



"On the Internet, nobody knows you're a dog."

Table des matières

1	Abstraction du réseau : les «couches»	5
	La notion de protocole	6
	La pile TCP/IP	13
2	La programmation Socket ou la programmation de la couche 4	34
	Notion de port: multiplexage et identification d'un processus	35
	Le protocole TCP : connexion et échanges	38
	Les «Sockets» Berkeley utilisées dans TCP/IP	40
	Les «Sockets» Berkeley utilisées dans TCP/IP : version Python	46
3	Fondamentaux – réseaux diffusion et point-à-point	49
	Virtualisation et apprentissage avec Linux	55
	Le réseau TCP/IP	65
	Adressage des matériels : Adresse IPv4 et Adresse MAC	66
	Routage direct & indirect, encapsulation	74
	Le DNS, « <i>Domain Name Server</i> » : Principe de délégation	87
	Configuration manuelle d'une machine pour l'accès à Internet	95



Le réseau : différents points de vue

- * **vision programmeur** : un système de « couches OSI, *Open Systems Interconnection* » :
 - ◊ de la couche **physique**, n°1, à la couche **applicative** n°7 ;
 - ◊ des **protocoles de communications** à maîtriser : TCP, UDP, SMTP, POP, SSH etc. ;
- * **vision humaine et géographique** :
 - ◊ des échanges **au sein de la même structure** « humaine » (entreprise, université, etc) :
 - * matériels administrés par la **même autorité** : segmentation du réseau, DNS « Domain Name System » local, etc.
 - * demande de **performances** de haut niveau : QoS, partage d'accès à des ressources, applications distribuées, etc.
 - ◊ des échanges entre **réseaux répartis** sur la planète :
 - * de **l'organisation** : organisme internationaux, DNS, etc.
 - * des **réseaux interconnectés** : INTERconnected NETworks : du routage de haut niveau avec BGP, etc.
- * **vision théorique** :
 - ◊ réseau « point à point » vs « diffusion » ;
 - ◊ des **mesures** : latence, gigue, débit, bande passante, etc.
 - ◊ des **principes** : contrôle de flux, contrôle de congestion, contrôle d'erreur, etc.
- * **vision sécurité** :
 - ◊ **surveillance** du réseau : détection et identification du trafic, protection contre les attaques ;
 - ◊ **filtrage** des échanges ;
 - ◊ mise en place de **tunnels** d'échanges sécurisés ;
- * **vision concepts fondamentaux** :
 - ◊ adressage : niveau 2, IPv4, IPv6, VLAN, MPLS ;
 - ◊ *switching* ou commutation : sélectionner une sortie pour un paquet en entrée ;
 - ◊ *forwarding* ou relayage : sélectionner *intelligemment* une sortie pour un paquet en entrée ;
 - ◊ *routing* ou routage : construire une route pour l'acheminement d'un paquet.



Éléments importants

- ▷ Notion de couches et de processus pairs ;
- ▷ Notion de service et d'interface ;
- ▷ Modèle OSI et pile de protocoles TCP/IP ;
- ▷ Modélisation de protocole ;
- ▷ Problème de synchronisation et de programmation.



Organisation en série de couches

But réduire la complexité de conception.

Les réseaux sont organisés en série de couches ou niveaux, chacune étant construite sur la précédente.

Rôle d'une couche offrir certains services aux couches plus hautes, en leur masquant l'implémentation de ces services.

Relation entre couches sur différentes machines

La couche n d'une machine gère **la conversation** avec la couche n d'une autre machine.

Notion de «protocole»

Les **règles** et **conventions** utilisées pour cette conversation sont connues sous le nom de «protocole de la couche n » :

- ▷ un **protocole** est un accord entre les parties sur la **façon** de communiquer ;
- ▷ toute **Violation** du protocole rend la communication extrêmement difficile voire **impossible**.

Notion de «processus pairs»

Les couches correspondantes sur différentes machines sont appelés **processus pairs**, *peer*.

Ce sont les processus pairs qui **communiquent** à l'aide du protocole.

En réalité, aucune donnée ne passe directement de la couche n d'une machine à la couche n d'une autre machine, mais chaque couche passe par les données et les contrôle à la couche qui lui est immédiatement inférieure.

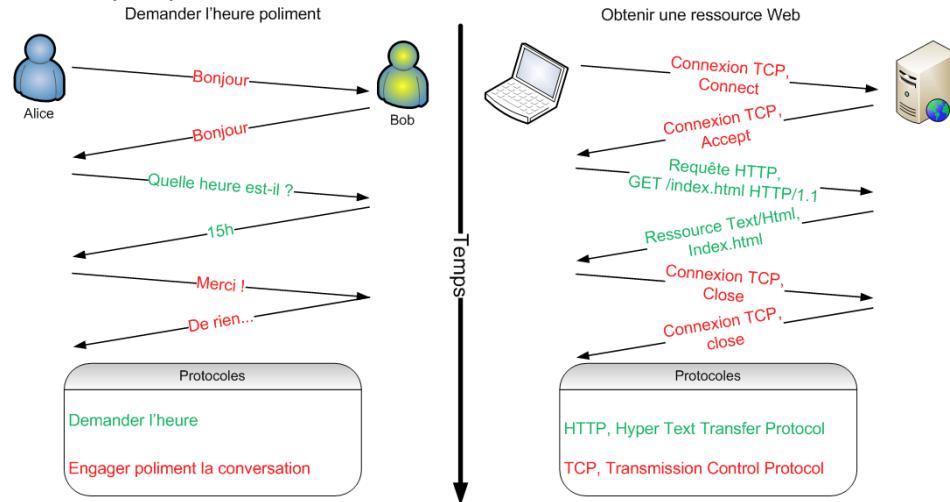


La notion de protocole

6

Un protocole humain et un protocole machine

« demander l'heure à quelqu'un » et « demander une ressource sur un serveur Web ».



Les protocoles définissent :

- * le **format** des données échangées ;
- * l'**ordre** des messages émis et reçus entre les entités réseaux;
- * ainsi que les **réactions** à ces messages.

Un protocole correspond à un **comportement** qui **évolue** en fonction des données échangées.

Exemple de protocole

7

Le protocole SMTP, «Simple Mail Transfer Protocol»

```
xterm
bonnefoi@msi:~$ socat - tcp:smtp.unilim.fr:25
220 smtp.unilim.fr ESMTP Sendmail 8.13.1/8.13.1; Thu, 15 Sep 2011 15:28:24 +0200
HELO msi.unilim.fr
250 smtp.unilim.fr Hello www.msi.unilim.fr [164.81.60.6], pleased to meet you
MAIL FROM: <bonnefoi@unilim.fr>
250 2.1.0 <bonnefoi@unilim.fr>... Sender ok
RCPT TO: <bonnefoi@unilim.fr>
250 2.1.5 <bonnefoi@unilim.fr>... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Subject: Message

Message de test envoyé directement !

.
250 2.0.0 p8FDSOJe031646 Message accepted for delivery
QUIT
221 2.0.0 smtp.unilim.fr closing connection
bonnefoi@msi:~$
```

La commande «socat» permet de simplement établir une connexion TCP avec le serveur que l'on a désigné sur le port indiqué.



Message de test envoyé directement !



Exemple de protocole

Le protocole HTTP, «*Hyper Text Transfer Protocol*»

```
□ — xterm —
pef@darkstar-8:/Users/pef socat - tcp:www.unilim.fr:80
HEAD / HTTP/1.0

HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Mon, 11 Sep 2017 10:53:33 GMT
Content-Type: text/html
Content-Length: 178
Connection: close
Location: http://www.cryptis.fr/
```

Le protocole POP, «*Post Office Protocol*»

```
□ — xterm —
bonnefoi@msi:~$ socat - tcp:pop.unilim.fr:110
+OK courriel Cyrus POP3 v2.2.13-Debian-2.2.13-14.xm.1 server ready <299345444.1316380363@courriel>
USER bonnefoi
+OK Name is a valid mailbox
PASS bob
-ERR [AUTH] Invalid login
^C
bonnefoi@msi:~$
```

Ce protocole est utilisé pour consulter son courrier et le récupérer dans son logiciel de messagerie.

On lui préfère le protocole IMAP, port 143 en version non sécurisée, qui permet de consulter son courrier tout en le laissant sur le serveur.



Notion d'interface :

entre chaque couche adjacente existe une interface.

L'interface définit :

- * les **opérations élémentaires**, appelées « primitives » ;
- * les **services** que la couche inférieure offre à la couche supérieure.

La définition des interfaces doit être **claire** :

- chaque couche réalise un ensemble de fonctions bien définies ;
- le changement d'implémentation d'une couche est transparent : *il suffit à la nouvelle implémentation d'offrir exactement à sa voisine du dessus le même ensemble de services que l'ancienne*

Exemple : changement d'une carte réseau, passage d'une connexion filaire à du sans-fil...

L'ensemble des couches et protocoles est appelé **architecture réseau**.

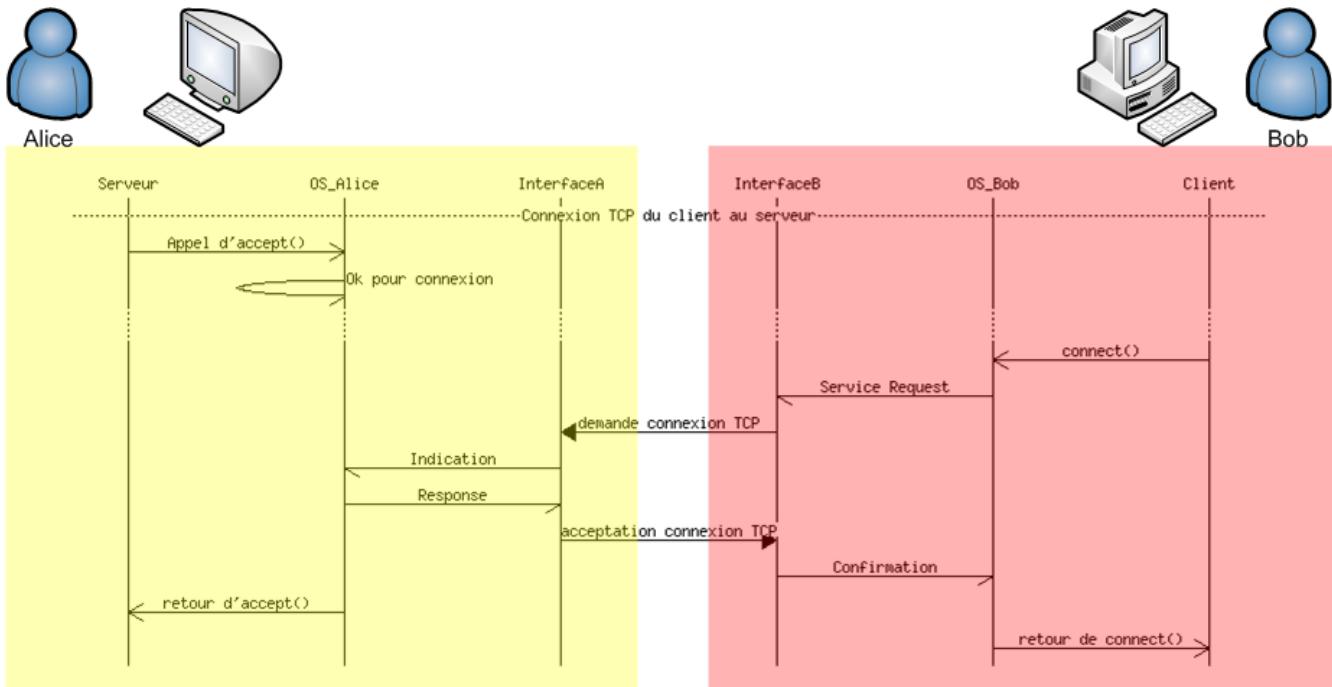
Au niveau de la conception du réseau :

- ▷ les **spécifications** de l'architecture doivent contenir suffisamment d'information pour permettre d'écrire le logiciel ou de construire le matériel pour chaque couche de manière qu'il obéisse correctement au protocole approprié ;
- ▷ ni les détails de mise en œuvre, ni les spécifications de l'**interface** ne font partie de l'architecture puisqu'ils sont invisibles à l'extérieur.
- ▷ il n'est **pas nécessaire** que les interfaces de toutes les machines du réseau soient identiques, pourvu que chaque machine puisse utiliser correctement les protocoles (windows et GNU/Linux par exemple).

L'**ensemble des protocoles** utilisés par un système, avec un protocole par couche, est appelé souvent «**pile de protocoles**».



- Le serveur indique qu'il est prêt et d'accord pour établir une connexion à l'aide de l'instruction **accept**;
- le client demande l'accès au service de connexion au travers d'une instruction **connect**:
 - ce connect réalise la primitive de service : requête ;
 - des paquets sont échangés à travers les interfaces;



- la réception de ces paquets sur le serveur déclenche une indication sur l'OS d'Alice qui possède déjà l'autorisation d'accepter la connexion, etc.



Deux concepts importants

- a. Relation entre communication virtuelle et effective
- b. Différence entre protocoles et interfaces

Exemple

les processus pairs de la couche 5 conçoivent leur communication de façon horizontale, grâce au protocole fournis par la couche 4 :

- * chacun des processus utilise :
 - une procédure appelée «envoi à l'autre côté»,
 - une procédure «réception de l'autre côté»,
- * **en réalité** : ces procédures communiquent avec les couches inférieures par l'intermédiaire de l'interface 3/4, *et non, comme elles en donnent l'impression, avec l'autre côté !*

Le **concept abstrait de processus pair** est crucial pour la conception des réseaux.

Cette technique d'abstraction permet de passer :

- ▷ d'un **problème insoluble** : la conception d'un réseau global,
- ▷ à **plusieurs problèmes solubles** : la conception de chaque couche.

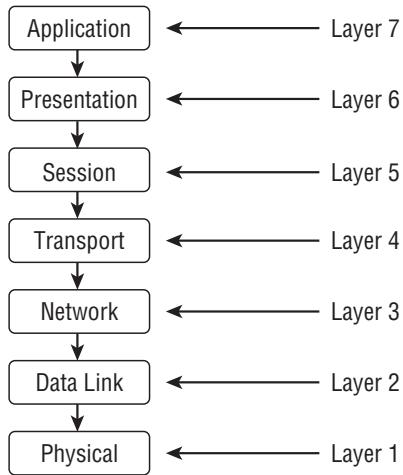
L'ensemble des couches a été normalisées par l'ISO en 7 couches.

Ce modèle a été appelé OSI (Open System Interconnection).

Il existe aussi celui, plus simple, de la **pile TCP/IP**.

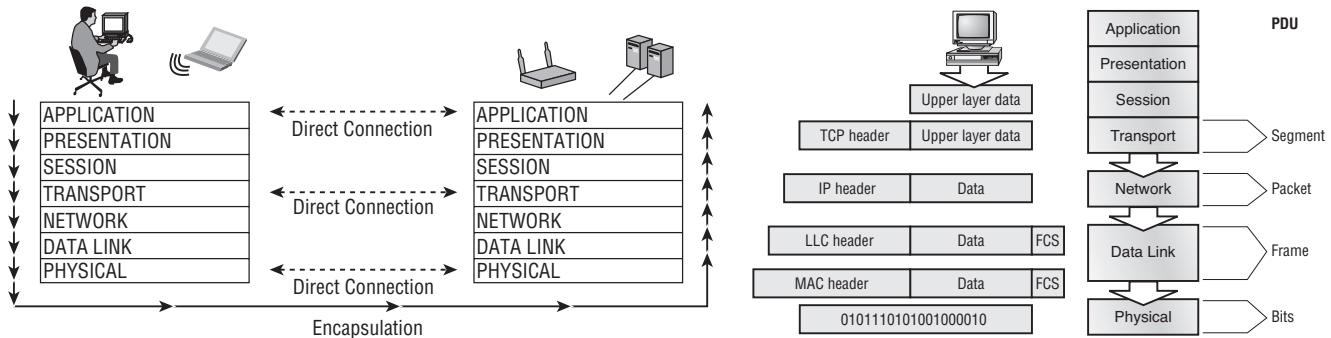
Seul la pile TCP/IP a été implémentée.

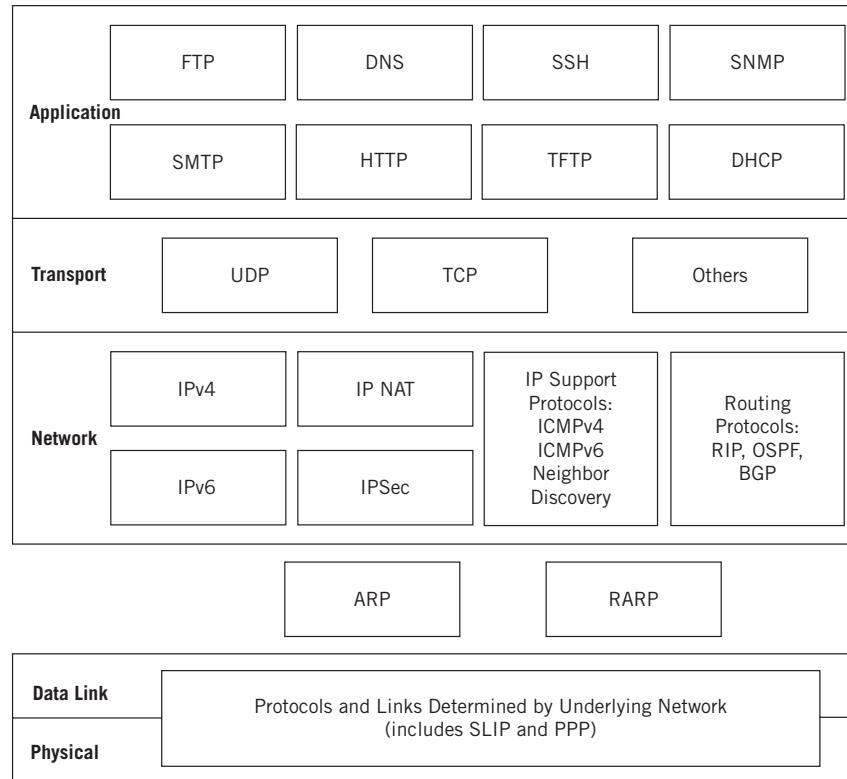




- * sert de référence pour décrire le fonctionnement du réseau :
 - ◊ 5 couches présentes dans TCP/IP : Physique, Liaison de données, Réseau, Transport, Application ;
 - ◊ 2 couches «applicatives» dans TCP/IP : Session, Présentation ;
- * chaque couche réalise un travail distinct avec des «moyens» interchangeables (protocoles, technologies) :
 - ◊ **Physique** : technologie WiFi, Ethernet, Bluetooth, etc.
 - ◊ **Liaison de données** : trame 802.3/Ethernet II ;
 - ◊ **Réseau** : datagramme IPv4, IPv6 ;
 - ◊ **Transport** : UDP, «User Datagram Protocol», TCP, «Transmission Control Protocol», SCTP, «Stream Control Transmission Protocol», etc.
 - ◊ **Application** : navigateur Web, logiciel de messagerie, etc.
- * **Présentation** : encodage des données, représentation ;
- * **Session** : ouverture/fermeture de session, identification des utilisateurs...

Communication «Pair à Pair» et encapsulation

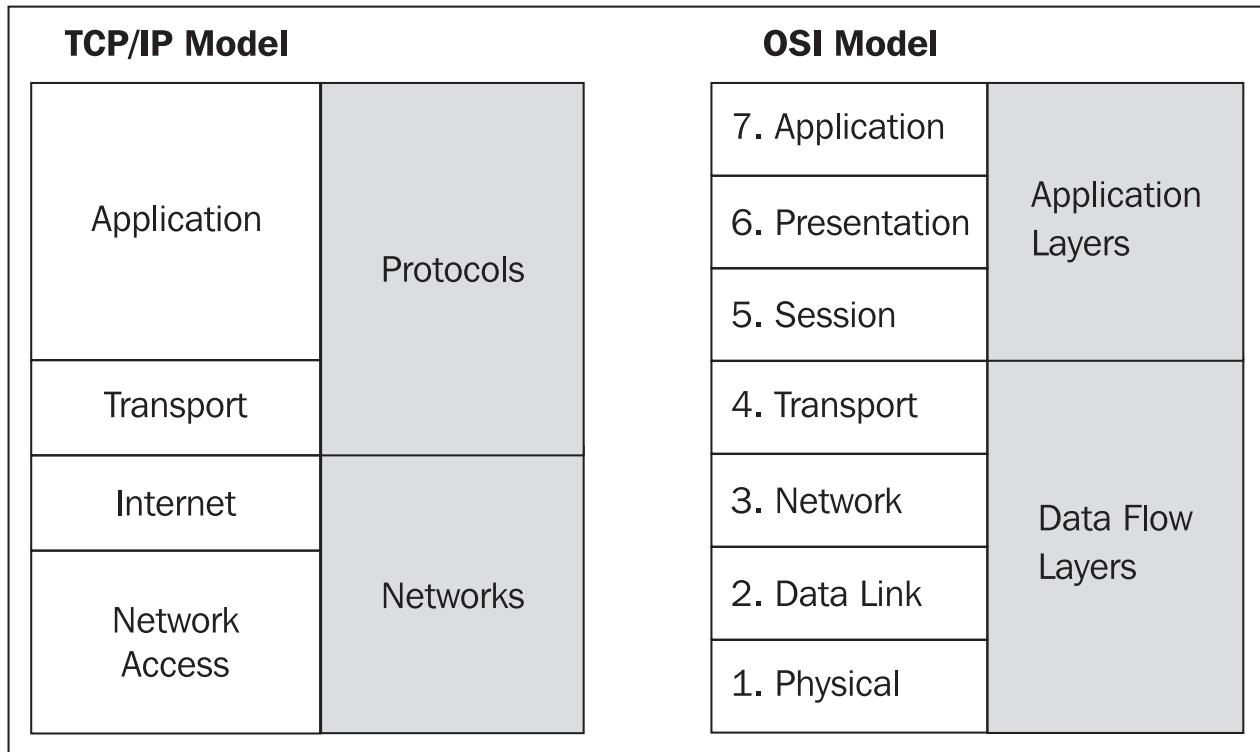




Attention :

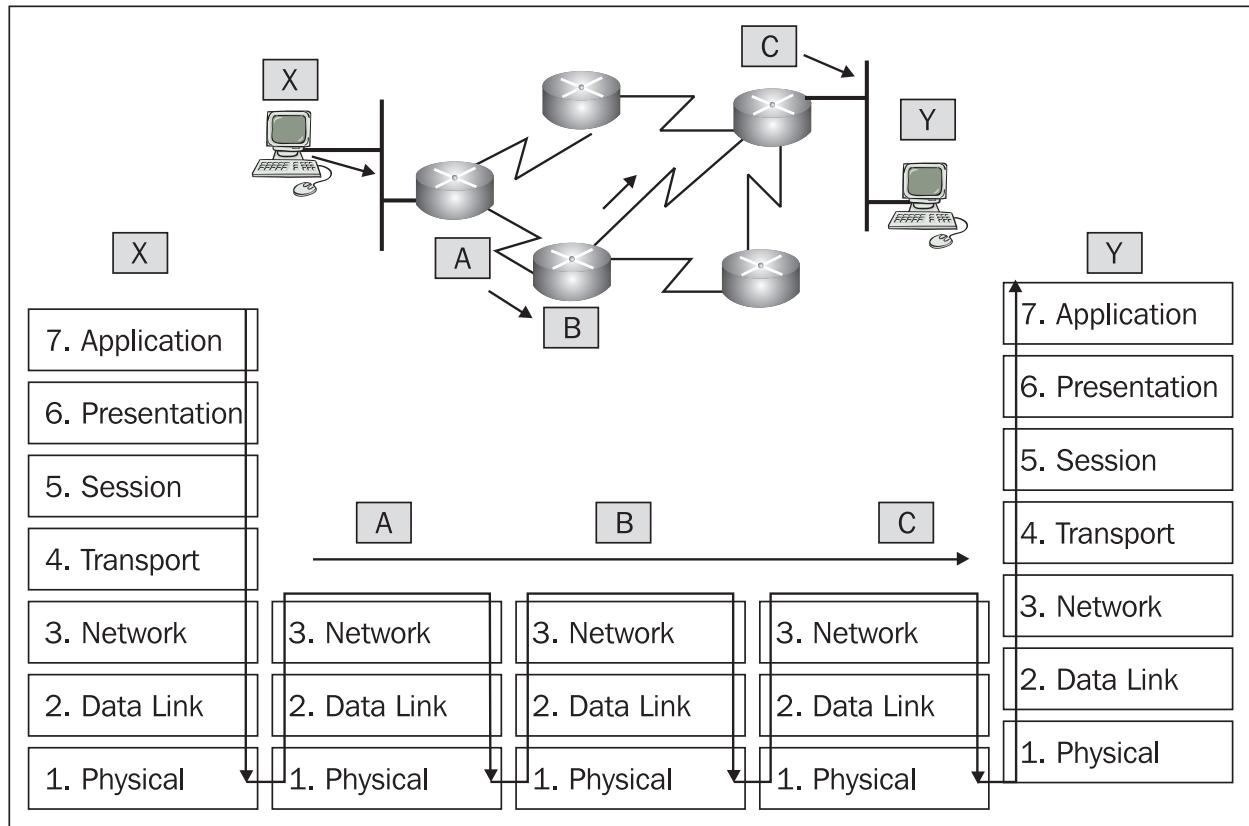
- * certains protocoles comme RIP, OSPF, BGP utilisent des protocoles de niveau 4, «Transport».
- * le protocole ARP est entre les couches 2 et 3.





