeurs dans la structure :

```
struct timeval { long tv_sec ; /* secondes */ long tv_usec ; /* microsecondes
*/} ;
```

Utilisation :

```
struct timeval attenteMax ;  attenteMax.tv_sec = (long) (5) ; attenteMax.tv_usec
= (long) (234) ; int erstice=select(nbsurv, &detennis,NULL,NULL,&attenteMax) ;
//si (erstice==0) {traiter dépassement attenteMax} ;
```

Enfin, *nbsurv* est *max(tous les descripteurs à scruter) + 1* . C'est la rançon du système fatigué de compter.

Exercices

1. Quel est l'état général des tableaux de descripteurs (ensembles `fd_set` déclarés) au retour de select ?
2. Que doit-on attendre comme résultat de *select()* si parmi les descripteurs positionnés en lecture on met celui associé à `stdin` ?

3. **Attention** : il faut distinguer les deux cas

   - délai avec une valeur nulle (nombre de secondes et microsecondes nul),
   - délai infini, donc avec un pointeur valant `NULL`.

   Analyser le premier cas et donner les résultats possibles de *select()*.

Le Grand Écouteur : Inetd

Rôle de portier pour plusieurs services ; les plus classiques : transferts de fichiers et connexions à distance.

On parle de *serveur multiport*.

1. crée un point de communication (*BR publique*) pour le compte de chaque service ;
2. demande le réveil (`select()`) sur l'ensemble des descripteurs créés ;
3. le réveil est provoqué par une demande de connexion d'un client sur l'un des ports <<surveillés>> ;
4. Dès qu'un descripteur est prêt en lecture alors génération du processus assurant le service correspondant : clône et recouvrement (`fork()` et `exec()`).

Schéma Serveur Multiport

[height=20ex, width=25em]inetd.pdf

- Après la génération du serveur correspondant, on a un fonctionnement classique, sans avoir à regénérer un clône.
- Noter qu'un serveur multiport peut écouter pour des services en tous modes (connecté ou non).
- **Question** : Comment peut-on savoir si un serveur est lié à une génération par inetd ? Plusieurs réponses possibles.
- **Plus difficile** : Citer un exemple où ce fonctionnement est inapproprié et même néfaste.

pgfexternal@did@a@shipout