Pas de couche «Présentation» et de couche «session» dans TCP/IP par rapport à OSI.





Chaque matériel dispose d'une pile TCP/IP, plus ou moins complète.

Protocoles de haut niveau

Les protocoles dans le monde TCP/IP sont décrits dans les RFC «*Request For Comment*».

RFC 1945 pour HTTP, RFC 821 pour SMTP et 1032 pour DNS, etc.

Un protocole est la définition de règles pour la communication entre deux entités paires.

Il est défini par :

- * un certain nombre de primitives (listen, socket, bind...),
- * la définition d'un ordre d'échange d'utilisation de ces primitives (bind doit être utilisé après socket),
- * la définition d'un ensemble de question/réponse attendu (un accept réussi après un connect)

Toute violation du protocole entraîne l'échec de la communication, mais peut également bloquer l'émetteur et/ou le récepteur.

Il est nécessaire de faciliter :

- ▷ la **conception** : définir les primitives, l'ordre d'échange et les interactions entre émetteur et récepteur ;
- ▷ la **validation** : vérifier qu'il fonctionne en permettant un dialogue tel qu'il a été décidé ;
- ▷ la **correction** : faire en sorte qu'il soit possible de sortir ou d'éviter une situation de blocage ;

- La solution :**
- a. Élaboration de scenarii pour définir le protocole ;
 - b. Modélisation du fonctionnement du protocole ;
 - c. Obtention d'une spécification formelle du protocole.

Les moyens : Utilisation d'automate fini et de réseaux de Pétri.



Organisation des échanges

- * choix du modèle de conception générale :
centralisé ou répartie, « client/serveur » ou « égal à égal »
- * choix du type de communication :
orienté connexion ou datagramme, TCP ou UDP

Comportement du serveur si nécessaire

- gestion d'un seul client à la fois ;
- gestion de plusieurs clients simultanément ;

Durée du déroulement du protocole

- * le protocole est limité à une seule transaction ;
- * le protocole peut s'étendre sur plusieurs transactions :
Comment mémoriser les étapes du protocole ?

Imaginer des scénarii d'usage du protocole

- dérouler le fonctionnement suivant les différents comportements prévus ;
- en cas d'erreur de l'interlocuteur, du réseau (pertes de messages, arrivée des données dans le désordre, etc.) ;

Formaliser le protocole

- * définir le format des échanges ;
- * combiner les différents scénarii pour définir le comportement complet ; choisir le comportement pour la fiabilité du protocole (éviter les blocages, garantir la disponibilité, se protéger des comportements malveillants, etc.).

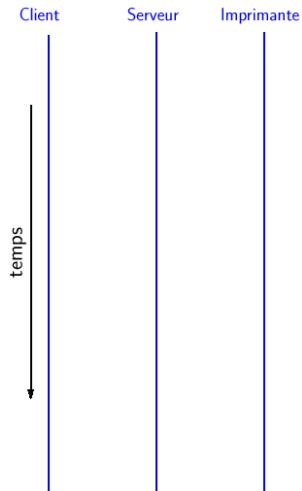


Modélisation d'une transaction : «Message Sequence Charts»

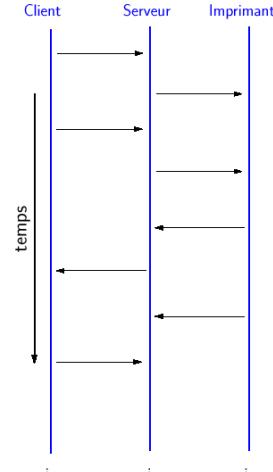
18

- * utilisés pour modéliser les échanges au cours du temps entre un nombre fini de processus.
- * permettent de définir des «scenarii» pour détailler le comportement d'un protocole sur des exemples de cas concrets.

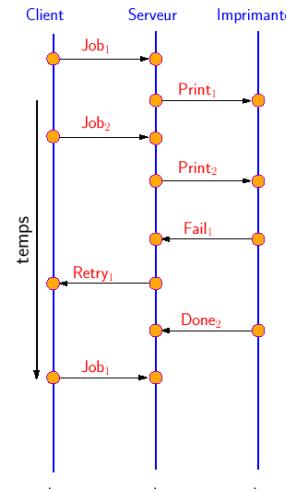
les différents acteurs



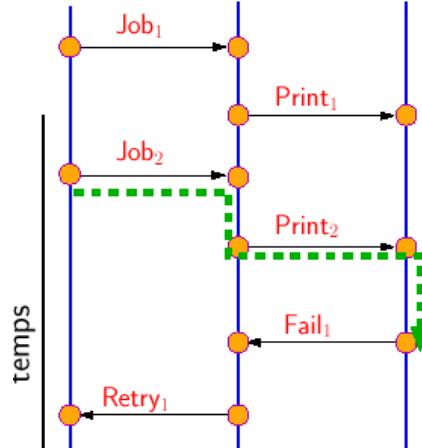
un scenario d'échange



le protocole

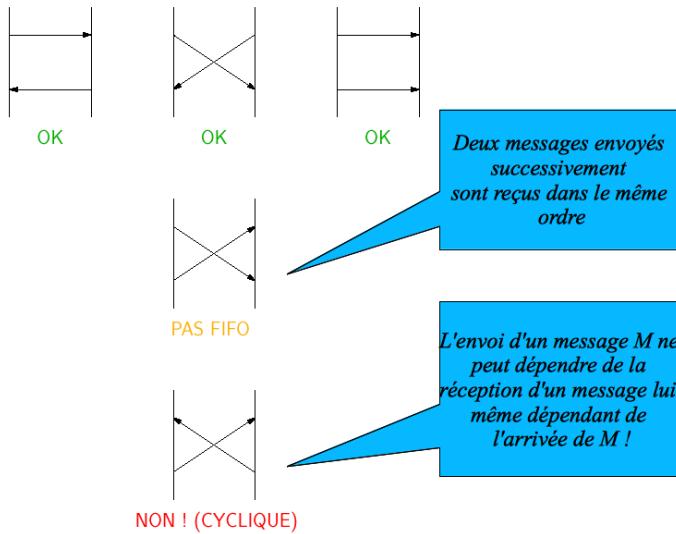


une transaction complète



- * des événements (en orange) : réception et émission placé sur chaque processus ;
- * des échanges : des flèches ;
- * le contenu des messages : étiquette sur les flèches.
- * un ordre sur les échanges ;
- * le MSC permet de :
 - ◊ définir des **exigences** ;
 - ◊ détecter les **mauvais comportements**.





Limitations

- modélisation limitée à une transaction ou à une session complète (plusieurs transactions la suite) ;

Pour décrire tout le protocole, il faut envisager de nombreux scenarii, voire tous les scenarii possibles!

Lorsque plusieurs échanges sont nécessaires pour une même transaction ou lorsque plusieurs transactions sont nécessaires pour une même session :

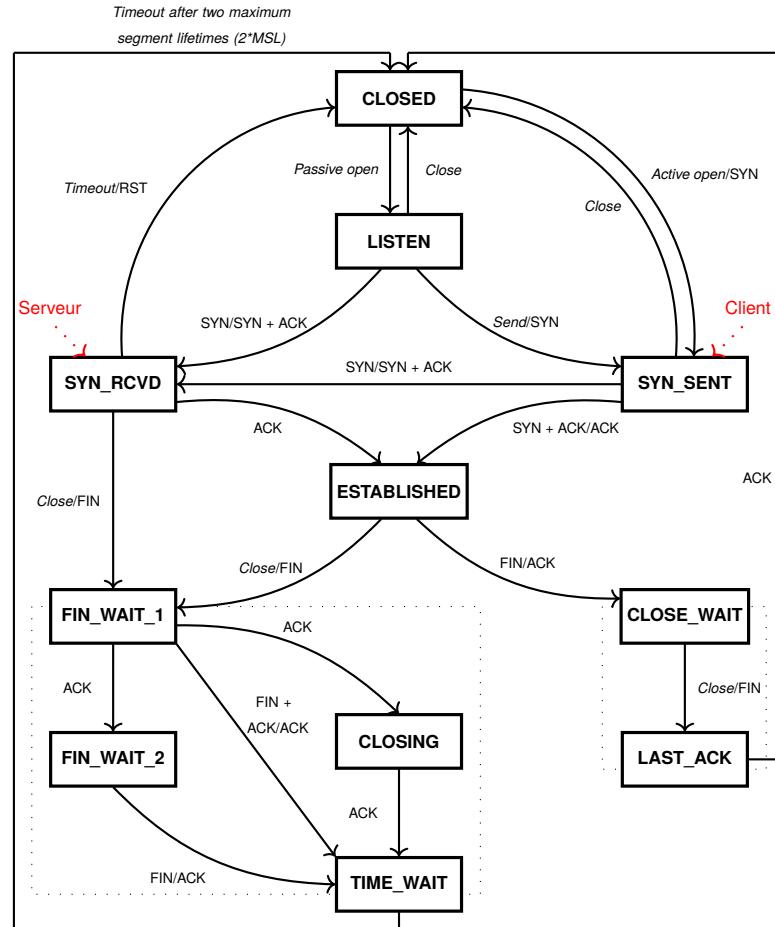
- nécessité de mémoriser «où on en est» par rapport à ces différents échanges/transactions :
 - chaque mémorisation peut modifier le scenarii : reprendre une transaction peut être impossible à partir d'un certain temps, une session peut s'interrompre automatiquement au bout d'un certain temps, etc.
 - pour chaque mémorisation il faudrait reprendre et définir tous les scenarii...

Un scenario décrit un cas d'usage ou «use case» : il sert à illustrer, pour une situation donnée comment le protocole va se comporter.

Pour décrire complètement le protocole il faut un outil permettant de décrire son « comportement » global en tenant compte des « mémorisations » possibles : la modélisation par automate à nombre fini d'états.



La modélisation d'un protocole avec un automate fini



- * **protocole «texte»** : échange de lignes de commandes au format ASCII 7bits ;
- * utilise le protocole de transport entre le client et le serveur de type **TCP**.
- * **très simple**, ce qui explique sa popularité et sa facilité de mise en oeuvre.

Differentes versions :

- o HTTP/0.9 version de base avec requête/réponse le document est renvoyé directement ;
- o HTTP/1.0 version normalisée (RFC 1945) avec comme amélioration, l'ajout d'en-tête pour la description des ressources échangées (utilisation du format MIME RFC822), d'informations supplémentaires envoyées par le client (format de données supporté ou désiré, description de la version et de la marque du navigateur...)
- o HTTP/1.1 ajout de connexions persistantes entre le client et le serveur en vue de l'échange de plusieurs ressources par l'intermédiaire de la même connexion (transfert des différents éléments d'un même document composite).
- o HTTP/2 les requêtes et les réponses peuvent être fragmentées et ces fragments peuvent être envoyés dans un ordre quelconque pour accélérer leur prise en charge par le navigateur
- o HTTP/3 aka QUIC, «*Quick UDP Internet Connections*», basé sur UDP, combine des éléments de TCP/TLS/HTTP2

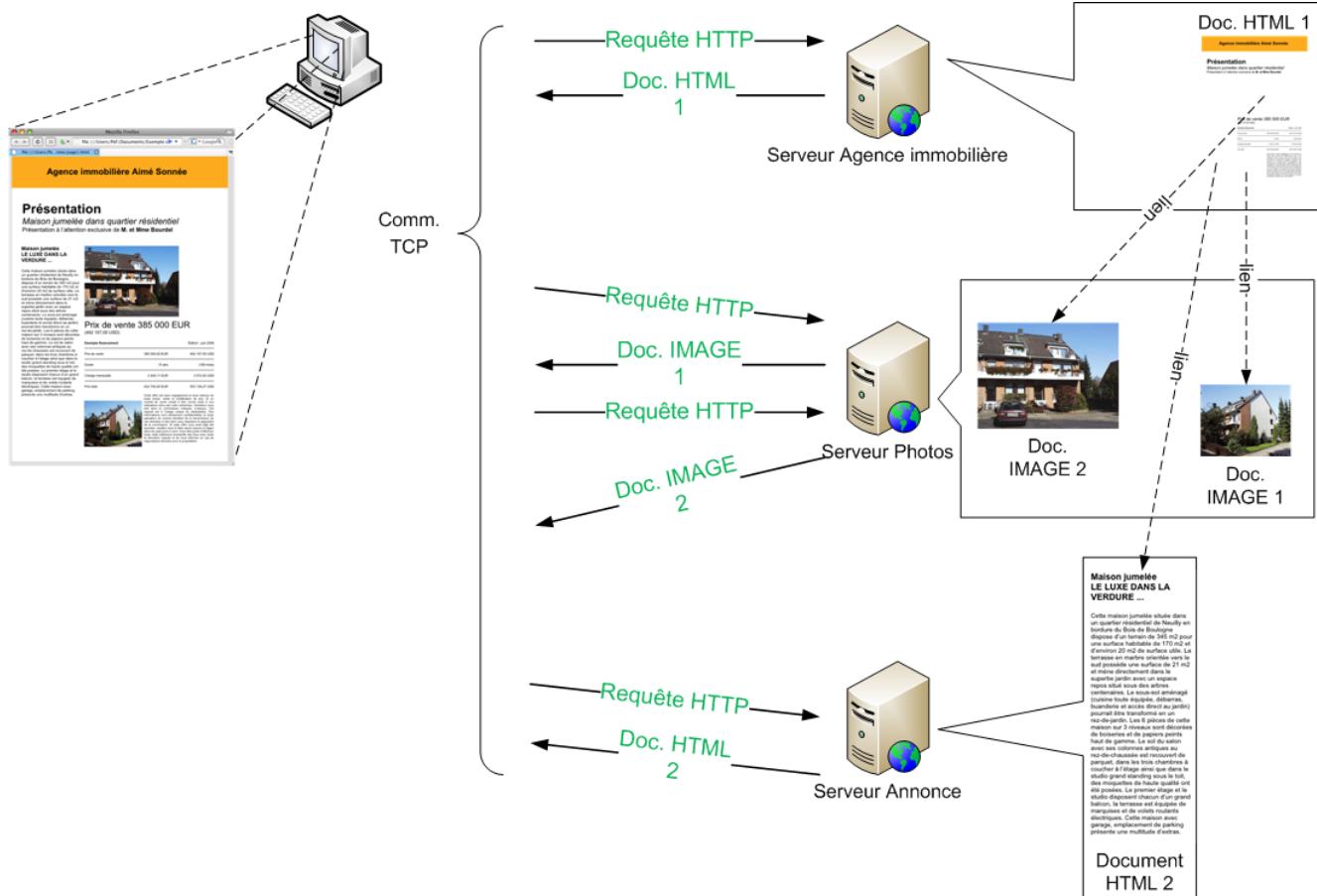
```
xfce4-terminal xterm
pef@darkstar:~$ socat stdio tcp:p-fb.net:80
GET /toto HTTP/1.0
Host: p-fb.net

HTTP/1.1 301 TYPO3 RealURL redirect for missing slash
Server: nginx
Date: Wed, 04 Sep 2019 21:53:27 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Location: http://p-fb.net/toto/
```



Le protocole HTTP, «HyperText Transfer Protocol», RFC 1945

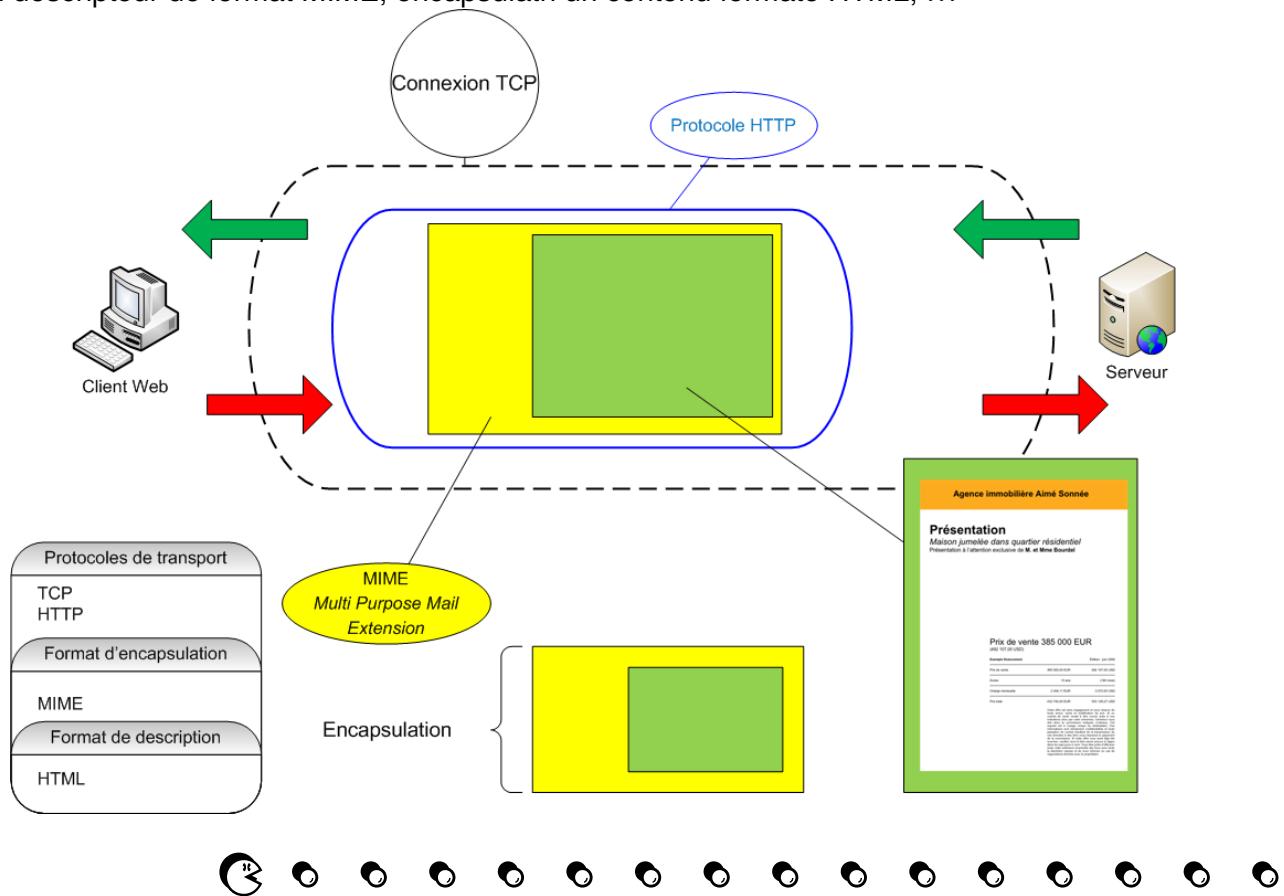
22



Le protocole HTTP

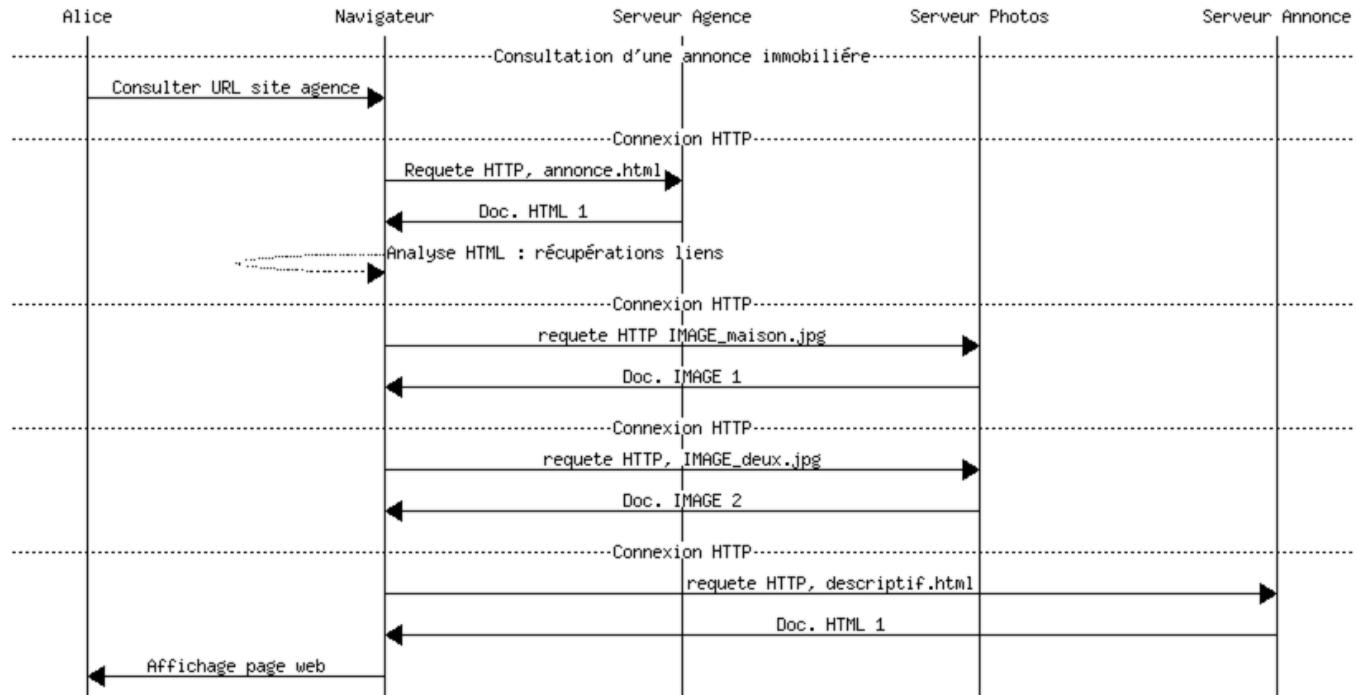
23

Utilisation du «mode connecté» : protocole de transport TCP, encapsulant le protocole HTTP pour échanger un descripteur de format MIME, encapsulatn un contenu formaté HTML, ...



Une pile de protocole

- * protocole « utilisateur » ou abstrait: consultation d'une page web ;
- * protocole de transport: TCP ;
- * protocole d'échange : HTTP ;
- * format d'échange : MIME.



Rapport entre les différents protocoles

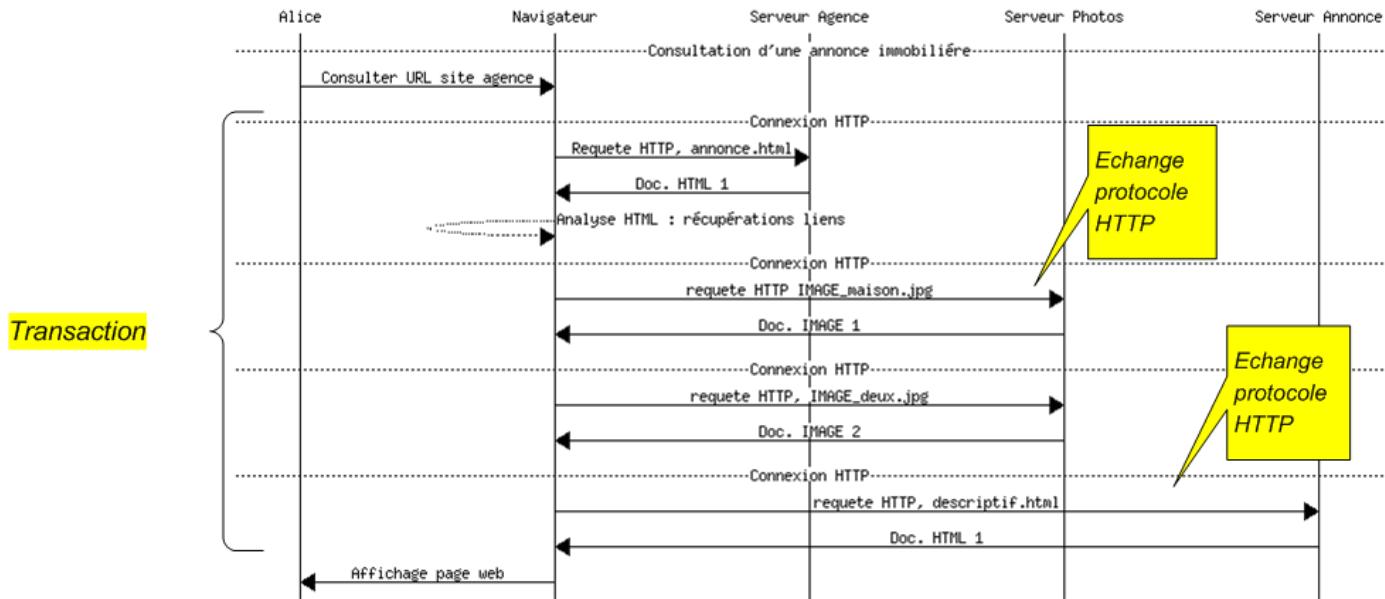
Le protocole abstrait est celui qui intéresse l'utilisateur (ici, Alice), c-à-d. « naviguer sur le Web ».

L'unité élémentaire de ce protocole est la transaction (Alice charge une page Web).

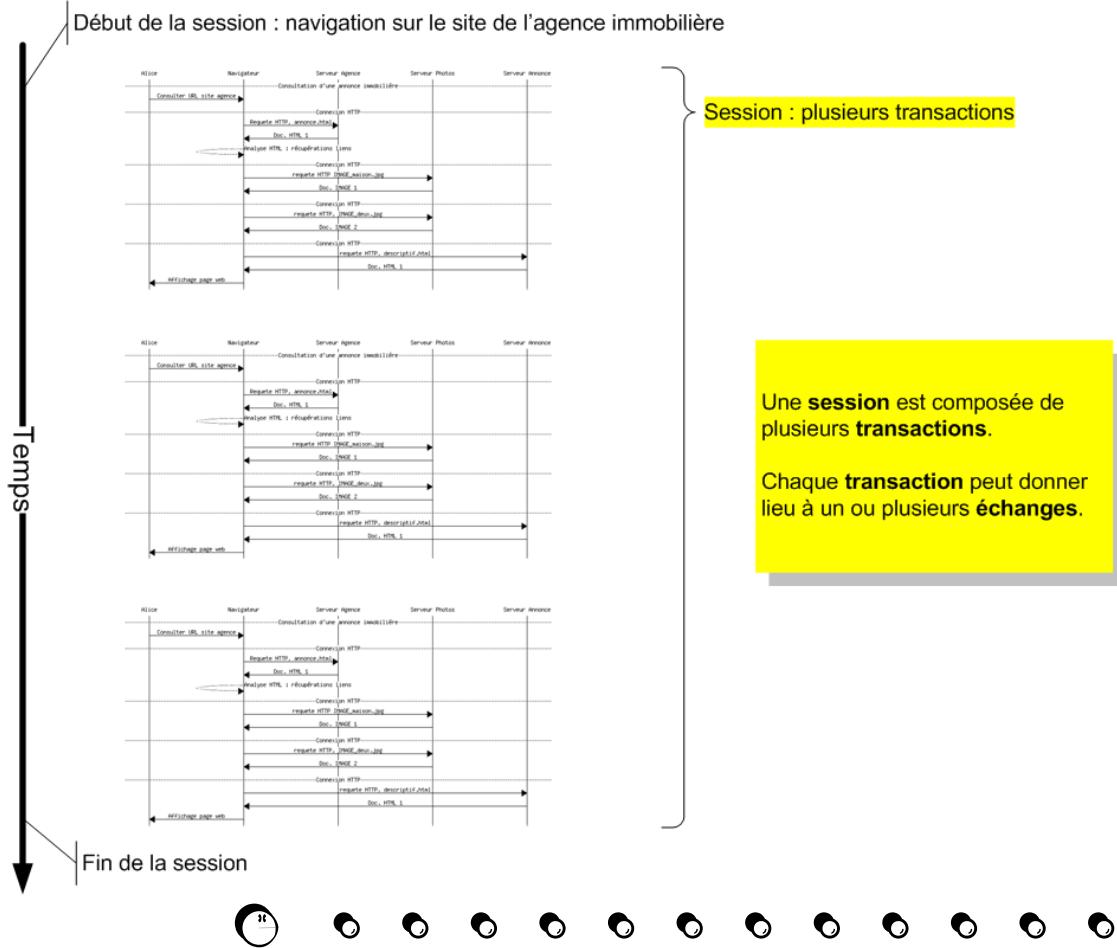
Le navigateur d'Alice réalise plusieurs échanges au format HTTP pour récupérer les contenus multimédia.

La notion de session décrit l'ensemble des transactions qui ont un certain lien entre elles.

Par exemple : entrer dans le magasin virtuel, s'identifier, remplir son caddie, payer et quitter le site.



Utilisation de cookies, de données de formulaires, de contenus JSON, d'URL particulière (REST), etc.



Le protocole HTTP : utilisation de cURL

La commande «curl» va récupérer les données au format JSON et ces données vont être formatées par le module «json.tool» :

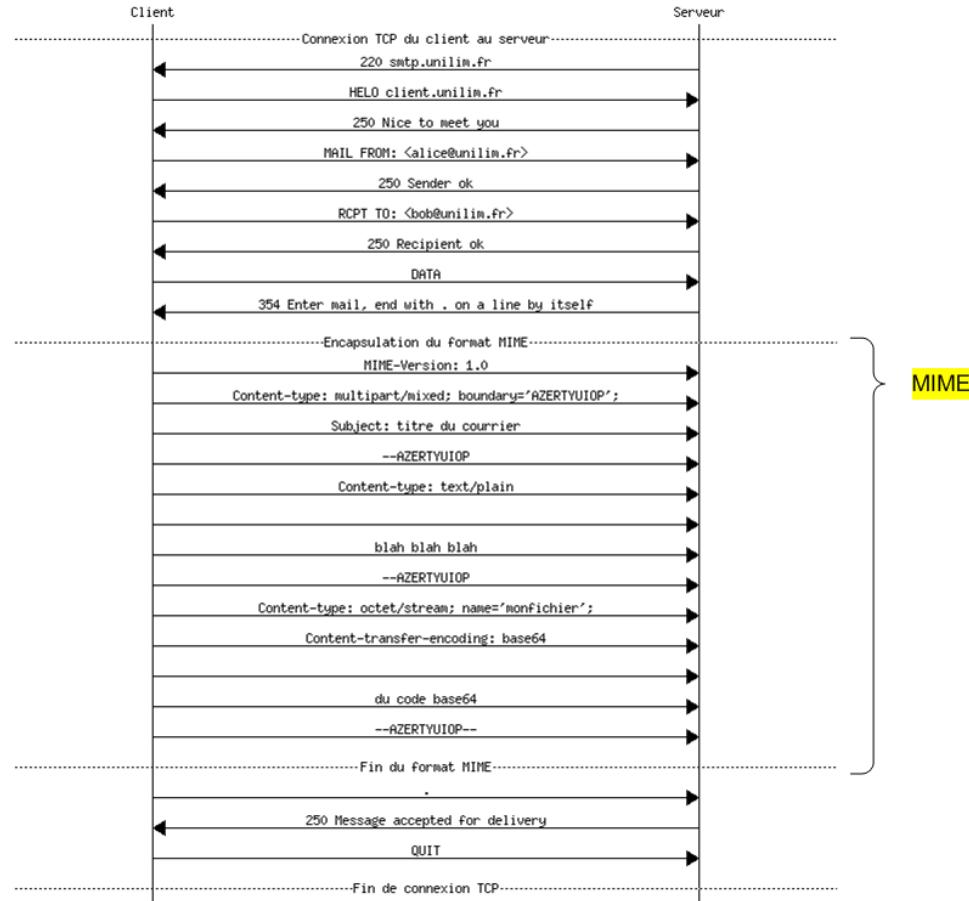
```
xterm └── pef@darkstar:/Users/pef $ curl -sH 'Accept: application/json' http://api.icndb.com/jokes/random | python3 -m json.tool
{
    "type": "success",
    "value": {
        "categories": [
            "nerdy"
        ],
        "id": 543,
        "joke": "Chuck Norris's programs can pass the Turing Test by staring at the interrogator."
    }
}
}

xterm └── pef@darkstar-8:/Users/pef $ curl -vI http://www.unilim.fr/
*   Trying 164.81.1.97...
* TCP_NODELAY set
* Connected to www.unilim.fr (164.81.1.97) port 80 (#0)
> HEAD / HTTP/1.1
> Host: www.unilim.fr
> User-Agent: curl/7.54.0
> Accept: */*
>
< HTTP/1.1 301 Moved Permanently
HTTP/1.1 301 Moved Permanently
< Server: nginx
Server: nginx
< Date: Mon, 11 Sep 2017 10:56:46 GMT
Date: Mon, 11 Sep 2017 10:56:46 GMT
< Content-Type: text/html
Content-Type: text/html
< Connection: keep-alive
Connection: keep-alive
< Location: https://www.unilim.fr/
Location: https://www.unilim.fr/
<
* Connection #0 to host www.unilim.fr left intact
```



Le protocole SMTP

28



Le protocole UPnP, «Universal Plug and Play»

* HTTPU, «HTTP unicast»

0000	01 00 5E 7F FF FA B4 07 F9 F3 3A 73 08 00 45 60	..^.....:s..E`
0010	00 F9 00 00 40 00 04 11 DA 04 A4 51 07 44 EF FF@.....Q.D..
0020	FF FA 07 6C 07 6C 00 E5 F1 76 4E 4F 54 49 46 59	...1.1...vNOTIFY
0030	20 2A 20 48 54 54 50 2F 31 2E 31 0D 0A 48 6F 73	* HTTP/1.1..Hos
0040	74 3A 20 32 33 39 2E 32 35 35 2E 32 35 35 2E 32	t: 239.255.255.2
0050	35 30 3A 31 39 30 30 0D 0A 4E 54 3A 20 75 72 6E	50:1900..NT: urn
0060	3A 6E 75 6C 6C 73 6F 66 74 2E 63 6F 6D 3A 64 65	:nullsoft.com:de
0070	76 69 63 65 3A 41 6E 64 72 6F 69 64 3A 31 0D 0A	vice:Android:1..
0080	4E 54 53 3A 73 73 64 70 3A 61 6C 69 76 65 0D 0A	NTS:ssdp:alive..
0090	43 61 63 68 65 2D 43 6F 6E 74 72 6F 6C 3A 6D 61	Cache-Control:ma
00a0	78 2D 61 67 65 3D 33 30 0D 0A 4C 6F 63 61 74 69	x-age=30..Locati
00b0	6F 6E 3A 68 74 74 70 3A 2F 2F 31 36 34 2E 38 31	on:http://164.81
00c0	2E 37 2E 36 38 3A 33 34 34 38 38 0D 0A 69 64 3A	.7.68:34488..id:
00d0	39 37 37 34 64 35 36 64 36 38 32 65 35 34 39 63	9774d56d682e549c
00e0	0D 0A 6E 61 6D 65 3A 73 61 6D 73 75 6E 67 20 47	..name:samsung G
00f0	54 2D 49 39 30 30 30 0D 0A 70 6F 72 74 3A 33 34	T-I9000..port:34
0100	34 38 38 0D 0A 0D 0A	488



Des processus qui échangent des messages

▷ **synchrones**

- ◊ l'émetteur attend que le récepteur ait reçu le message (*envoyer un fax par exemple*) ;
- ◊ le récepteur qui attend un message est bloqué jusqu'à sa réception ;
- ◊ **Avantage** : l'émetteur et le récepteur sont dans un état connu ;
- ◊ **Inconvénient** : fort couplage entre les processus.

▷ **asynchrones**

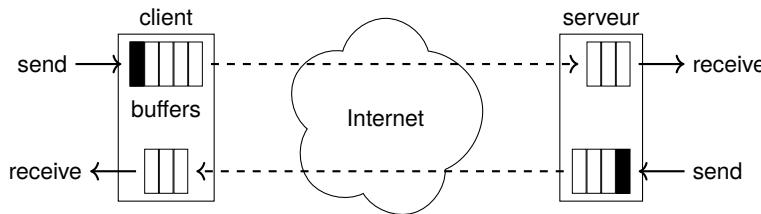
- ◊ l'émetteur n'est pas bloqué en attente de réception (*poster une lettre*) ;
- ◊ le récepteur peut fonctionner suivant deux modes :
 - * réception bloquante si pas de message ;
 - * réception non bloquante avec témoin de réception (*boîte aux lettres américaine*) ;
- ◊ **Avantage** : l'émetteur et le récepteur sont indépendants au cours du temps
- ◊ **Inconvénients** :
 - * pas d'acquittement implicite
 - * pas de relation entre les états de l'émetteur et du récepteur
 - * difficultés en cas d'erreurs !

Solution : le modèle de l'Invocation à distance, «*Remote Procedure Call*» ou «rendez-vous»

- ◊ correspond à la demande d'exécution d'une fonction à un autre processus (*téléphoner*) ;
- ◊ est accompagnée d'un passage de messages entre processus ;
- ◊ comme pour l'appel d'une fonction :
 - ◊ l'appelant attend la réponse de l'appelé (la réponse est facultative) ;
- ◊ correspond au **modèle client/serveur** ;
- ◊ peut être **mis en oeuvre** par messages synchrones ou asynchrones (utilisation de tampons, modèle producteur/consommateur).



Modèle producteur/consommateur en mode message asynchrone



Deux points de vue simultanés

- **asynchronisme** entre sites distants : propagation sur le réseau (le réseau n'est pas modélisé mais il pourrait l'être) ;
- **synchronisation locale** sur les tampons d'émission et de réception.

Asynchronisme entre l'émetteur et le récepteur

- * L'émetteur
 - ◊ effectue un envoi ;
 - ◊ reprend son exécution immédiatement après.

Le message est transmis par le réseau de façon asynchrone par rapport à l'émetteur.

Attention

- Le message est transmis **uniquement** à travers le réseau :
- ◊ si le buffer d'envoi est **plein** ;
 - ◊ ou si le programmeur le **décide**.

- * Le récepteur
 - ◊ décide de traiter un message ;
 - ◊ prend le premier message disponible ;

Le message était dans une file d'attente de réception.



Synchronisation locale

Le **processus émetteur** a fourni l'adresse d'un tampon (contenant le message) partagé avec l'interface réseau.

Différents cas possibles :

- ▷ l'émetteur reste bloqué tant que le message n'a pas été envoyé (attente du tampon redevenu libre) ;
 ⇒ *émission bloquante (celle que l'on utilisera en TP !)*
- ▷ l'émetteur reste bloqué tant que le message n'a pas été recopié dans l'interface réseau ;
 ⇒ *émission bloquante moins longtemps (pas accessible en Python)*
- ▷ l'émetteur reprend le contrôle alors que l'interface réseau utilise le tampon, il sera avertit par un signal quand il sera réutilisable.
 ⇒ *émission non bloquante (déconseillée dans le cas d'un échange : on n'a pas de garantie sur l'émission effective, et on peut attendre une réponse qui ne viendra jamais)*

Le **processus récepteur** a fourni l'adresse d'un tampon partagé avec l'interface réseau.

Différents cas possibles :

- ▷ le récepteur reste bloqué tant qu'un message reçu n'a pas été écrit dans le tampon ;
 ⇒ *réception bloquante (celle que l'on utilisera en TP !)*
- ▷ le récepteur continue son exécution et se bloque quand il ne peut plus avancer sans message reçu ;
 ⇒ *réception non bloquante + attente (déconseillé car compliqué à mettre en œuvre)*
- ▷ le récepteur continue mais il peut savoir si un message a été reçu ;
 ⇒ *réception non bloquante + opération de test de message (déconseillé, car débouche souvent sur de l'«attente active»)*

Attention

Gestion du buffer d'envoi et de réception dans le protocole TCP : rôle du bit PUSH !



Éléments importants

- ▷ TSAP, «*Transport Service Access Point*» ;
- ▷ Mode «orienté connexion» vs mode datagramme ;
- ▷ Protocoles TCP et UDP ;
- ▷ Mode «client/serveur» ;



Une **interface de programmation** définie pour mettre en place **simplement** des communications :

- * chaque communication a lieu avec :
 - ◊ **un interlocuteur** : communication «point à point», ou «unicast» ;
 - ◊ **plusieurs interlocuteurs** : communication par «diffusion» ou «multicast» ;
- * la communication correspond à l'échange de données entre les interlocuteurs :
 - ◊ des **données en continu** : flux d'octets de taille indéfinie, non connue à l'avance ;
 - ◊ des **paquets** : données de taille fixe et réduite connue à l'avance.

Deux types de communication uniquement en TCP/IP

1. mode «connecté»

- ◊ elle ne concerne que **deux interlocuteurs** : un de chaque côté (point à point) ;
- ◊ les données arrivent les unes après les autres dans «l'ordre d'émission» ;
- ◊ la communication est **bi-directionnelle** (dans les deux sens) ;
- ◊ elle est «full-duplex», les deux interlocuteurs peuvent échanger **simultanément** ;
- ◊ il y a une **garantie contre la perte de données**.

*C'est le mode offert par le protocole **TCP**, «Transmission Control Protocol».*

2. mode «datagramme»

- ◊ elle peut concerner un ou plusieurs interlocuteurs (unicast ou multicast) ;
- ◊ les données sont groupées dans des **paquets de taille limitée** ;
- ◊ il peut y avoir des **pertes de paquets**.

*C'est le mode offert par le protocole **UDP**, «User Datagram Protocol».*

Attention

Le mode «connecté» est simulé par TCP sur un réseau en mode «datagramme».



Modèle Client/Serveur

Un logiciel «serveur» **attend** la communication en provenance d'un logiciel «client».

Localisation du logiciel serveur

- un ordinateur est localisable sur Internet grâce à son adresse IP ;
- un ordinateur ne possède habituellement qu'une adresse IP joignable ;
- un ordinateur peut exécuter plusieurs programmes qui peuvent vouloir communiquer simultanément ;
- il faut **multiplexer ces communications** en «sachant» avec quel programme communiquer : notion de «port» !

À chaque processus communiquant est associé un port

Pour une communication en «mode connecté» :

- * un Serveur qui **attend** la connexion du client ;
- * un Client qui **effectue** la connexion au serveur.

Pour localiser le Serveur ? Connaître le numéro de port où attend la communication !

Comment connaître le numéro de port ?

Le point sur les communications sur un ordinateur :

- * chaque communication est associée à **un seul programme donné** (logiciel de messagerie, navigateur web, client de chat, etc) ;
- * chaque communication se fait suivant un **protocole donné** (SMTP, POP pour récupérer le courrier, HTTP, etc) ;
- * chaque protocole est associé à un «serveur» particulier : serveur SMTP pour l'envoi de courrier, serveur Web, serveur FTP, etc.
- * un **numéro de port identifie un serveur donné** : il faut rendre **standard** les numéros de port !

Exemple : http : 80, ftp : 21, smtp : 25, DNS : 53 etc, la liste dans le fichier /etc/services.

Le client veut communiquer avec un serveur donné ? il utilise le port standard associé !



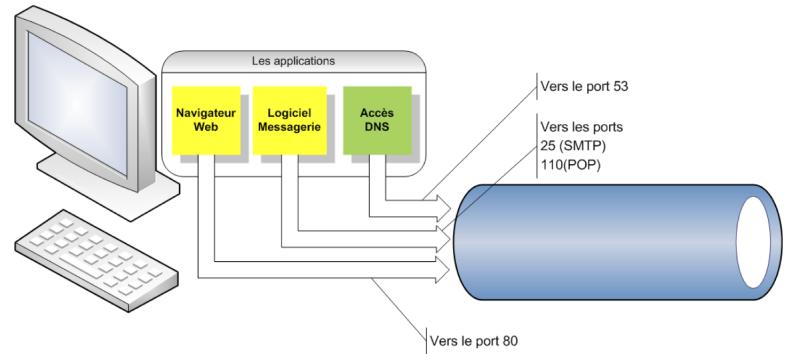
Notion de numéro de port

- ◊ différentes communications peuvent avoir lieu pour des protocoles différents, donc des programmes différents, donc des numéros de port différents ;
- ◊ chaque communication sur une machine est identifiée par un **TSAP**, «*Transport Service Access Point*», c-à-d un couple (@IP, numéro de port).

Comment un ordinateur peut-il voir plusieurs communications simultanément ?

On ajoute également la notion de **numéro de port** :

- * il varie de 1 à 65535 (sur 16 bits) ;
- * il est associé à un seul programme ;
- * du côté de la machine *cliente*, il peut prendre n'importe quelle valeur ;
- * du côté de la machine *serveur*, il permet à la machine cliente de désigner le programme que l'on veut contacter ;



Le port permet de **multiplexer** les communications :

- ◊ chaque datagramme sera identifié par le $TSAP_{source}$ duquel il transporte les données ;
- ◊ tous les datagrammes utilisent le même lien de communication ;
- ◊ lors de leur arrivée sur la machine destination, ils sont identifiés par leur $TSAP_{destination}$ et remis au bon processus.



Objectifs

- * **fournir des moyens de communications** entre processus, IPC, «Inter Processus Communication», utilisable en toutes circonstances : échanges locaux sur la même machine (boucle) ou sur le réseau.
- * **masquer les détails d'implémentation** des couches de transport ;
- * fournir une **interface d'accès** qui se rapproche des **accès fichiers** pour simplifier la programmation

La notion de socket ou de «prise»

C'est un **point d'accès pour les services de transport** :

- ◊ Elle possède un type : *Quel protocole de transport va être utilisé ? Pour quel mode de communication ?*
- ◊ Elle permet d'utiliser un ensemble de **primitives de service** ;
- ◊ Elle **encapsule** des données : un descripteur et des files d'attente des messages en entrée et en sortie ;
- ◊ Elle est **identifiée** par un nom unique : le TSAP, c-à-d «le numéro de port» et une @IP.

Le protocole TCP

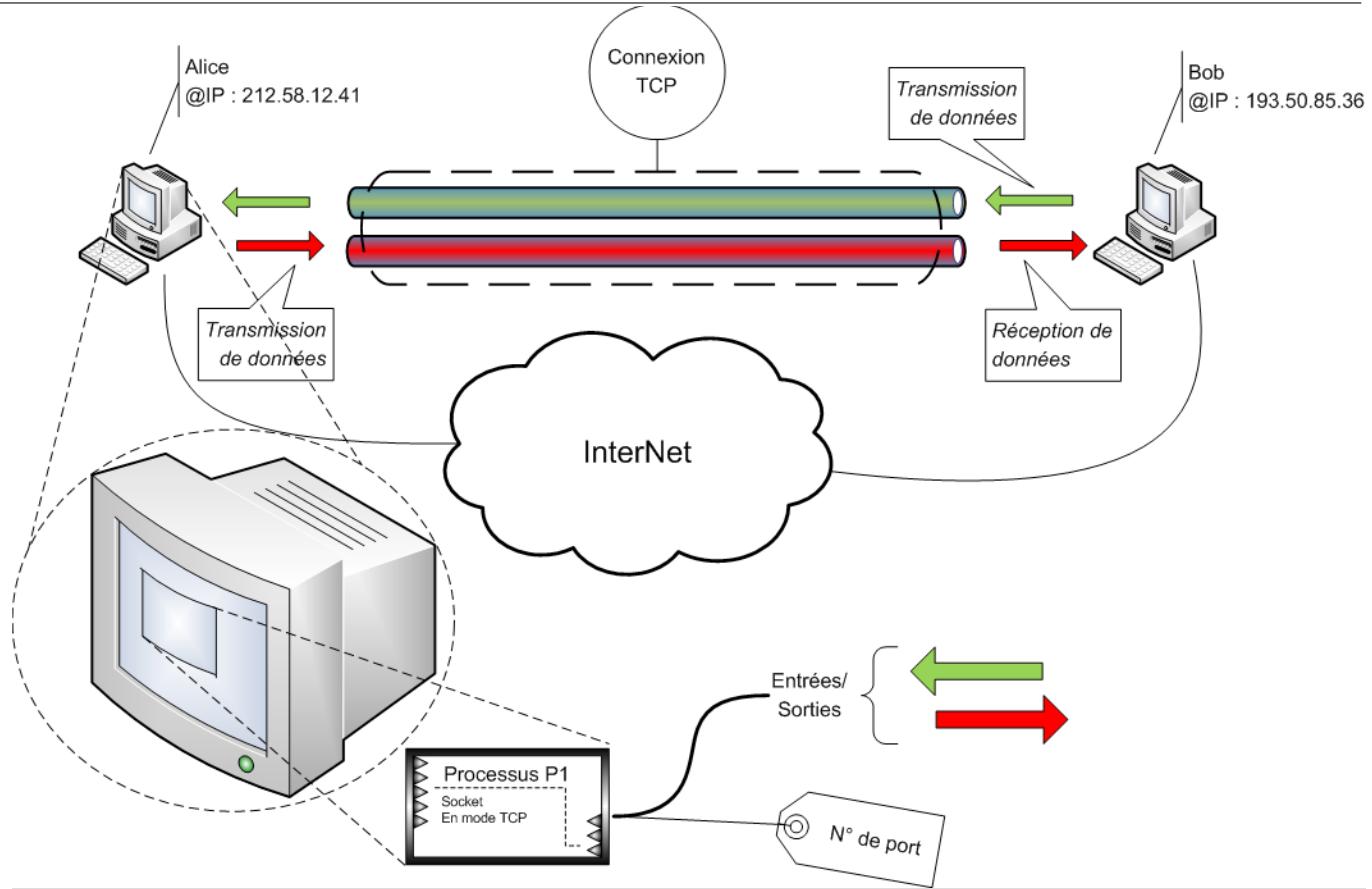
- * C'est un protocole de transport fiable, en mode connecté, en mode bidirectionnel.
- * Une socket TCP peut être utilisée par plusieurs connexions TCP simultanément du côté serveur :
 - ◊ on peut traiter simultanément plusieurs clients ;
 - ◊ on peut configurer une «file d'attente» des clients : si un client se présente il est mis en attente ou renvoyé directement.
- * Une communication est identifiée par le couple d'adresse IP/port, TSAP, des deux extrémités :
$$TSAP_{client} \Leftrightarrow TSAP_{serveur}$$
- * Un échange TCP est un flot continu d'octets. Les données sont reçues dans l'ordre de leur transmission.

Une optimisation de TCP est de temporiser l'envoi des données dans des tampons pour augmenter la taille des envois, mais il est possible de demander l'émission immédiate des données.



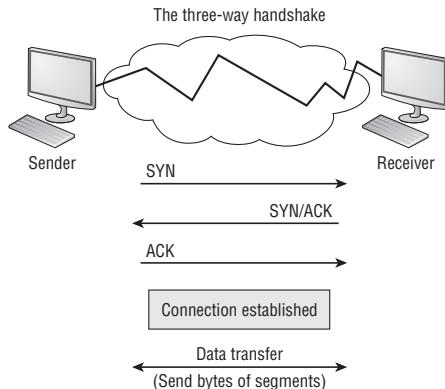
Le protocole TCP : connexion et échanges

38



Une communication «orientée connexion» correspond à :

- ▷ une **demande d'accord** de la part de l'interlocuteur **avant de lui envoyer des données**, c-à-d une 1/2 connexion;
- ▷ un envoi de données **sans perte et sans erreur**;
- ▷ une communication **bi-directionnelle** (d'un interlocuteur vers l'autre et vice-versa) et **«full duplex»** (chaque interlocuteur peut communiquer simultanément avec l'autre);



Celui qui **initie** la communication est le client.

Celui qui **attend** la communication est le serveur.

Firewall & Filtrage

Communication autorisée (non filtrée) :

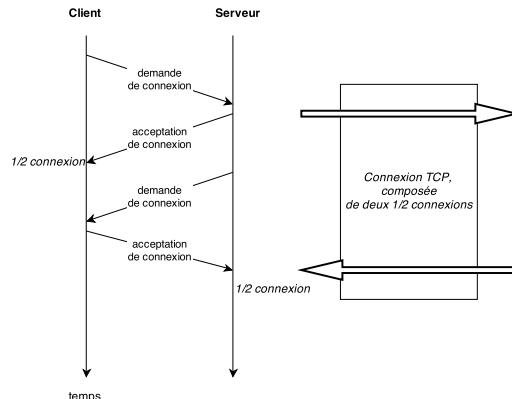
Si le client est dans le LAN et le serveur sur Internet ⇒

Communication Intérieur → Extérieur

L'appartenance du Client au LAN est déduite grâce à la présence du «SYN».

«The three-way handshake», correspond à l'établissement d'une communication depuis le client vers le serveur :

- une demi-connexion du client vers le serveur ;
- une demi-connexion du serveur vers le client ;



À noter : la demande de 1/2 connexion du serveur (SYN), ainsi que son acceptation de la demande de 1/2 connexion du client (ACK), sont transmises dans le même message, d'où la simplification en 3 échanges seulement.



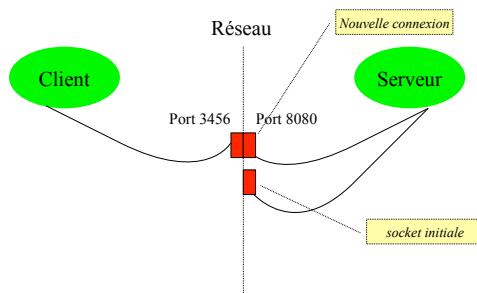
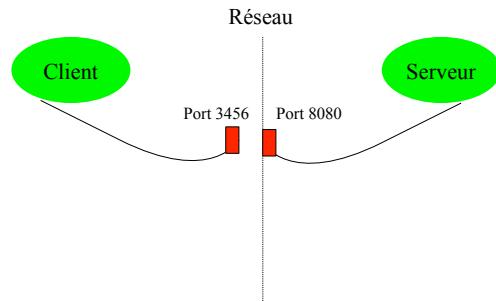
Les «Sockets» Berkeley utilisées dans TCP/IP

40

Les différentes étapes pour l'établissement de la connexion :

Les instructions réseaux à utiliser sont indiquées dans un cadre en fin de ligne.

1. Le serveur attend sur le SAP [`@IP serveur, numéro de port`]
`socket, bind, listen, accept`
2. Le client obtient automatiquement un numéro de port libre (par ex. 3456)
`socket`



3. Le client se connecte au serveur

Le système d'exploitation du client et du serveur, mémorise la connexion par un couple :

$$(TSAP_{client} \Leftrightarrow TSAP_{serveur}) \\ [@IP client, 3456] \Leftrightarrow [@IP serveur, 8080]$$

Cette connexion peut être affichée avec la commande Unix « `netstat` » ou « `ss` » sous Linux.

Remarques :

- ◊ La primitive de programmation `accept` retourne au serveur une nouvelle socket associée à la connexion avec le client.
C'est par cette socket que l'on communique avec le client.
- ◊ Le serveur peut recevoir la connexion de nouveaux clients sur la socket initiale.
Un serveur peut avoir plusieurs communications simultanées avec différents clients.
Chacune de ces communications correspond à un couple différent de TSAP (le même du côté du serveur associé à un TSAP côté client différent pour chaque communication).



Schéma de fonctionnement en TCP

Serveur

```
1 socket  
2 bind  
3 listen  
4 accept  
5 recv, send  
6 close
```

Client

```
1 socket  
2 connect  
3 recv, send  
4 close
```

Le protocole UDP

- ◊ C'est un protocole de transport **non fiable, sans connexion**.
- ◊ L'échange de données se fait par datagrammes : un «datagramme UDP» = un «datagramme IP».
- ◊ L'ordre dans lequel les paquets sont envoyés peut ne pas être respecté lors de leur réception.

Schéma de fonctionnement en UDP

Serveur

```
1 socket  
2 bind  
3 recvfrom, sendto  
4 close
```

Client

```
1 socket  
2 recvfrom, sendto  
3 close
```



Les primitives de l'interface socket

- * **socket** : permet de créer un TSAP, «Transport Service Access Point», c-à-d un nouveau point d'accès de service de transport

Trois paramètres d'appel :

1-famille d'adresses utilisée :

- 1 AF_UNIX (communication locale à une machine)
- 2 AF_INET (communication réseau avec IPv4)
- 3 AF_INET6 (communication réseau avec IPv6)

2-type de service demandé :

- 1 SOCK_STREAM (flot d'octets en mode connecté)
- 2 SOCK_DGRAM (datagramme en mode non connecté)

3-protocole de transport utilisé

- 1 IPPROTO_TCP
- 2 IPPROTO_UDP
- 3 IPPROTO_ICMP

Le prototype de la fonction socket :

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 #include <netinet/in.h>
4
5 int socket(int famille, int type, int protocole);
```



Le choix d'un numéro de port n'est pas systématique lors de la création de la socket :

- *un client n'a pas besoin de fixer un numéro de port particulier (choisi automatiquement par le système) ;*
 - *un serveur qui attend des connexions doit définir sur quel numéro de port il les attend.*
- * **bind**: permet d'attribuer un numéro de port à la socket.

Trois paramètres d'appel :

- ◊ le numéro de descriptif de la socket (retourné par la fonction `socket`);
- ◊ une structure de données adresse de socket de type `sockaddr_in`;
- ◊ la taille de cette structure ;

```
1 #include <sys/socket.h>
2 struct sockaddr_in {
3     short           sin_family;
4     u_short         sin_port;
5     struct in_addr  sin_addr;
6     char            sin_zero[8];
7 }
```

Exemple d'utilisation :

```
1 struct sockaddr_in adresse_socket;
2 adresse_socket.sin_family = AF_INET;
3 adresse_socket.sin_port = 16;
4 /* Conversion htonl dans le sens reseau */
5 adresse_socket.s_addr = htonl(INADDR_ANY);
6
7 if ((s = socket(AF_INET, SOCK_STREAM, 0)) < 0)
8 { /* cas d'erreur */ }
9 if (bind(s, &sin, sizeof(sin)) < 0)
10 { /* cas d'erreur */ }
```



- * **listen :**
 - ◊ Utilisée dans le **mode connecté** lorsque plusieurs clients sont susceptibles de demander une ou plusieurs connexions avec le serveur.
 - ◊ Il permet de fixer le nombre d'appel maximum que pourra traiter le serveur avant de les rejeter (les appels non gérés immédiatement sont alors mis en attente).

```
1 int listen (int descripteur_socket, int max_connection);
```

- * **accept :**
 - ◊ Utilisée dans le **mode connecté**, permet de se bloquer en attente d'une nouvelle demande de connexion.
 - ◊ Après l'accept, la connexion est complète entre les deux processus.
 - ◊ Pour chaque nouvelle connexion entrante, la primitive accept renvoie un pointeur sur une **nouvelle socket** de structure identique à la précédente :
 - * la socket originale sert à établir une nouvelle connexion ;
 - * la nouvelle socket permet l'échange avec le client associé.

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 int accept (int nouvelle_socket, struct sockaddr_in *adresse_client,
4             int longueur_adresse_client);
```



- * **connect :**

- ◊ Cette primitive permet à un client de se **connecter** à un serveur.
- ◊ Elle ouvre une connexion entre le client et le serveur.
- ◊ On doit lui fournir l'adresse IP du serveur (la partie locale est renseignée automatiquement).
- ◊ Pour chaque opération d'écriture/lecture, seul le descripteur de la socket est à fournir à chaque fois.

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3 int connect(int descripteur_socket,
4             struct sockaddr_in *adresse_serveur,
5             int longueur_adresse);
```

- * **send, recv**

- ◊ Ces primitives permettent l'**échange d'information** au travers de la socket.
- ◊ Elles s'utilisent de la même façon que les instructions `read` et `write` sur fichier avec une option supplémentaire pour préciser des options de communication.

```
1 #include <sys/types.h>
2 #include <sys/socket.h>
3
4 int send (int socket, char *zone, int longueur_zone, int options);
5 int recv (int socket, char *zone, int longueur_zone, int options);
```

Les options permettent d'indiquer si les données urgentes, etc.



Le client

```
1 import socket, sys
2
3 adresse_symbolique_serveur = 'localhost' # la machine elle-même
4 numero_port_serveur = 6688 # le numéro de port sur le serveur
5
6 # réalisation de la requête DNS pour obtenir l'adresse IP
7 adresse_serveur = socket.gethostbyname(adresse_symbolique_serveur)
8
9 ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
10
11 try:
12     ma_socket.connect((adresse_serveur, numero_port_serveur)) #TSAP désignant le serveur
13
14 except Exception as e:
15     print (e.args)
16     sys.exit(1)
17
18 while 1:
19     entrée_clavier = input(':>')
20     if not entrée_clavier:
21         break
22     ma_socket.sendall(bytes(entrée_clavier, encoding='UTF-8')+b'\n')
23
24 ma_socket.close()
```



Le serveur

```
1 import socket
2
3 masque_acces = '' # filtre les clients, ici aucun n'est filtré
4 numero_port_serveur = 6688 # identique à celui du client
5
6 # création de la socket d'attente de connexion
7 ma_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM, socket.IPPROTO_TCP)
8
9 # Permet de ne pas attendre pour réutiliser le numéro de port
10 ma_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
11
12 # Accroche le numéro de port à la socket
13 ma_socket.bind((masque_acces, numero_port_serveur))
14
15 # Configure la file d'attente
16 ma_socket.listen(socket.SOMAXCONN)
17
18 # L'accept renvoie une nouvelle socket
19 (nouvelle_connexion, tsap_depuis) = ma_socket.accept()
20 print ("Nouvelle connexion depuis ", tsap_depuis)
21 while 1:
22     ligne = nouvelle_connexion.recv(1000) # au plus 1000
23     if not ligne :
24         break
25     print (ligne)
26 nouvelle_connexion.close() # fermeture de la connexion vers le client
27
28 ma_socket.close() # fermeture de la socket d'attente de connexion
```



Éléments importants

- ▷ Deux topologies théoriques : diffusion et «point-à-point» ;
- ▷ Les réseaux utilisés et le matériel d'interconnexion ;
- ▷ La gouvernance d'Internet: les organisations et les RFCs ;
- ▷ Le réseau TCP/IP : adressage, encapsulation, routage direct & indirect ;
- ▷ Le DNS : global et local ;
- ▷ La configuration du poste de travail.

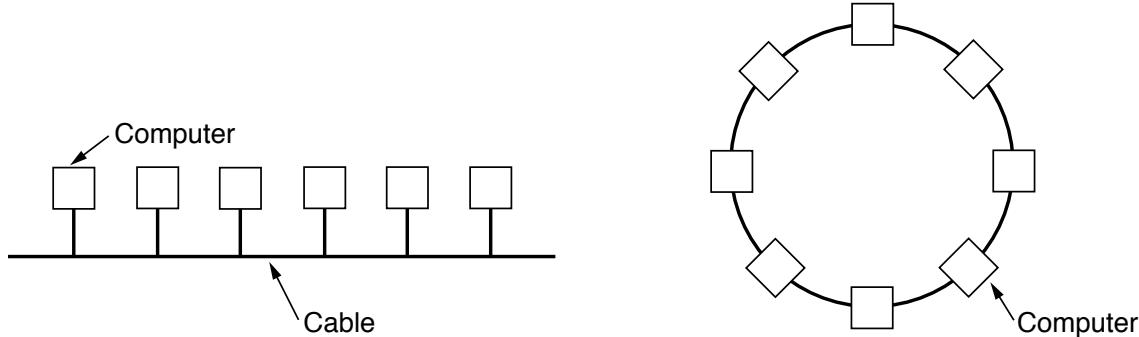


3 Fondamentaux – réseaux diffusion et point-à-point

49

Le réseau en mode «diffusion»

Les réseaux à diffusion, *broadcast network*, n'ont qu'un **seul canal de communication** que toutes les machines partagent.



Une machine envoie de **petits messages** qui sont reçus par toutes les autres machines :

- * dans le message un **champ d'adresse** permet d'identifier le destinataire
- * à la réception du message, une machine teste ce champ :
 - ◊ si le message est pour elle, elle le traite ;
 - ◊ sinon, elle l'ignore.

Exemple :

un couloir sur lequel débouche un certain nombre de portes de bureau.
quelqu'un sort dans le couloir et appelle une personne,
tout le monde entend l'appel mais une seule personne répond à l'appel
cas des annonces dans les gares ou les aéroports



Le réseau en mode «diffusion»

Contraintes

- * chaque machine appartenant au réseau **doit disposer** d'une adresse.

Avantages

- envoyer un message **vers tout le monde** en utilisant une adresse particulière :
 - Ce message est traité par toutes les machines.
 - Ce procédé est appelé «diffusion générale» ou «broadcasting».
- transmettre un message à **un sous-ensemble** de machines :
 - Ce procédé est appelé «diffusion restreinte» ou «multipoint» ou «multicast».
 - Une façon de faire consiste à utiliser les n bits d'adresse de la manière suivante :*
 - * associer un bit à l'indication de mode multipoint
 - * utiliser les n-1 bits restants pour l'identification du groupe.
 - * permettre à chaque machine d'appartenir à un ou plusieurs groupes.
- connaître le **temps de transmission** d'un message :
permet de simplifier des algorithmes de communication : «je suis sûr que le récepteur a reçu mon message, mais je ne connais pas son état et s'il a pu le traiter.»

Inconvénients

- * la **rupture** du support de transmission **entraîne l'arrêt du réseau**.
Le «hub» ou «switch» tombe en panne.
- * la **panne d'un des matériels** connectés au réseau **ne provoque pas de panne du réseau** (en général...).



Dans le cas d'Ethernet, mais aussi de WiFi, de Bluetooth...

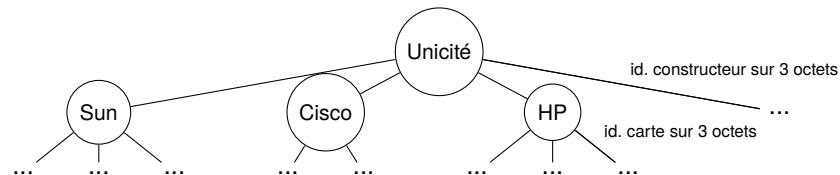
Chaque carte réseau possède une adresse matérielle appelée adresse MAC (Medium Access Control) :

- ▷ **unique** par rapport à toutes les cartes réseaux existantes !
- ▷ **exprimée sur 48 bits** ou 6 octets.
 - ◊ **Syntaxe**: 08:22:EF:E3:D0:FF
 - ◊ **Adresse de Broadcast**: FF:FF:FF:FF:FF:FF (en IPv4).

Pour garantir l'**unicité** :

- a. des «tranches d'adresses» sont affectées aux différents constructeurs :

00:00:0C:XX:XX:XX	Cisco
08:00:20:XX:XX:XX	Sun
08:00:09:XX:XX:XX	HP
00:09:BF:XX:XX:XX	Nintendo
00:D0:F1:XX:XX:XX	Sega



Ce préfixe est appelé OUI, «Organization Unique Identifier».

La liste est consultable à <http://standards.ieee.org/regauth/oui/index.shtml>.

- b. **chaque constructeur** numérote différemment chaque carte réseau qu'il construit.

Avantage

impossible de trouver deux fois la même adresse dans un même réseau

Inconvénient

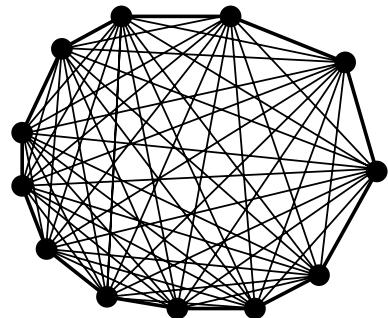
elle ne donne **aucune information sur la localisation** d'une machine
«dans quel réseau est la machine avec qui je veux parler ?»

Solution

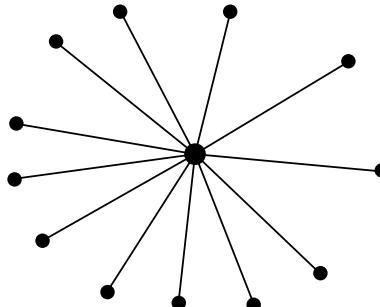
utilisation de l'adresse IP !



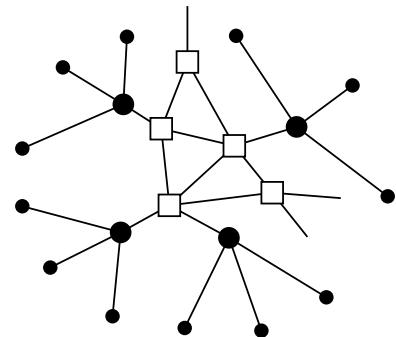
Réseaux en mode «point à point»



(a)



(b)



(c)

Ces réseaux sont formés d'un **grand nombre de connexions** entre les machines prises **deux à deux**.

Le trajet des communications est rendu plus complexe :

- ▷ pour aller de la source au destinataire, un message doit alors **passer par un plusieurs intermédiaires**.
- ▷ il existe plusieurs routes de **longueurs différentes** pour joindre ces deux machines, il est nécessaire d'utiliser de **bons algorithmes d'acheminement des messages** ;

Inconvénient

- ▷ le **temps de transfert** d'un message devient presque impossible à prévoir.
- ▷ il est nécessaire de faire du **routage** des messages dans le réseau.



Le **support physique** relie uniquement **une paire** d'équipements à la fois.

La communication est :

- ▷ **directe** : si les deux équipements sont connectés entre eux ;
- ▷ **indirecte** sinon : deux équipements ne peuvent communiquer que par l'**intermédiaire d'autres nœuds** du réseau.

Le besoin de «routage»

Problème ?

Comment choisir par quels intermédiaires passer, ce qui s'appelle du «*routage*» ?

Cas d'un réseau en «étoile»

L'algo. de routage est simple : le site central reçoit et renvoie tous les messages.

Le fonctionnement est simple, mais la panne du site central paralyse tout le réseau.

Cas d'une «boucle simple»

chaque nœud recevant un message de son voisin en **amont** le réexpédie à son voisin en **aval**.

En cas de **non remise**, le message est **retiré** par le nœud émetteur pour éviter des boucles inutiles.

Si l'un des nœuds **tombe en panne**, le réseau est **bloqué**.

Une solution partielle est d'utiliser une «double boucle».

Ça se complique, on a deux possibilités :

Maillage régulier

l'interconnexion est **totale** (plus besoin de passer par un intermédiaire)

la **fiabilité** est **maximale**

le **coût** en câblage est **maximal**

Maillage irrégulier

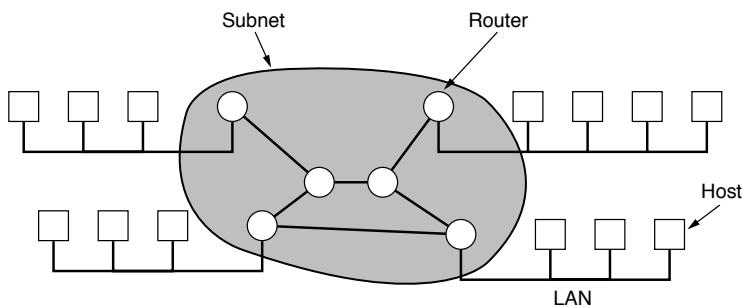
l'interconnexion **n'est plus totale**

la **fiabilité** **diminue**

le **routage** des messages peut devenir **complexe**

impossible de prévoir le temps de transfert d'un nœud à l'autre.

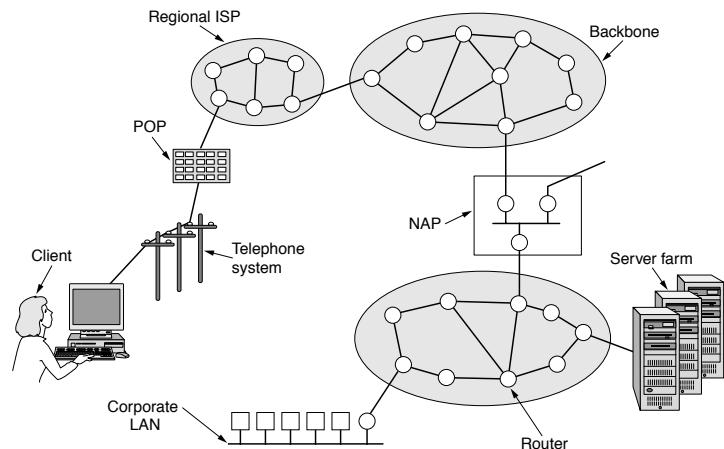




- ▷ **diffusion** : réseau de petite taille, LAN, *Local Area Network* ;
Exemple : Ethernet
- ▷ **point-à-point** : réseau d'interconnexion, constitué uniquement de routeur et de ligne de transmission
Exemple : liaison satellite.
- ▷ la **combinaison des deux** : WAN, *Wide Area Network*.

Inter(connexion)Net(work) :

- * du client à la maison ;
- * de la ligne téléphonique au POP, «Point of Presence» vers ATM ;
- * ou en passant par de la fibre optique pour une liaison IP directe vers l'ISP ;
- * en passant par l'ISP, *Internet Service Provider* ;
- * au réseau national : *backbone* ;
- * par une connexion à un réseau, *Network Access Point* ;
- * vers le LAN de l'entreprise...



Expérimentation et compréhension

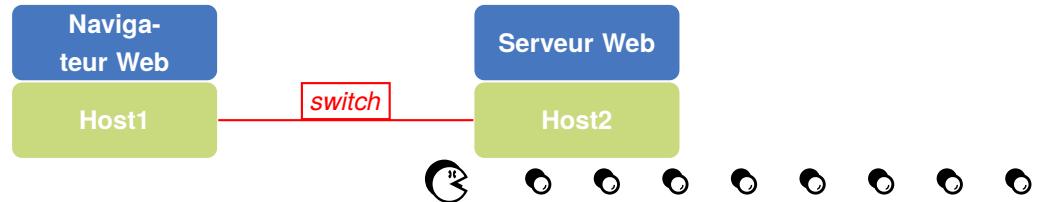
- ▷ «apprendre par la pratique» : se rapprocher du réel, expérimenter ;
- ▷ pour la programmation ? Programmer, modifier, compiler etc.
- ▷ pour le réseau ? Disposer de plusieurs machines, de switches, de routeurs etc.
- ▷ pour l'administration réseaux ? Manipuler les commandes de configuration et de diagnostique.

Plateforme	Avantages	Inconvénients
Matériel	rapide, «réelle» mais impossible de disposer de tous les matériels des différents constructeurs	coûteux, difficile à partager, difficile à reconfigurer et à changer
Simulateur	peu coûteux, flexible, permet d'accélérer l'expérimentation (temps artificiel)	peut nécessiter des ajustements, ne correspond pas à des opérations normales du système d'exploitation, pas toujours interactif
Émulateur	<i>peu coûteux, flexible, assez réelle, interactif</i>	<i>plus lent qu'une solution matérielle, peut ne pas donner de bons résultats lors du multiplexing de plusieurs communications</i>

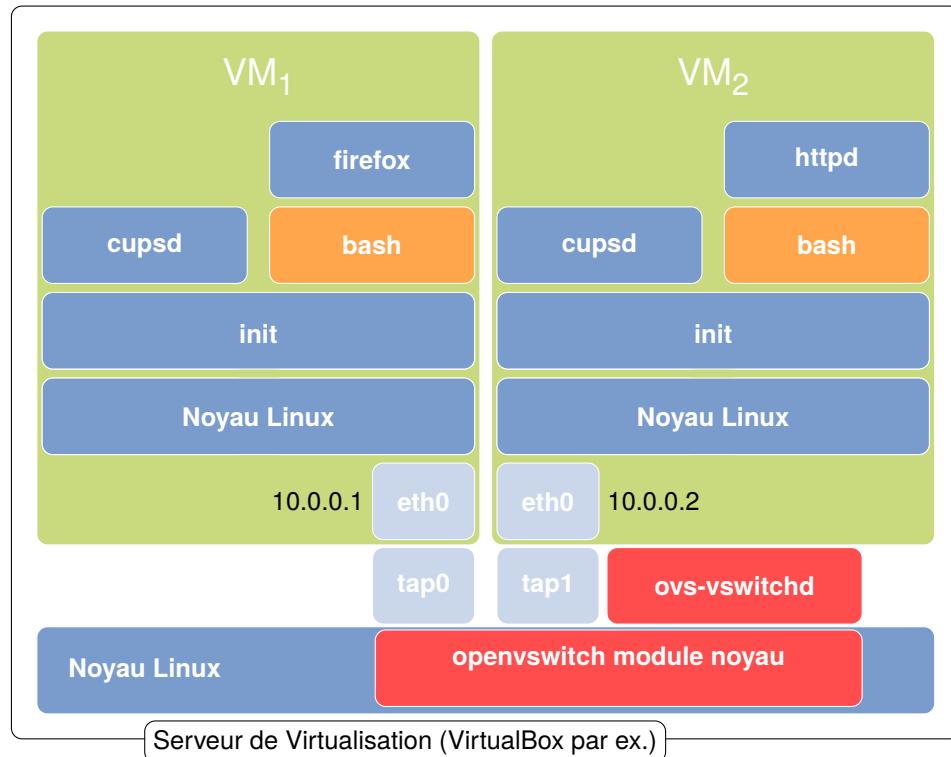
Exemple de réseau

Deux hôtes connectés par un **switch** :

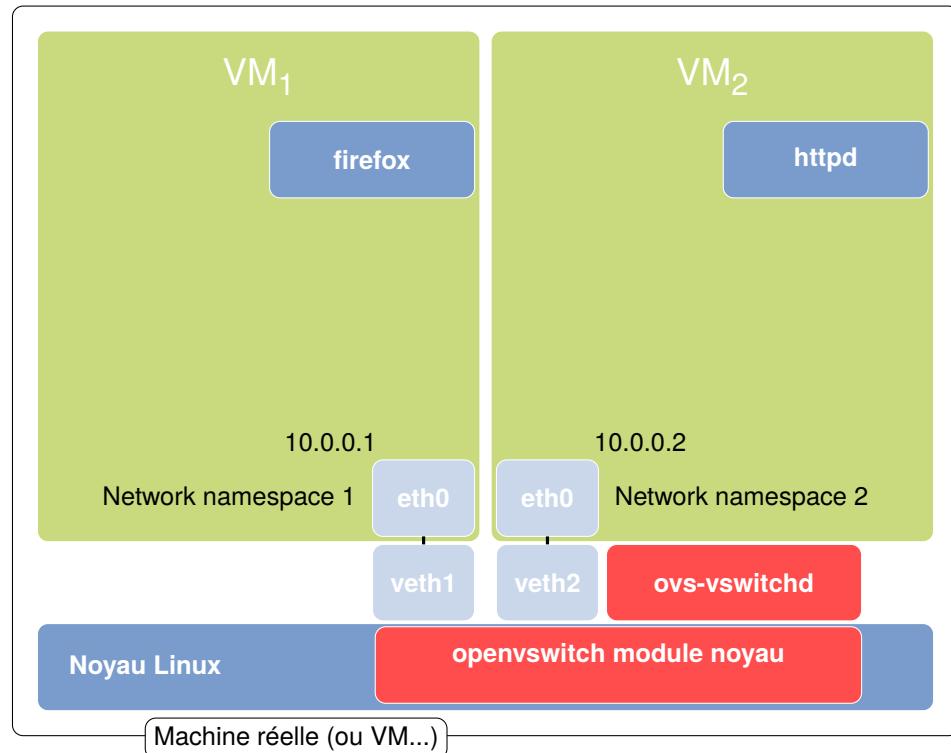
- le «Host2» héberge un serveur Web, c-à-d un processus «httpd» ;
- le «Host1» exécute un navigateur Web, qui va se connecter au serveur Web ;
- les deux machines sont connectées à un switch et appartiennent au même réseau IP.



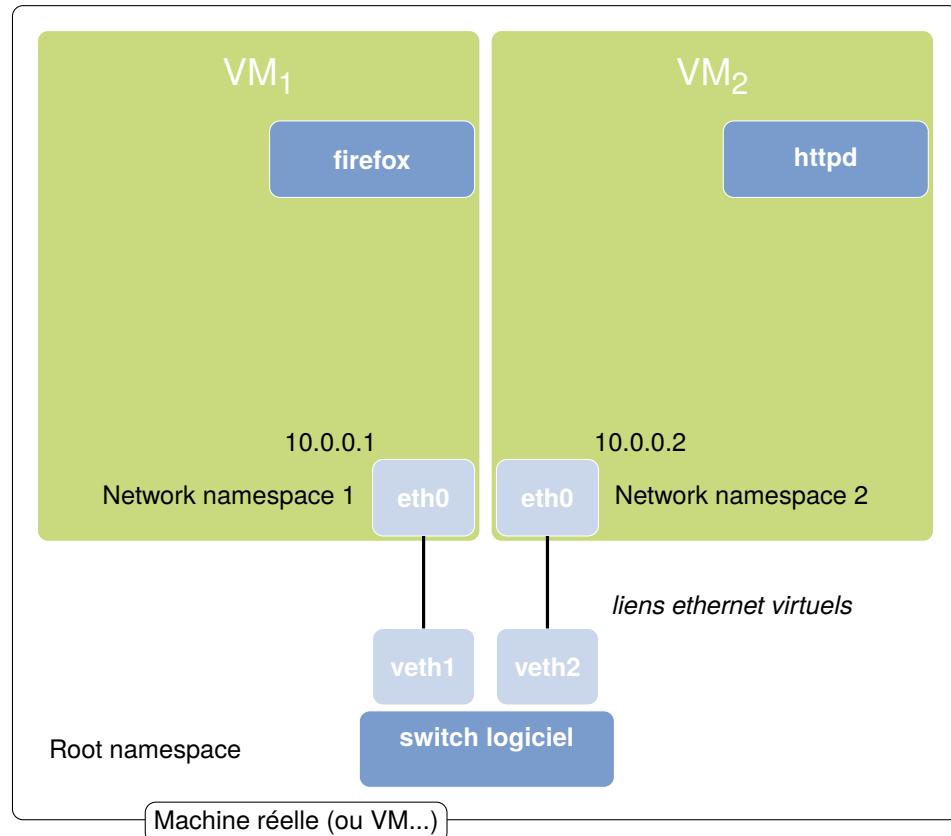
- installation de Virtualbox : solution gratuite de virtualisation (permet par ex. de faire tourner un Linux sur un Windows) ;
- création de deux VMs : chaque VM est une machine complète sur laquelle il faut installer Linux (~ 10Go par VM) ;
- configuration du réseau connectant les deux VMs.



- une seule machine sous Linux ou une seule VM sous Windows ;
- création d'espace de nom réseau, «network namespace» ;
- un switch logiciel ;



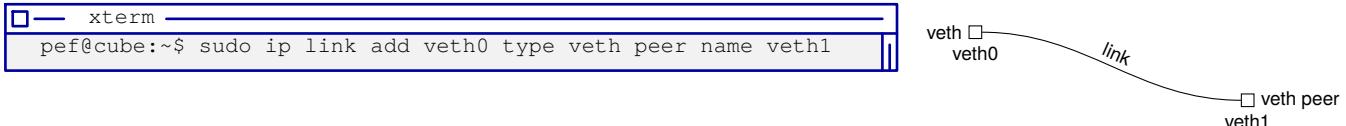
- l'espace de nom «root» : celui correspondant à la machine, c-à-d sa pile réseau TCP/IP ;
- deux espaces de noms en plus, 1 & 2 : deux piles TCP/IP supplémentaires (soient 3 au total liés par un switch logiciel).



- les **interfaces**:

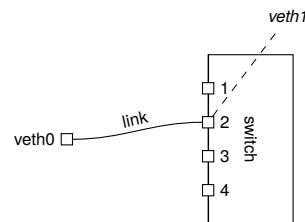
```
□—— xterm
pef@cube:~$ sudo ip link add toto type dummy
pef@cube:~$ ip link
10: toto: <BROADCAST,NOARP> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
      link/ether 5a:01:c1:2f:d9:81 brd ff:ff:ff:ff:ff:ff
```

- les **liens**, «virtual ethernet»:, un lien dispose de deux extrémités, «veth» et «veth peer»:



- les **switches**:

```
□—— xterm
pef@cube:~$ sudo ovs-vsctl add-br mon_switch
pef@cube:~$ sudo ovs-vsctl add-port mon_switch veth1
```



- le **routeur**:

```
□—— xterm
pef@cube:~$ sudo sysctl -w net.ipv4.conf.all.forwarding=1
```

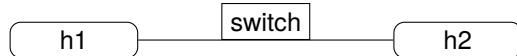
On active la fonction de routage du noyau Linux.

- le **Firewall** pour activer le NAT:

```
□—— xterm
pef@cube:~$ sudo iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -j MASQUERADE
```



Création des deux «hôtes», du switch et câblage



```

xterm
# passe en administrateur
sudo -s
# créer les namespaces pour les hôtes
ip netns add h1
ip netns add h2
# créer le switch
ovs-vsctl add-br s1 ①
# créer les liens
ip link add h1-eth0 type veth peer name s1-h1
ip link add h2-eth0 type veth peer name s1-h2
ip link show ③
# accrocher les liens aux namespaces
ip link set h1-eth0 netns h1
ip link set h2-eth0 netns h2
# afficher le lien depuis le namespace
ip netns exec h1 ip link show ④
ip netns exec h2 ip link show
# connecter les liens au switch
ovs-vsctl add-port s1 s1-h1 ②
ovs-vsctl add-port s1 s1-h2
ovs-vsctl show
# activer les interfaces du namespace root ⑤
ip link set dev s1-h1 up
ip link set dev s1-h2 up
# activer les interfaces des namespaces h1 et h2 ⑥
ip netns exec h1 ip link set dev h1-eth0 up
ip netns exec h2 ip link set dev h2-eth0 up
  
```

Remarques

- ① => La création du switch est conservé lors d'un redémarrage du système.
- ② => L'ajout d'un lien est également conservé **même** si le lien n'existe plus (il sera automatiquement rajouté lors de la réutilisation du même nom pour le lien veth).
- ③ => On affiche la liste des interfaces présentes.
- ④ => Pour exécuter une commande dans le namespace h1: ip netns exec h1 <commande>
- ⑤ => le «netnamespace root» correspond à la pile réseau racine, c-à-d celle initiale de l'hôte.
- ⑥ => l'interface de chaque netnamespace doit être activé depuis le netns correspondant.

Si on veut changer l'adresse MAC d'une interface :

```

xterm
ip link set dev eth0 address 01:02:03:04:05:06
  
```



Configuration de la couche de niveau 3, «*network*»

61

Configuration d'une adresse IP pour h1 et h2 appartenant au même réseau

```
└── xterm ━
    ip netns exec h1 ip a add dev h1-eth0 10.0.0.1/24
    ip netns exec h2 ip a add dev h2-eth0 10.0.0.2/24
```

Test de la connectivité

```
└── xterm ━
    pef@cube:~$ sudo ip netns exec h2 ping -c 1 10.0.0.1
    PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
    64 bytes from 10.0.0.1: icmp_seq=1 ttl=64 time=0.967 ms

    --- 10.0.0.1 ping statistics ---
    1 packets transmitted, 1 received, 0% packet loss, time 0ms
    rtt min/avg/max/mdev = 0.967/0.967/0.967/0.000 ms
```

Écoute du trafic directement sur l'interface d'un netns

```
└── xterm ━
    root@cube:~# tcpdump -l -i s1-h1
    20:38:51.697615 ARP, Request who-has 10.0.0.2 tell 10.0.0.1, length 28
    20:38:51.698127 ARP, Request who-has 10.0.0.1 tell 10.0.0.2, length 28
    20:38:51.698189 ARP, Reply 10.0.0.1 is-at d2:a8:f5:cc:9d:8b (oui Unknown), length 28
    20:38:51.698289 ARP, Reply 10.0.0.2 is-at 7a:68:37:b4:47:0e (oui Unknown), length 28
    20:39:01.781418 IP 10.0.0.2 > 10.0.0.1: ICMP echo request, id 3410, seq 1, length 64
    20:39:01.781492 IP 10.0.0.1 > 10.0.0.2: ICMP echo reply, id 3410, seq 1, length 64
```

Pour changer le prompt et indiquer le netns du shell courant

```
└── xterm ━
    root@cube:~# cat change_prompt
    INTERFACE=$(ip l | awk -F"[ -]" '/eth0/ { print "[" $2 "]" }')
    PS1="$INTERFACE$PS1"
    #PS1="h1:\w"
    root@cube:~# source change_prompt
    [h1]root@cube:~#
```

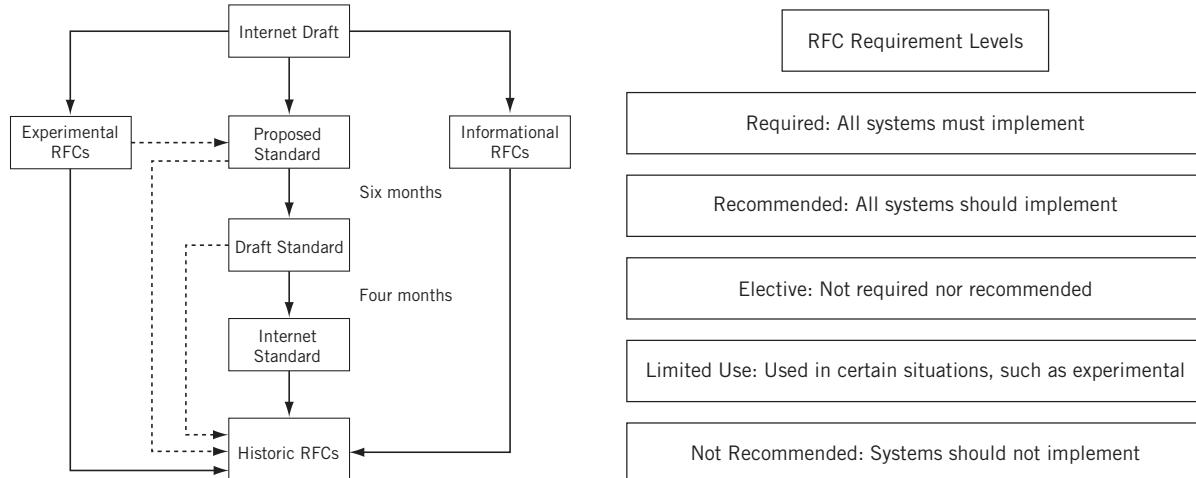


- * **IEEE**, «*Institute of Electrical and Electronics Engineers*» : organisation internationale chargée de superviser le développement et l'adaptation de standards internationaux.
Par exemple dans le cadre des communications sans fil avec l'IEEE 802.11.
- * **ANSI**, «*American National Standards Institute*» : organisation non gouvernementale à but non lucratif contribuant à l'élaboration de standards pour l'industrie en protégeant les intérêts du public.
Par exemple, le code ASCII, American Standard Code for Information Interchange.
- * **EIA**, «*Electronic Industries Association*» : organisation à but non lucratif proche de l'ANSI dédiée à la résolution des problèmes de fabrication de composants électroniques.
La prise du «twisted-pair» appelé RJ-45.
- * **ISO**, «*International Standards Organization*» : le nom vient du grec «isos» qui veut dire «égaux», organisation de standardisation internationale dont les membres appartiennent à des comités de standardisation de différents pays. Elle est basée sur le volontariat (pour les Etats-Unis, c'est l'ANSI qui participe).
*Dans le cadre des réseaux, sa contribution principale est le modèle OSI : «Open Systems Interconnection Reference Model» qui sert de base à l'analyse et la conception des **protocoles** de communication.*
- * **ITU-T**, «*International Telecommunications Union-Telecommunication Standards Sector*» : permettre une infrastructure mondiale non seulement dans les réseaux de données mais également dans la téléphonie, PSTN, «public switched telephone network». Les Nations Unies ont formées un comité le CCITT, «Consultive Committee for International Telegraphy and Telephony» compris dans l'ITU, «International Telecommunications Union».
Tout communication qui traverse les frontières d'un pays doit se conformer aux recommandations de l'ITU-T.
Une technologie comme ATM, «Asynchronous Transfer Mode» est un standard ITU-T.
- * Des **forums** : promouvoir une technologie et sa standardisation.
Exemple le MEF, «Metro Ethernet Forum».
- * Obligations de se conformer aux régulateurs nationaux, comme le **FCC**, «*Federal Communications Commission*» qui surveille, par exemple, l'utilisation des différents fréquences dans le cas de transmission sans fil.



Les «Request for Comment» & l'IETF, «Internet Engineering Task Force»

- IETF : organisme responsable du développement des standards de l'Internet.
 - ◊ son propre système de standardisation des **protocoles** utilisés par les matériels connectés à Internet.
Les protocoles concernés sont plus proches de l'utilisateur (applications) que du matériel.
- Les standards Internet : des spécifications testées et qui doivent être suivies.
 - ◊ la procédure d'élaboration est stricte :
 - * «Internet draft» : document de travail, souvent modifié sans statut particulier et de durée de vie d'au plus 6 mois ;
 - * les développeurs travaillent à partir de ces «drafts» ;
 - * un draft peut être publié sous la forme d'un RFC (juste une appellation, il n'y a plus besoin de retour ou *feedback*).
 - ◊ RFC identifiée par un numéro, librement consultable et, suivant le niveau de *requirements*, **obligatoire**, ou pas.
 - ◊ Tous les RFCs ne sont pas des standards, même ceux définissant des protocoles entiers.
 - ◊ Après un certain temps, la RFC peut arriver à maturité et finir dans les «RFC historiques».



- **InterNIC**, «*Internet Network Information Center*» entre 1992-1998 :
 - ◊ organisme public américain chargé de la gestion centrale des adresses et des noms de domaines Internet et de l'accréditation d'un organisme homologue dans chaque pays, les organismes délégués :
 - * AfriNIC (Afrique),
 - * APNIC (Asie, Pacifique),
 - * ARIN (Amérique du Nord),
 - * LACNIC (Amérique du Sud, îles Caraïbes),
 - * RIPE NCC (Europe, Moyen-Orient)
 - * NIC France ou afnic, NIC Angleterre, etc.
- **ICANN**, «*Internet Corporation for Assigned Names and Numbers*» :
 - ◊ Organisation créée en octobre 1998, pour s'ouvrir à la concurrence
 - ◊ traite les noms de domaine et leur délégation (par exemple VERISIGN Inc. : zone « .com ») ;
 - ◊ exploitation des serveurs de la racine du DNS (ceux qui font autorité) ;
 - ◊ allocation de blocs de numéro IP ;
 - ◊ en France, les prestataires (fournisseurs d'accès) font l'intermédiaire avec l'afNIC
- **IANA**, «*Internet Assigned Numbers Authority*» :
 - ◊ tient l'annuaire : adresses IP & numéros de protocoles ;
 - ◊ adresses IP et numéros d'AS : déléguées aux RIR régionaux, «Regional Internet Registries» ;
 - ◊ numéros de protocoles et de ports (entre 1 et 1023) ;
 - ◊ déléguées aux LIRs, «Local Internet Registry» (eg. FAI).
- Les «**Registrar**» :
 - ◊ Un registrar (bureau d'enregistrement) est une société ou une association permettant le dépôt de noms de domaine internet, dans les TLD, «Top Level Domain», où il n'y a pas de vente directe.
 - ◊ Il faut payer un certain montant pour acquérir et protéger un nom de domaine.
- **GIP Renater** (Groupement d'Intérêt Public) :
 - ◊ Réseau de la recherche en France, «Réseau National de télécommunications pour la Technologie l'Enseignement et la Recherche»



La conception

Contraintes du protocole IP «Internet Protocol» RFC 791 :

- * utiliser la topologie réseau point à point (pour permettre des grandes distances c'est obligatoire) ;
- * la panne d'un équipement du sous-réseau ne doit pas entraîner une rupture du réseau ;
- * privilégier la **disponibilité** du réseau : il doit servir au maximum.

Le but est que les transactions continuent :

- ▷ du moment que l'ordinateur source et l'ordinateur destination fonctionnent ;
- ▷ même si certains routeurs ou certaines lignes de transmission tombent en panne (origine militaire de la création d'Internet par le DoD états-unisens).

Les moyens

- définir une **architecture très souple** pour pouvoir mettre en œuvre des applications très diverses comme le transfert de fichiers ou la transmission de la parole en temps réel (TCP et UDP) ;
- faciliter le **routage** : construire une méthode simple et rapide (opérations binaires par exemple) ;
- permettre le **regroupement de machines** pour les gérer ensemble (regroupement en réseau) ;
- faciliter le travail de l'administrateur (*sisi...*).



Adressage pour le protocole IP (IPv4)

Chaque ordinateur et chaque routeur du réseau Internet possède une adresse IP.

L'adresse IP est une **adresse binaire** composée de deux parties <id. réseau><id. machine> :

- * un **identifiant de réseau** ;
- * un **identifiant machine** pour la distinguer dans le réseau.

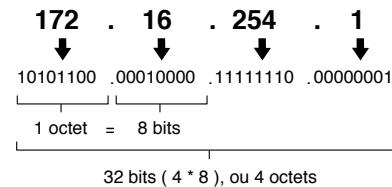
Chaque adresse IP doit être unique pour permettre de la localiser sur la planète.

- * Il existent **différentes répartitions** des 32 bits entre identifiant réseau et identifiant machine :
 - ◊ ces **différentes répartitions** définissent un ensemble de **classes de réseaux** ;
 - ◊ ces classes **ne sont plus utilisées** en CIDR, où on indique uniquement le nombre de bits de la partie réseau ;

Propriétés

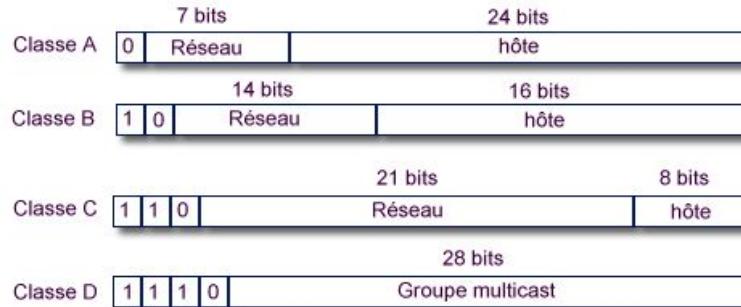
- ▷ Codée sur **32 bits**.
- ▷ Représentée par **commodité** en «décimale pointée» : 4 entiers variant entre 0 et 255 séparés par des points
exemple : 164.81.1.4

Une adresse IPv4 (notation décimale à point)



- ▷ un **organisme officiel**, le NIC, «Network Information Center», est seul habilité à délivrer des numéros d'identification des réseaux.
- ▷ il y a, **en général**, une **seule adresse IP** par interface réseau.
Dans le cas d'un routeur interconnectant 2 réseaux différents, il possède une adresse IP pour chacune de ses interfaces connectées à un réseau.



Les classes de réseaux définies par un « préfixe »

Les adresses de classe A sont peu utilisées. Exemple : 3.0.0.0/8, AS80, GE-CRD - General Electric Company.

Aux niveaux des adresses IP

	32-bit Address Starts with:	Number of Addresses:	% of Address Space
Class A	0 (0-127)	$2^{31} = 2,147,483,648$	50
Class B	10 (128-191)	$2^{30} = 1,073,741,824$	25
Class C	110 (192-223)	$2^{29} = 536,870,912$	12.5
Class D	1110 (224-239)	$2^{28} = 268,435,456$	6.25
Class E	1111 (240-255)	$2^{28} = 268,435,456$	6.25
	First byte	Second byte	Third byte
			Fourth byte



Ces **adresses** permettent :

- * des envois de messages **multi-destinataires** ;
- * désigner la **machine courante** ;
- * désigner le **réseau courant**.

Tout à zéro	L'ordinateur lui-même	
Tout à zéro	id. de machine	Un ordinateur sur le réseau lui-même
Tout à 1	Diffusion limitée au réseau lui-même	
Id. de réseau	Tout à 1	Diffusion dirigée vers ce réseau
127	Nombre quelconque	Boucle

L'**adresse de «boucle»** (127.X.Y.Z) permet d'effectuer :

- ◊ des communications inter-programme sur la même machine
- ◊ des tests de logiciels réseaux. *Dans ces cas là, les paquets ne sont pas réellement émis sur le réseau.*

D'autres **adresses particulières** :

- ◊ 0.0.0.0 est utilisé par une machine pour connaître sa propre adresse IP lors d'un processus d'amorçage (BOOTP).
Elle devra se procurer une adresse IP par l'intermédiaire d'une autre machine.
- ◊ 255.255.255.255 est une adresse de diffusion locale car elle désigne toutes les machines du réseau auquel appartient l'ordinateur qui utilise cette adresse \Rightarrow **pas besoin de connaissance du réseau**.

Réseaux privés, RFC1918

Les adresses pour réseau **privé** ou **intranet** (sans **accès direct** à l'extérieur) :

- * 10.0.0.0/8 : de 10.0.0.0 à 10.255.255.255 \Rightarrow **classe A**
- * 172.16.0.0/12 : 172.16.0.0 à 172.31.255.255 \Rightarrow **pas de classe**
- * 192.168.0.0/16 : 192.168.0.0 à 192.168.255.255 \Rightarrow **classe B**

Ces réseaux ne sont pas «routables» !



Faire le point...

- sur un réseau à datagramme, il circule...**des datagrammes !** ;
- le réseau à datagramme est appelé **réseau IP** : il utilise des algorithmes, des formats de messages définis dans la norme IP, «Internet Protocol» ;
- un **réseau à diffusion** fait circuler des messages de format différent : les **trames** (on parle de **trame Ethernet** ou **IEEE 802.3**) ;
- un datagramme doit emprunter un réseau à diffusion pour atteindre un ordinateur :
 - ◊ principe **d'encapsulation** : le datagramme est «inclus» dans une trame Ethernet;
 - ◊ à l'**adresse IP** d'une machine doit correspondre l'identifiant de cette machine dans le réseau à diffusion : une **adresse MAC** :
 - * l'adresse MAC est attachée à la carte réseau et est choisie par le constructeur de cette carte ;
 - * l'adresse IP est choisi par l'administrateur réseau suivant la configuration qu'il veut donner à son réseau ;

Comment faire la correspondance entre @MAC et @IP ?

- a. c'est **l'ordinateur** qui connaît l'adresse MAC de sa carte réseau ;
- b. c'est **l'ordinateur** qui connaît son adresse IP ;
- c. **Qui** peut dire à quelle adresse IP correspond tel adresse MAC ? **L'ordinateur lui même !**
- d. **Définition d'un protocole** pour « questionner » les ordinateurs : ARP, «Address Resolution Protocol»



Transmission physique des datagrammes IP

La **couche liaison de données** est chargée de :

- ▷ la mise en correspondance des @IP avec les @MAC des interfaces physiques.
- ▷ l'**encapsulation** des datagrammes IP afin qu'ils puissent être transmis sur un support physique particulier.

Lorsque le protocole IP doit envoyer un datagramme à un équipement relié à un réseau à diffusion, la couche liaison de donnée doit construire une trame ethernet avec l'@MAC du destinataire.

Correspondance entre adresses physiques, @MAC, et adresses IP, @IP

Le **protocole ARP**, «*Address Resolution Protocol*» fournit une **correspondance dynamique** entre une adresse IP connue et l'adresse matérielle correspondante.

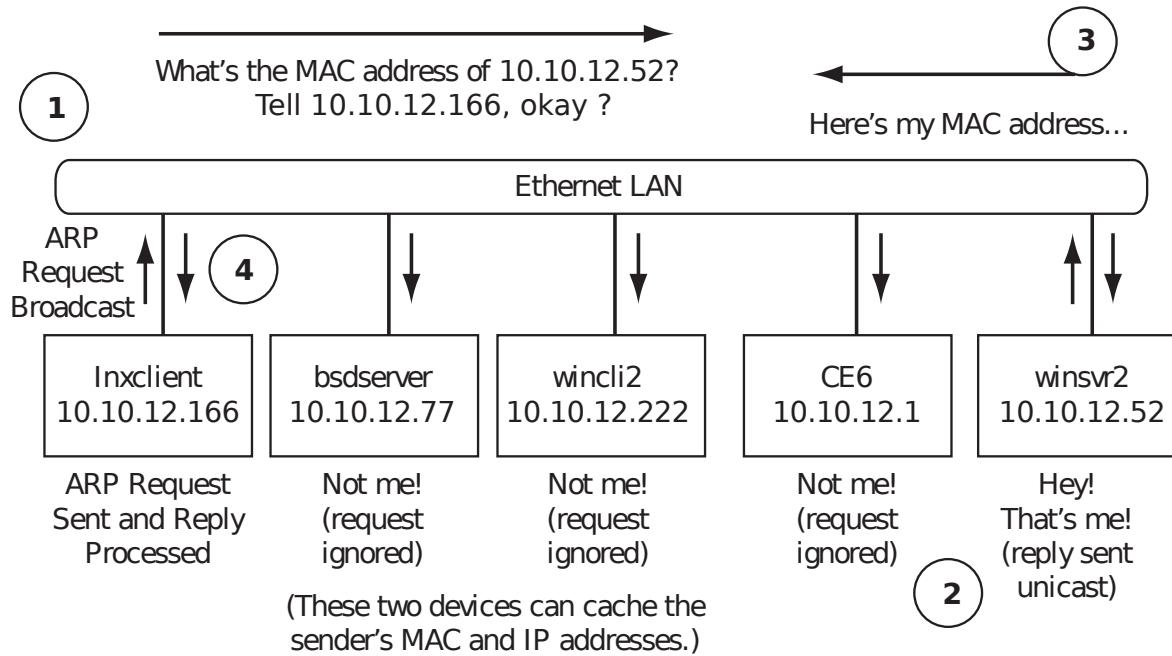
Fonctionnement :

- ▷ ARP dispose d'une **mémoire cache** : lors de la demande de l'@MAC associée à une @IP, il consulte sa **mémoire cache ARP** pour voir si l'@IP distante y est mise en correspondance avec @MAC.
 - ◊ Si c'est le cas le datagramme IP est émis immédiatement, enveloppé dans une **trame Ethernet** envoyée à l'adresse physique destination (@MAC).
 - ◊ Sinon la couche liaison de données construit une **requête ARP**.
- ▷ ARP utilise le principe de «*diffusion*» du réseau local : la requête ARP est transmise en «*broadcast*».
- ▷ Lorsqu'un **message ARP** est reçu, la couche liaison de donnée fait :
 - ◊ une **première vérification** pour voir si c'est une **requête ARP** et que l'@IP demandée correspond à l'@IP locale alors une **réponse ARP** est renvoyée à destination de l'@MAC de l'expéditeur.
La machine répond parce qu'elle est concernée : c'est son adresse qui est demandée.
 - ◊ une **seconde vérification** pour vérifier si l'adresse IP de l'émetteur se trouve déjà dans la **mémoire cache ARP locale** sinon il y a **mise à jour** de la mémoire cache avec cette nouvelle association.
Elle apprend l'association, comme dans le cas d'un «gratuitous ARP», c-à-d une réponse ARP non sollicitée envoyée en broadcast.



Comment échanger réellement sur un réseau local à diffusion ?

- * Les machines ont chacune une carte réseau ;
- * Chaque carte a une **adresse MAC unique** donnée par le constructeur ;
- * Chaque machine dispose d'une **adresse IP** donnée par l'administrateur du réseau.



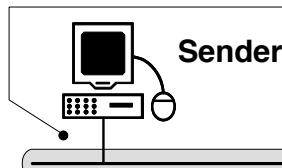
La machine 10.10.10.166 demande l'@MAC de la machine 10.10.12.52.



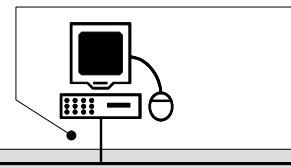
Illustration d'ARP : les échanges

72

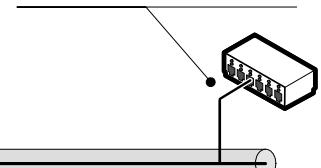
IP: 192.200.96.21
MAC: 01-23-45-67-89-AB



IP: 192.200.96.22
MAC: 00-01-03-1D-CC-F7



IP: 192.200.96.20
MAC: 49-BD-2F-54-1A-0F



ARP Request: to FF-FF-FF-FF-FF-FF

Sender MAC: 01-23-45-67-89-AB
Sender IP: 192.200.96.21
Target IP: **192.200.96.23**

ARP Reply

Sender MAC: **A3-B0-21-A1-60-35**
Sender IP: 192.200.96.23
Target MAC: 01-23-45-67-89-AB
Target IP: 192.200.96.21



IP: 192.200.96.23
MAC: A3-B0-21-A1-60-35

- ◊ La requête de la machine «192.200.96.21» demande l'@MAC de la machine 192.200.96.23;
- ◊ La réponse de la machine 192.200.96.23 donne la réponse @MAC A3:B0:21:A1:60:35.

Le protocole RARP «Reverse Address Resolution Protocol»

Il réalise l'opération inverse :

- une machine sans adresse IP connue peut envoyer une requête RARP pour demander son adresse IP.
- une machine particulière (un serveur gérant le réseau) lui répond et lui affecte son adresse IP.
- cette machine dispose d'une table de correspondance : (adresse physique, adresse IP).

Le protocole RARP est utile pour amorcer une station sans disque, un Terminal X-Window, ou une imprimante.

Pour consulter la table ARP**Sous Linux**

```
— xterm —  
pef@solaris:~$ ip neighbor  
192.168.42.254 dev eth0 lladdr 00:07:cb:cc:d4:e5 STALE  
192.168.42.122 dev eth0 lladdr 64:b9:e8:d2:23:ba REACHABLE
```

Sous Windows

```
on C:\WINDOWS\system32\cmd.exe  
C:\Documents and Settings\PeF>arp -a  
Interface : 192.168.144.128 --- 0x20002  
Adresse Internet      Adresse physique      Type  
192.168.144.254      00-50-56-e5-45-36    dynamique  
C:\Documents and Settings\PeF>
```



Pour envoyer un datagramme d'une source vers une destination, il faut savoir **localiser** la machine destination.

Deux possibilités :

- ▷ les deux machines **font partie** du même réseau local : on parle de **routage direct** (sur Ethernet, on utilisera le protocole ARP et l'envoi direct sur le réseau à diffusion) ;
- ▷ les deux machines **ne font pas partie** du même réseau local : on parle de **routage indirect**.

On doit passer par un intermédiaire qui permet de sortir du réseau local pour aller vers l'extérieur : le **routeur** (ou appelé «passerelle» ou *gateway*).

Pour faire du routage direct ou indirect pour un datagramme

- ▷ savoir si les deux machines font **partie du même réseau** local ;
- ▷ connaître l'@IP d'un **routeur de sortie** ;
- ▷ remettre **directement** le datagramme à la machine destination si elles sont dans le même réseau locale ;
- ▷ remettre le datagramme **au routeur sinon**.

Comment savoir si Source et Destination sont dans le même réseau ?

Il faut **comparer** l'<id. réseau> des deux adresses : si c'est **la même** \Rightarrow les deux sont dans le **même réseau local**.

Comment remettre le datagramme au routeur

Il faut utiliser le mécanisme **d'encapsulation** d'un datagramme dans une trame :

- ▷ la **trame** sert à remettre des données d'une machine connectée à un réseau local à une autre machine connectée au même réseau local ;
- ▷ la **trame** possède une @MAC de destination **indépendante** de l'@IP : il est possible d'envoyer la trame à une machine dont l'@IP **ne correspond pas** à son @IP !

Par exemple : on peut envoyer une datagramme à destination de l'extérieur du réseau local à l'@MAC du routeur.

Attention : les attaques MiTM, «Man-in-the-Middle», opèrent sur l'association @MAC \Leftrightarrow @IP du routeur !



Localisation de la machine destinataire

Chaque ordinateur connecté au réseau Internet dispose :

- ▷ d'une @IP ;
- ▷ d'un **masque de sous-réseau** (indiquant la répartition des 32 bits d'@IP entre <id. réseau><id. machine>).

Lors de l'envoi d'un paquet de S à destination d'une machine D, l'algorithme de routage est :

- * combinaison avec l'opérateur binaire «ET», &, de l'@IP de S et du masque de sous-réseau de S ;

164.81.128.34	10100100	01010001	10000000	00100010
255.255.192.0	11111111	11111111	11000000	00000000

ET

- * combinaison avec l'opérateur binaire «ET», &, de l'@IP de D et du masque de sous-réseau de S ;

164.81.128.0	10100100	01010001	10000000	00000000
--------------	----------	----------	----------	----------

- * comparaison entre ces deux valeurs :

- ◊ Si $Masque(S) \& @IP(S) \equiv Masque(S) \& @IP(D)$ alors **envoi en direct** sur le réseau local
- ◊ Sinon c'est un **routage indirect** : envoi par l'**intermédiaire d'un routeur de sortie** du réseau local.

Nouvelle version : la notion de préfixe

Actuellement, on ne tient plus compte de la notion de classe (CIDR : *Classless Inter-Domain Routing*), on indique la répartition <id. réseau><id. machine> directement à l'aide d'un «/n».

Exemple: 164.81.128.34/18 où /18 indique l'affectation des 18 premiers bits de l'adresse pour indiquer le réseau (notation: 255.255.192.0 pour un masque).

D'où: Si $prefixe(taille_S, @IP(S)) \equiv prefixe(taille_S, @IP(D)) \Rightarrow routage direct$

Mise en œuvre du routage direct

Utilisation du réseau à diffusion et donc de la fameuse @MAC et du protocole ARP...

Mise en œuvre du routage indirect

- ▷ il faut connaître l'adresse d'un routeur de sortie ;
- ▷ il faut remettre le datagramme à ce routeur en routage direct.



Sous Linux

Commandes ip address & ip link

permet de connaître la configuration des interfaces actives : carte réseau, pont (bridge), interface virtuelle...

```
pef@solaris:~$ ip link show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff

pef@solaris:~/RESEAUX_I/FIREWALL$ ip address show eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast
state UP qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff
        inet 192.168.42.83/24 brd 192.168.42.255 scope global eth0
            inet6 fe80::211:deff:fead:beef/64 scope link
                valid_lft forever preferred_lft forever
```

Pour :

* activer une interface :

```
sudo ip link set dev eth0 up
```

* désactiver :

```
sudo ip link set dev eth0 down
```

* enlever les adresses IP associées à une interface :

```
sudo ip address flush dev eth0
```

Commande ip route

```
pef@solaris:~$ ip route
137.204.212.128/25 dev bridge_dmz proto kernel scope link src 137.204.212.129
137.204.212.0/25 dev bridge_internal proto kernel scope link src 137.204.212.1
192.168.42.0/24 dev eth0 proto kernel scope link src 192.168.42.83 metric 1
169.254.0.0/16 dev eth0 scope link metric 1000
default via 192.168.42.254 dev eth0 proto static
```

On remarque que la route par défaut pointe vers 192.168.42.254, c-à-d que tout datagramme qui n'est pas destiné à un réseau connu de la table de routage sera transmis vers le routeur associé à cette adresse.



Sous Windows

Commande ipconfig

```
ex C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\PeF>ipconfig /all

Configuration IP de Windows

    Nom de l'hôte . . . . . : macbookpro
    Suffixe DNS principal . . . . . : 
    Type de nœud . . . . . : Inconnu
    Routage IP activé . . . . . : Non
    Proxy WINS activé . . . . . : Non
    Liste de recherche du suffixe DNS : localdomain

Carte Ethernet Connexion au réseau local 2:

    Suffixe DNS propre à la connexion : localdomain
    Description . . . . . : VMware Accelerated AMD PCNet Adapter
#2
    Adresse physique . . . . . : 00-0C-29-7E-4A-1E
    DHCP activé . . . . . : Oui
    Configuration automatique activée . . . . . : Oui
    Adresse IP . . . . . : 192.168.144.128
    Masque de sous-réseau . . . . . : 255.255.255.0
    Passerelle par défaut . . . . . : 192.168.144.2
    Serveur DHCP . . . . . : 192.168.144.254
    Serveurs DNS . . . . . : 192.168.144.2
    Bail obtenu . . . . . : mercredi 12 décembre 2007 11:22:16
    Bail expirant . . . . . : mercredi 12 décembre 2007 11:52:16

C:\Documents and Settings\PeF>
```

Commande route print

```
ex C:\WINDOWS\system32\cmd.exe
=====
C:\Documents and Settings\PeF>route print
=====
Liste d'Interfaces
0x1 . . . . . MS TCP Loopback interface
0x20002 . . . . . 00 0c 29 7e 4a 1e . . . . . Carte AMD PCNET Family Ethernet PCI #2 - Min
import d'ordonnancement de paquets
=====

Itinéraires actifs :
Destination réseau   Masque réseau   Adr. passerelle   Adr. interface   Métrique
          0.0.0.0       0.0.0.0      192.168.144.128  192.168.144.128      10
          127.0.0.0      255.0.0.0     127.0.0.1       127.0.0.1        1
          192.168.144.0   255.255.255.255  192.168.144.128  192.168.144.128      10
          192.168.144.128  255.255.255.255  192.168.144.128  192.168.144.128      10
          192.168.144.255  255.255.255.255  192.168.144.128  192.168.144.128      10
          224.0.0.0       240.0.0.0      192.168.144.128  192.168.144.128      10
          255.255.255.255 255.255.255.255  192.168.144.128  192.168.144.128      1
Passerelle par défaut :      192.168.144.2
=====

Itinéraires persistants :
Hucun
=====

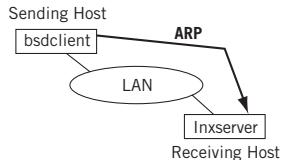
C:\Documents and Settings\PeF>
```



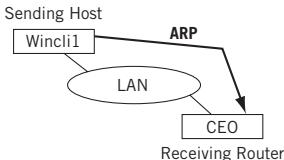
Routage & ARP : Quatre cas de figure

78

Case 1: Find the address of a host on the same subnet as the source.



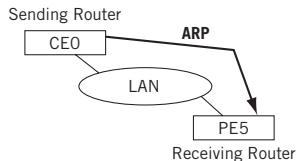
Case 2: Find the address of a router on the same subnet as the source.



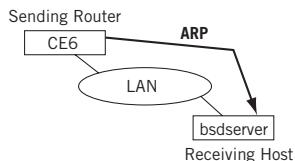
1. **hôte vers hôte**: l'émetteur est un hôte qui veut envoyer un paquet à un autre hôte dans le même réseau local.

Dans ce cas, l'@IP de destination est connue et l'@MAC de destination doit être trouvée.

Case 3: Find the address of a router on the same subnet as the source router.



Case 4: Find the address of a host on the same subnet as the source router.



2. **hôte vers routeur**: l'émetteur est un hôte et veut envoyer un paquet à un autre hôte **n'appartenant pas** au réseau local.

Il doit consulter la table de routage, «*forwarding table*» ou «*Forwarding Information Base*», pour trouver l'@IP du routeur.

L'@IP du routeur est connue et l'@MAC du routeur doit être trouvée.

3. **routeur vers routeur**: l'émetteur est un routeur et veut «faire suivre» un paquet à un autre routeur connecté dans le même réseau local.

La table de routage est utilisée pour trouver l'@IP du routeur.

L'@IP du routeur est connue et l'@MAC du routeur destination doit être trouvée.

4. **routeur vers hôte**: l'émetteur est un routeur et veut «faire suivre», «*forward*», un paquet vers un hôte dans le même réseau local.

L'@IP de l'hôte est connue, elle est contenue dans le paquet et l'@MAC de l'hôte doit être trouvée.

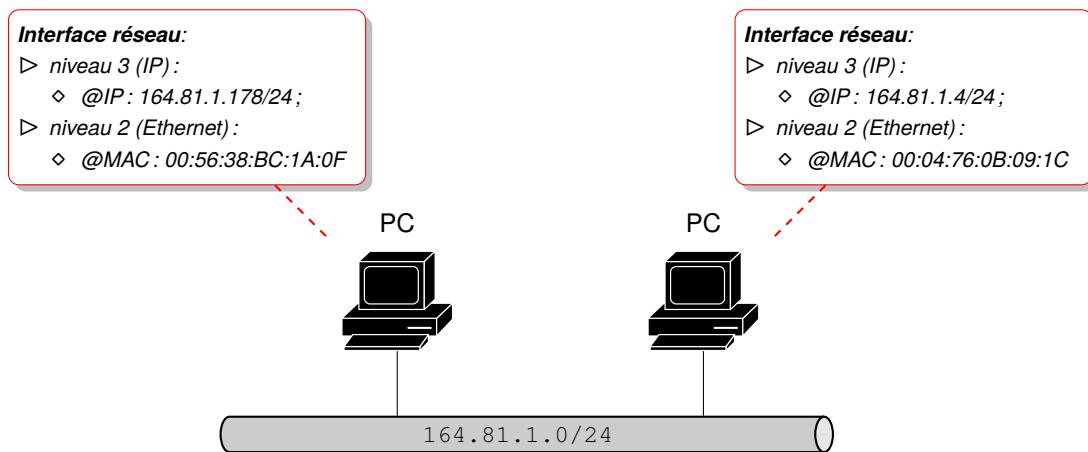
Attention

«...dans le même réseau local» ! \Rightarrow ARP ne sert quand dans un réseau local.



Chaque machine est identifiée par :

- * une adresse de niveau 2 (@MAC) ;
- * une adresse de niveau 3 (@IP) ;
- * un réseau d'appartenance connu :
 - ◊ à l'aide du **préfixe** «/n» indiquant n le nombre de bits de l'identifiant réseau»
 - ◊ ou à l'aide du **masque réseau**, *netmask*, adresse où chaque bit de l'identifiant réseau est à 1, les autres sont à 0.

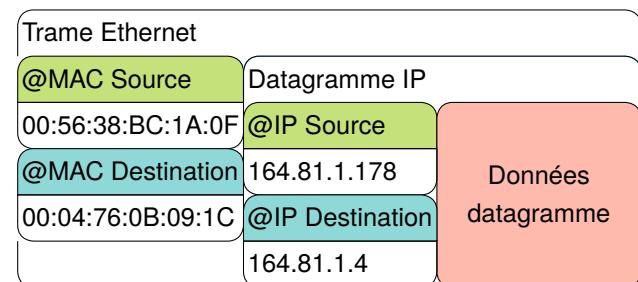
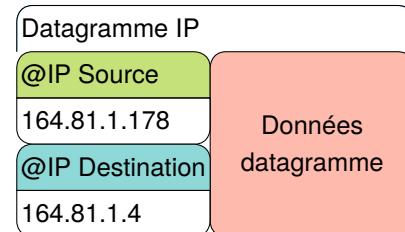
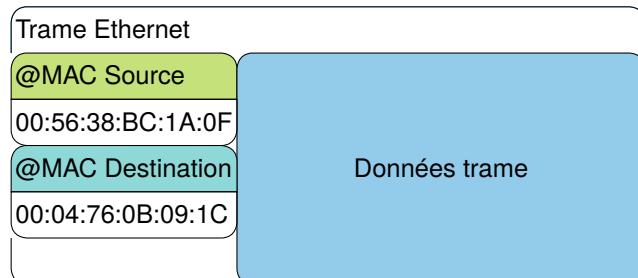


Pour être échangé, les datagrammes IP sont encapsulés dans des trames Ethernet

- * la **trame** contient :
 - ◊ une @MAC source ;
 - ◊ une @MAC de destination ;

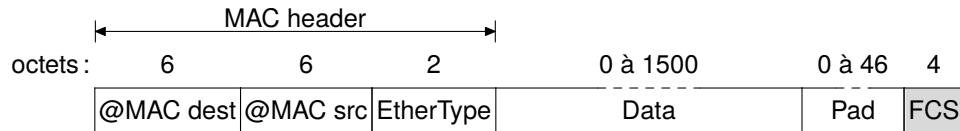
- * le **datagramme** contient :
 - ◊ une @IP source ;
 - ◊ une @IP destination et des **données** ;

- * la **trame encapsule** le datagramme IP :



Le réseau à diffusion utilise une technologie différente du réseau IP :

- * il dispose de **ses propres adresses** : @MAC ;
- * il utilise des messages sous un **format particulier** : la trame Ethernet ou IEEE 802.3 :



Où :

- ◊ **@MAC destination puis @MAC source** : ⇒ le récepteur détermine immédiatement s'il est destinataire ;
- ◊ **EtherType** (valeur > 1536) :
 - * 0x0800 pour un datagramme IPv4, 0x08DD pour un datagramme IPv6, 0x806 pour un message ARP, 0x888E pour du 802.1x, 0x8100 pour du VLAN...;
 - * **mais, si la valeur < 1500 ⇒ trame 802.3 avec LLC**, «Logical Link Control» ;
- ◊ **Data** : les données de la trame, complétées éventuellement par du bourrage (la longueur de ce champ est comprise entre 0 et 1500 octets) ;
- ◊ **PAD** ou bourrage : octets de bourrage sans signification, insérés **si la longueur du champ Data est insuffisante** (inférieure à 46 octets) ;
- ◊ **FCS**, *frame check sequence* ou Checksum : champ pour la **détection d'erreurs** (rarement transmis aux programmes).

Attention

- * Si le FCS est incorrect alors la trame n'est **pas transmise** par la carte réseau au système d'exploitation.
- * La trame a une taille de 64 à 1518 octets ($6 + 6 + 2 + 46 + 4 = 64$ à $6 + 6 + 2 + 1500 + 4 = 1518$).

Encapsulation en technologie Ethernet vu dans Wireshark

82

Un datagramme IP est contenu dans une trame, par exemple une trame Ethernet :

The screenshot shows a Wireshark capture of an Ethernet frame (Frame 7) with the following details:

- Ethernet II:** Src: AppleCom_d6:d3:b9 (00:19:e3:d6:d3:b9), Dst: HonHaiPr_d7:a9:d2 (00:16:ce:d7:a9:d2)
- Type: IP (0x0800)
- Frame check sequence: 0xa2275cc2 [correct]
- Internet Protocol:** Src: 192.168.1.51 (192.168.1.51), Dst: 88.87.11.119 (88.87.11.119)
- User Datagram Protocol:** Src Port: 64617 (64617), Dst Port: 51534 (51534)
- Data (71 bytes)

The data bytes are displayed in three panes:

- Hex View:** Shows the raw bytes of the frame, including the Ethernet header (00:16:ce:d7:a9:d2, 00:19:e3:d6:d3:b9), the IP header (40 bytes), and the UDP header (12 bytes). The payload data starts at byte 74.
- ASCII View:** Displays the ASCII representation of the captured data, showing the structure of the Ethernet frame, IP header, and UDP header, followed by the data payload.
- Text View:** Provides a detailed breakdown of the captured data, identifying the frame type, source and destination MAC addresses, IP addresses, port numbers, and the actual data payload.

Le datagramme IP encapsule lui-même un datagramme UDP...

Routage direct illustré

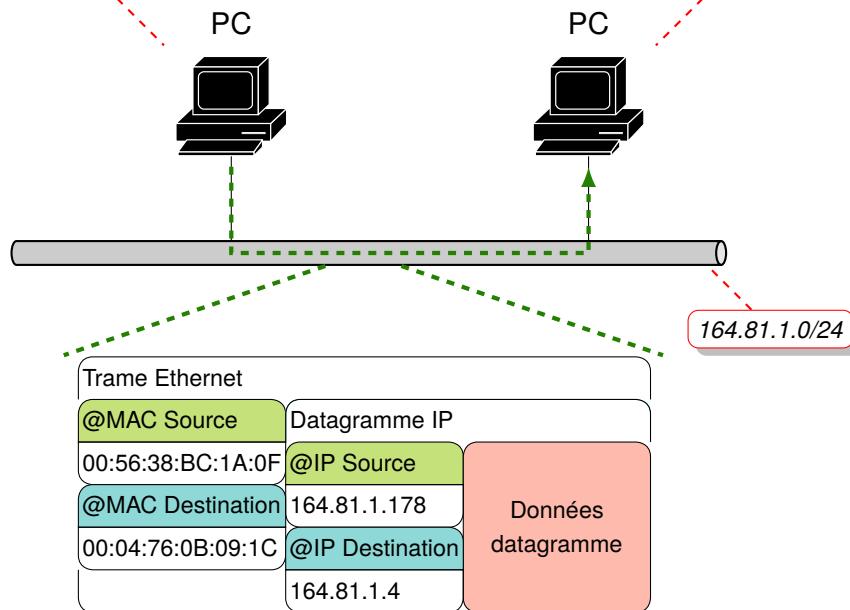
83

Interface réseau:

- ▷ niveau 3 (IP):
 - ◊ @IP: 164.81.1.178/24;
- ▷ niveau 2 (Ethernet):
 - ◊ @MAC : 00:56:38:BC:1A:0F

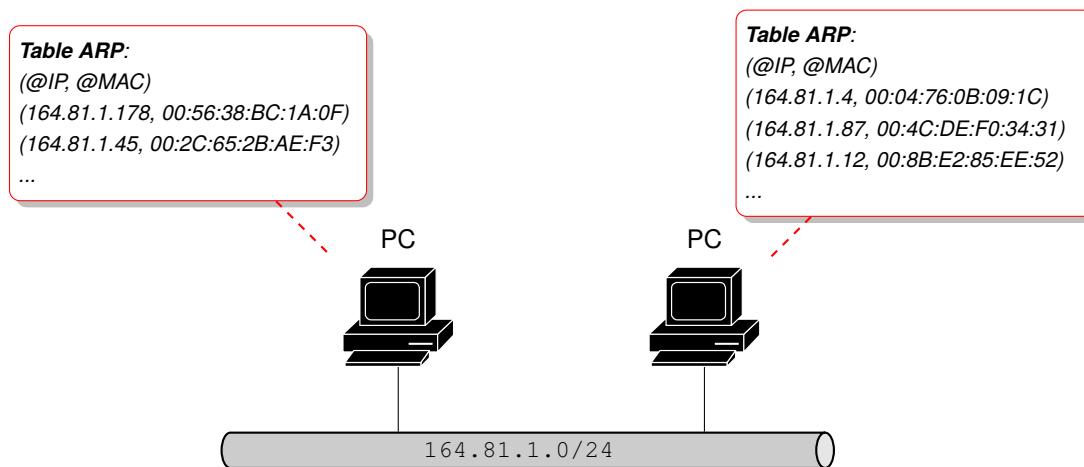
Interface réseau:

- ▷ niveau 3 (IP):
 - ◊ @IP: 164.81.1.4/24;
- ▷ niveau 2 (Ethernet):
 - ◊ @MAC : 00:04:76:0B:09:1C



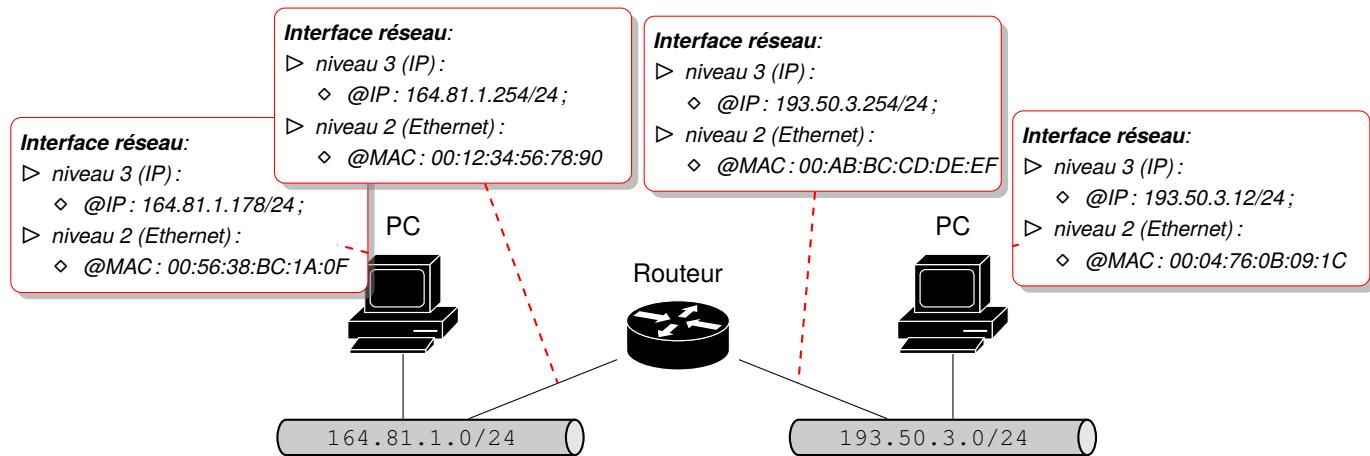
Pour connaître la correspondance entre adresse IP et adresse MAC :

- ▷ mise en oeuvre du protocole ARP (Address Resolution Protocol) ;
- ▷ construction d'une table de correspondance entre @ IP et MAC sur chaque machine (cache ARP).



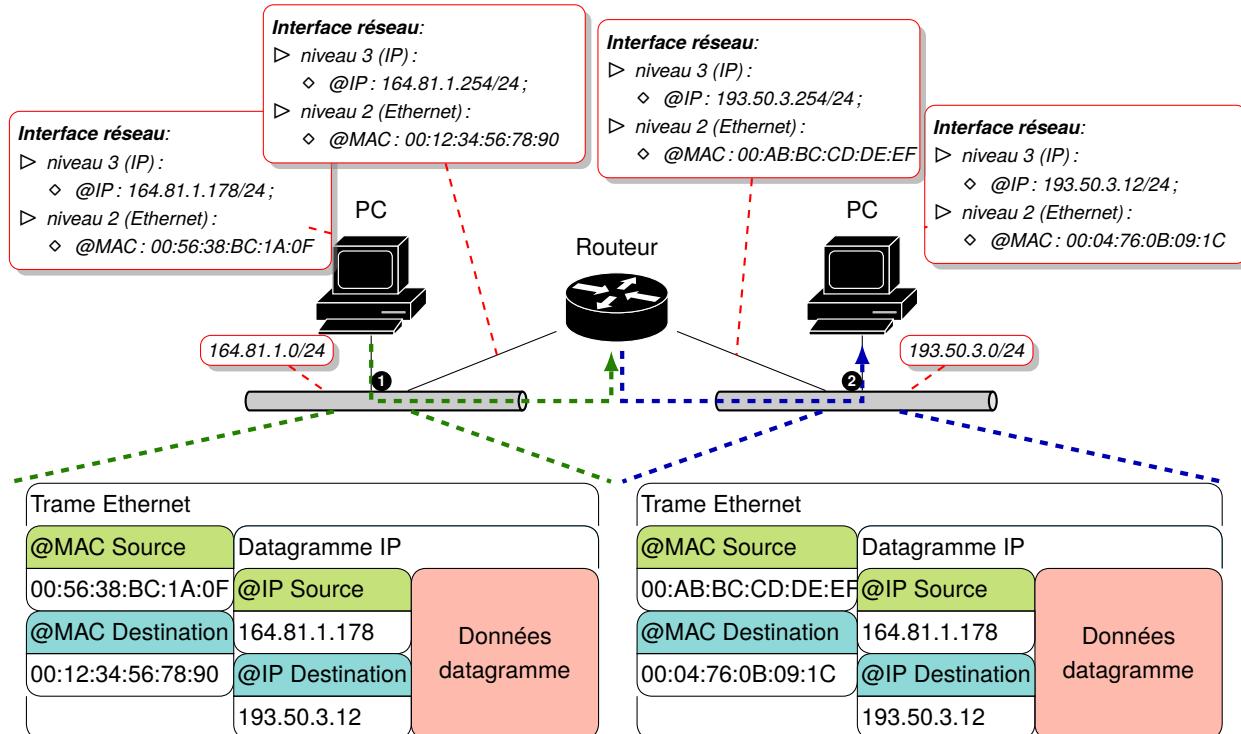
La **modification malveillante** de cette table est possible : «ARP Spoofing» (usurpation d'identité), «ARP Cache Poisoning» (insertion d'association erronée).

Le paquet de la machine 164.81.1.178 est routé par l'intermédiaire du routeur vers la machine 193.50.3.12.



Le datagramme IP est **encapsulé**:

- ▷ par la machine 164.81.1.178, dans une trame à **destination du routeur**;
- ▷ puis, par le routeur, dans une nouvelle trame à destination de la machine 193.50.3.12.



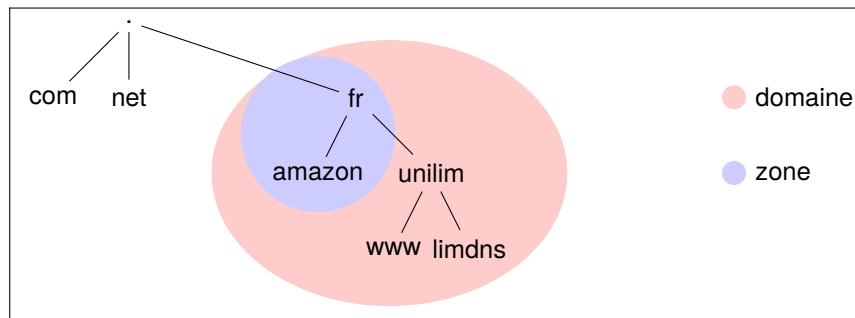
L'encapsulation permet la redirection vers le routeur sans modifier les @IP du datagramme.

Le système DNS est entièrement distribué au niveau planétaire en utilisant la **délégation de domaine**.

À tout domaine est associé une **responsabilité administrative**.

Une organisation responsable d'un domaine peut :

- ▷ **découper** le domaine en sous-domaines ;
- ▷ **déléguer** les sous-domaines à d'autres organisations :
 - ◊ qui deviennent responsables du (des) sous-domaine(s) qui leurs sont délégué(s) peuvent, à leur tour, déléguer des sous-domaines des sous-domaines qu'elles gèrent.
 - ◊ *Le domaine parent contient alors seulement un pointeur vers le sous-domaine délégué;*
- * Les serveurs de nom enregistrent les données propres à une partie de l'espace nom de domaine dans une **zone**.
- * le serveur de nom à **autorité administrative** sur cette zone ;
- * un serveur de nom peut avoir **autorité** sur plusieurs zones ;
- * une **zone** contient les informations d'un domaine **sauf** celles qui sont **déléguées** :



Whois «unilim.fr»

```
darkstar:~ pef$ whois unilim.fr
%%
%% This is the AFNIC Whois server.
%%
%% complete date format : DD/MM/YYYY
%% short date format     : DD/MM
%% version                : FRNIC-2.5
%%
%% Rights restricted by copyright.
%% See http://www.apnic.net/apnic/web/mentions-legales-whois_en
%%
%% Use '-h' option to obtain more information about this service.
%%
%% [2a01:0e35:8a71:bec0:66b9:e8ff:fed2:23ba REQUEST] >> unilim.fr
%%
%% RL Net [#####] - RL IP [#####]
%%
domain:      unilim.fr
status:      ACTIVE
hold:        NO
holder-c:    UDL3-FRNIC
admin-c:     JPL1325-FRNIC
tech-c:      GRST1-FRNIC
tech-c:      NV70-FRNIC
tech-c:      GU245-FRNIC
zone-c:      NFC1-FRNIC
ns1-id:      NSL5796-FRNIC
registrar:   GIP RENATER
Expiry Date: 01/01/2016
created:     01/01/1995
last-update: 15/12/2014
source:      FRNIC

ns-list:     NSL5796-FRNIC
nserver:    limdns.unilim.fr [164.81.1.4]
nserver:    limdns2.unilim.fr [164.81.1.5]
nserver:    cnudns.cines.fr [193.48.169.40 2001:660:6301:301::2:1]
source:      FRNIC

registrar:   GIP RENATER
type:        Isp Option 1
address:    23-25 Rue Daviel
address:    PARIS
country:    FR
phone:      +33 1 53 94 20 30
fax-no:     +33 1 53 94 20 31
e-mail:     domaine@renater.fr
website:   http://www.renater.fr
anonymous:  NO
registered: 01/01/1998
source:      FRNIC

nic-hdl:    JPL1325-FRNIC
type:        PERSON
contact:   Jean-Pierre Laine
address:  Unvi. de Limoges
address:  123, Avenue Albert Thomas
address:  87060 Limoges
country:    FR
phone:      +33 5 55 45 77 08
e-mail:     jean-pierre.laine@unilim.fr
registrar:  GIP RENATER
changed:   24/10/2013 nic@nic.fr
anonymous:  NO
obsoleted:  NO
source:      FRNIC
```

Le responsable du domaine : Jean-Pierre Laine



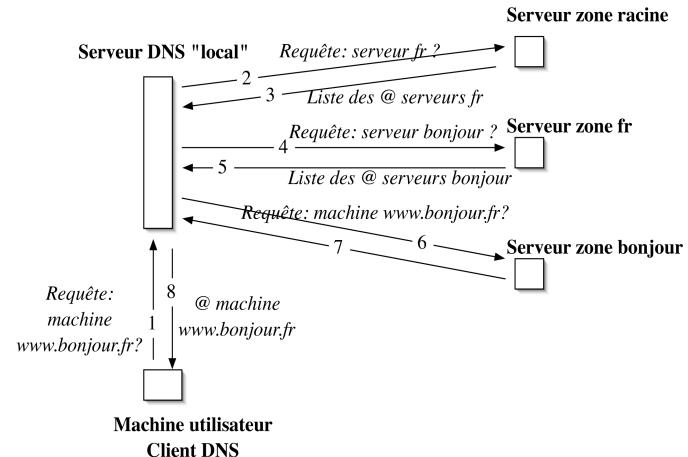
Modèle de fonctionnement client / serveur

- l'utilisateur utilise un nom de domaine dans une application (exemple : ping www.bonjour.fr) ;
- l'application cliente demande la **traduction du nom de domaine** auprès d'un serveur de nom (DNS)
 - ◇ si le serveur connaît la réponse il répond ;
 - ◇ sinon,
 - * s'il **fait autorité** pour le domaine demandé, alors pas de réponse (la machine n'existe pas) ;
 - * s'il ne fait pas autorité, le serveur de nom **interroge d'autres serveurs de nom** (de plus grand suffixe commun) jusqu'à ce que l'association nom de domaine / adresse IP soit trouvée ;
 - ◇ le serveur de nom retourne l'adresse IP au logiciel client : @IP de la machine «www» (il mémorise la réponse dans un cache et pour une certaine durée) ;

Le logiciel client contacte le serveur, comme si l'utilisateur avait spécifié une adresse IP.

Exemple :

```
xterm
$ telnet ishtar.msi.unilim.fr
connexion établie avec 164.81.60.43
```



Resolver

Les «resolvers» sont les processus clients qui contactent les serveurs de nom Fonctionnement :

- ◊ contacte un « name server » (dont l'(les) adresse(s) est (sont) configurées sur la machine exécutant ce resolver) ;
- ◊ interprète les réponses et retourne l'information au logiciel appelant ;
- ◊ gère un cache (dépend de la mise en oeuvre).

Le serveur de nom interroge également d'autres serveurs de nom, lorsqu'il n'a pas autorité sur la zone requise (fonctionnement **itératif** ou **récursif**).

Si le serveur de nom est en dehors du domaine requis, il peut être amené à contacter un serveur racine.

Amélioration des performances

Mécanisme de cache dans les serveurs pour limiter le nombre d'interrogations

- ◊ évite la surcharge du réseau ;
- ◊ diminue les délais de réponse ;
- ◊ baisse la charge des serveurs de haut niveau (les serveurs racines).

Remplissage du cache lors des requêtes des clients et durée de vie limitée dans le cache

- ◊ TTL (Time To Live) spécifié dans les réponses pour éviter qu'une association soit conservée trop longtemps.

Types de serveur de nom

Serveur de nom **primaire** :

- ◊ maintient la base de données de la zone dont il a l'**autorité administrative**

Serveur de nom **secondaire** :

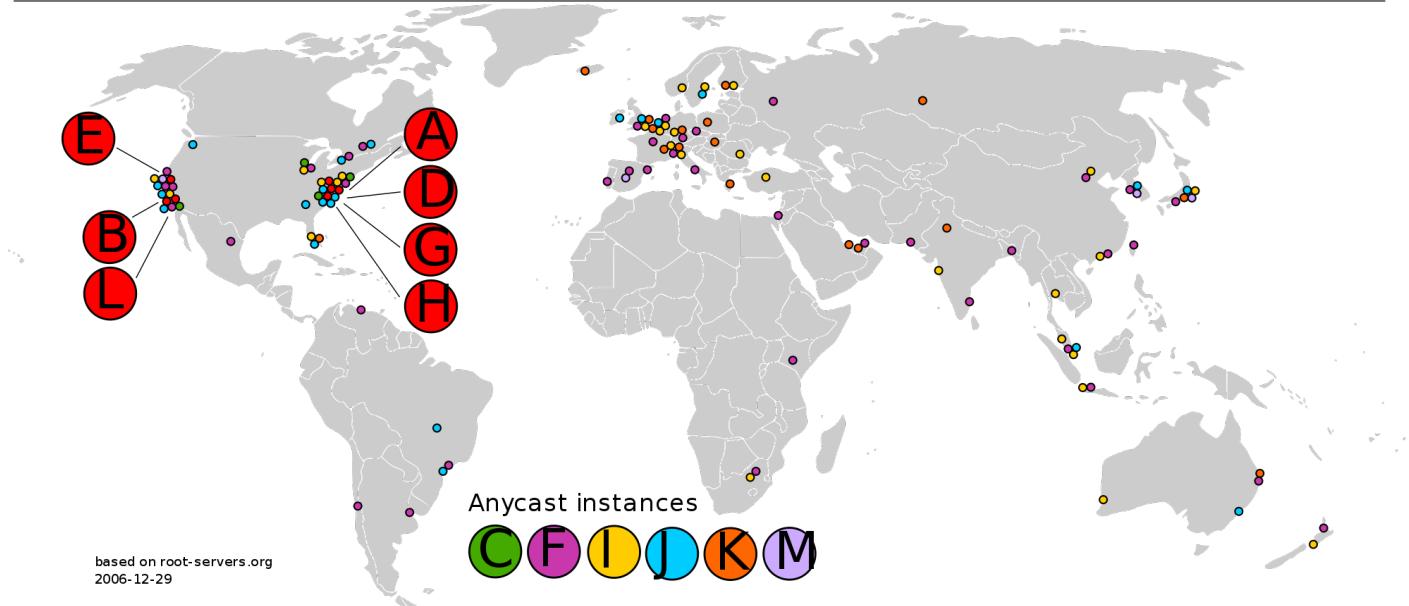
- ◊ interroge périodiquement le serveur de nom primaire et met à jour les données

Il y a **un** serveur primaire et généralement **plusieurs** secondaires.

La redondance permet la défaillance éventuelle du primaire et du (des) secondaire(s)

Un serveur de nom peut être primaire pour une (des) zone(s) et secondaire pour d'autre(s).





Le terme «anycast» permet d'offrir des services DNS de proximité à l'aide de cette capacité offerte par IPv6.

On peut remarquer que ce service de proximité permet même de délocaliser géographiquement les serveurs et améliore la disponibilité et la sécurité du système DNS.

Exemple : une requête DNS vers «google.com»

```

xterm
bonnefoi@msi:~$ dig +trace @164.81.1.5 www.google.com
; <>> DiG 9.7.3 <>> +trace @164.81.1.5 www.google.com
; (1 server found)
;; global options: +cmd
.          IN NS j.root-servers.net.
.          IN NS g.root-servers.net.
.          IN NS l.root-servers.net.
.          IN NS k.root-servers.net.
.          IN NS c.root-servers.net.
.          IN NS f.root-servers.net.
.          IN NS e.root-servers.net.
.          IN NS m.root-servers.net.
.          IN NS b.root-servers.net.
.          IN NS d.root-servers.net.
.          IN NS h.root-servers.net.
.          IN NS a.root-servers.net.
.          IN NS i.root-servers.net.
;; Received 272 bytes from 164.81.1.5#53(164.81.1.5) in 1 ms
com.        172800  IN NS a.gtld-servers.net.
com.        172800  IN NS b.gtld-servers.net.
com.        172800  IN NS c.gtld-servers.net.
com.        172800  IN NS d.gtld-servers.net.
com.        172800  IN NS e.gtld-servers.net.
com.        172800  IN NS f.gtld-servers.net.
com.        172800  IN NS g.gtld-servers.net.
com.        172800  IN NS h.gtld-servers.net.
com.        172800  IN NS i.gtld-servers.net.
com.        172800  IN NS j.gtld-servers.net.
com.        172800  IN NS k.gtld-servers.net.
com.        172800  IN NS l.gtld-servers.net.
com.        172800  IN NS m.gtld-servers.net.
;; Received 492 bytes from 193.0.14.129#53(k.root-servers.net) in 31 ms
google.com. 172800  IN NS ns2.google.com.
google.com. 172800  IN NS ns1.google.com.
google.com. 172800  IN NS ns3.google.com.
google.com. 172800  IN NS ns4.google.com.
;; Received 168 bytes from 192.12.94.30#53(e.gtld-servers.net) in 34 ms
www.google.com. 604800  IN CNAME www.l.google.com.
www.l.google.com. 300  IN A 209.85.227.147
www.l.google.com. 300  IN A 209.85.227.106
www.l.google.com. 300  IN A 209.85.227.103
www.l.google.com. 300  IN A 209.85.227.99
www.l.google.com. 300  IN A 209.85.227.105
www.l.google.com. 300  IN A 209.85.227.104
;; Received 148 bytes from 216.239.32.10#53(ns1.google.com) in 26 ms

```

- * la configuration de la machine msi.unilim.fr:

```

xterm
bonnefoi@msi:~$ more /etc/resolv.conf
search msi.unilim.fr unilim.fr
nameserver 164.81.60.1
nameserver 164.81.1.4
nameserver 164.81.1.5

```

- * la requête fait appel au serveur DNS 164.81.1.5 connu de la machine msi.unilim.fr, mais demandé **explicitement** dans l'utilisation de l'outil dig.
- * le serveur, *resolver*, 164.81.1.5 réalise une requête à différents «root servers»
- * il obtient une réponse de la part du serveur racine «k.root-servers.net» qui lui donne la liste des serveurs «GTLD», «Generic Top Level Domain», chargés du domaine «.com»;
- * le serveur «e.gtld-servers.net» lui renvoi la liste des serveurs DNS en charge du domaine google.com;
- * le serveur DNS «ns1.google.com» renvoi une liste d'adresse correspondant à la machine google.com.

Les serveurs racines présents dans le système d'exploitation

```

; Cache file:
. IN NS A.ROOT-SERVERS.NET.
A.ROOT-SERVERS.NET. IN A 198.41.0.4
. IN NS B.ROOT-SERVERS.NET.
B.ROOT-SERVERS.NET. IN A 128.9.0.107
. IN NS C.ROOT-SERVERS.NET.
C.ROOT-SERVERS.NET. IN A 192.33.4.12
. IN NS D.ROOT-SERVERS.NET.
D.ROOT-SERVERS.NET. IN A 128.8.10.90
. IN NS E.ROOT-SERVERS.NET.

```



Le serveur de courrier

MX = Mail eXchanger Permet l'adressage email sur la base du nom de domaine plutôt que sur l'adresse du (des) serveur(s) de mail :

- ◊ bonnefoi@unilim.fr plutot que bonnefoi@msi.unilim.fr;
- ◊ permet à l'émetteur d'ignorer quelle est la machine serveur de mail ;
- ◊ permet le déplacement du gestionnaire de mail vers une autre machine ;
- ◊ permet la gestion de plusieurs serveurs de mail avec priorité dans l'ordre de consultation des serveurs

L'enregistrement MX est utilisés par les MTA, «*Mail Transfer Agent*», en tenant compte des priorités :

```
██████ xterm ██████████
bonnefoi@msi:~$ dig +short mx unilim.fr
50 mail.unilim.fr.
```

```
██████ xterm ██████████
bonnefoi@msi:~$ dig +short mx google.com
30 alt2.aspmx.l.google.com.
40 alt3.aspmx.l.google.com.
50 alt4.aspmx.l.google.com.
10 aspmx.l.google.com.
20 alt1.aspmx.l.google.com.
```

Le MTA utilise le protocole **SMTP**, «*Simple Mail Transfer Protocol*», en tant que client.



Protéger les requêtes DNS de l'utilisateur d'un observateur extérieur en utilisant un échange basé sur **HTTPS** pour les réaliser :

- nécessite l'emploi de **serveurs supportant** ce protocole : non accepté par tous les serveurs DNS ;
- établie une connexion HTTPS basée sur TLS ou HTTP/2 (pour accélérer les requêtes suivantes) ;
- utilise les méthodes GET ou POST, comme par exemple, celle de Google:
 - ◊ `https://dns.google/dns-query` – RFC 8484 (GET and POST)
 - ◊ `https://dns.google/resolve?` – JSON API (GET)
- accepte des **requêtes** au format:
 - ◊ texte, avec une URL contenant l'entrée DNS demandée ;
 - ◊ binaire, reprenant le format de la requête DNS originale en l'encodant en «*URL safe base64*» ;
- retourne des **réponses** au format:
 - ◊ texte, basée sur JSON ;
 - ◊ binaire, correspondant au format original du DNS, appelé aussi «*wireformat*» ;

```
xterm
$ socat - openssl: dns.google.com: 443, verify=0
GET /resolve?name=www.unilim.fr&type=A HTTP/1.0
Host: dns.google.com
HTTP/1.0 200 OK
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Access-Control-Allow-Origin: *
Date: Mon, 16 Sep 2019 08:25:11 GMT
Expires: Mon, 16 Sep 2019 08:25:11 GMT
Cache-Control: private, max-age=131
Content-Type: application/x-javascript; charset=UTF-8
Server: HTTP server (unknown)
X-XSS-Protection: 0
X-Frame-Options: SAMEORIGIN
Alt-Svc: quic=":443"; ma=2592000; v="46, 43, 39"
Accept-Ranges: none
Vary: Accept-Encoding

{"Status": 0, "TC": false, "RD": true, "RA": true, "AD": false, "CD": false, "Question": [ {"name": "www.unilim.fr.", "type": 1}], "Answer": [ {"name": "www.unilim.fr.", "type": 5, "TTL": 131, "data": "web-proxy-2.unilim.fr."}, {"name": "web-proxy-2.unilim.fr.", "type": 1, "TTL": 131, "data": "164.81.1.97"}]}
```



- * Configuration de l'adresse IP et du réseau de connexion:

```
□ └── xterm
root@solaris:~# ip address flush dev eth0
root@solaris:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff
root@solaris:~# ip address add 192.168.42.57/24 brd + dev eth0
root@solaris:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff
        inet 192.168.42.57/24 brd 192.168.42.255 scope global eth0
root@solaris:~# ifconfig eth0
eth0      Link encap:Ethernet HWaddr 00:11:de:ad:be:ef
          inet adr:192.168.42.57 Bcast:192.168.42.255 Masque:255.255.255.0
          UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
          Packets reçus:2222687 erreurs:1723210 :7968 overruns:0 frame:0
          TX packets:915273 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 lg file transmission:1000
          Octets reçus:1148981328 (1.1 GB) Octets transmis:807289372 (807.2 MB)
          Interruption:19 Adresse de base:0x2024
```

- * Configuration de la **route par défaut**:

```
□ └── xterm
root@solaris:~# ip route add default via 192.168.42.254
root@solaris:~# ip route
192.168.42.0/24 dev eth0 proto kernel scope link src 192.168.42.57
default via 192.168.42.254 dev eth0
```

- * Tester le routage et trouver l'**interface de sortie** (celle considérée comme «*default*»):

```
□ └── xterm
root@solaris:~# ip route get 164.81.1.4
164.81.1.4 via 192.168.42.254 dev eth0 src 192.168.42.57
cache
```

Il ne manquera plus qu'à configurer la résolution DNS avec le fichier /etc/resolv.conf.



- * On édite le contenu de /etc/network/interfaces

```
□ — xterm —
root@solaris:~# more /etc/network/interfaces
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet static
address 192.168.42.57
netmask 255.255.255.0
gateway 192.168.42.254
```

- * le fichier /etc/resolv.conf:

```
□ — xterm —
root@solaris:~# more /etc/resolv.conf
domain pefnet
search pefnet
nameserver 192.168.42.53
```

Le «search» indique le nom de domaine à ajouter au nom d'une machine entrée sans le FQDN (exemple: msi au lieu de msi.unilim.fr)

- * on relance le «service» réseaux :

```
□ — xterm —
root@solaris:~# /etc/init.d/networking restart
```



- * configuration de l'interface pour utiliser le client DHCP, «*Dynamic Host*», dans le fichier /etc/network/interfaces :

```
auto lo
iface lo inet loopback

auto eth0
iface eth0 inet dhcp
```

- * en ligne de commande :

```
└── xterm ━━━━━━
$ sudo dhclient eth0
```

Le client DHCP configure l'interface, le fichier `resolv.conf` et la route par défaut si un serveur DHCP prend en charge sa demande.

```
└── xterm ━━━━━━
root@solaris:~# ip address show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
    link/ether 00:11:de:ad:be:ef brd ff:ff:ff:ff:ff:ff
    inet 192.168.42.83/24 brd 192.168.42.255 scope global eth0

root@solaris:~# ip route
192.168.42.0/24 dev eth0  proto kernel  scope link  src 192.168.42.83
default via 192.168.42.254 dev eth0
```

