Version : **2020.01**

Dernière mise-à-jour : 2020/01/30 03:28

SO214 - Gestion des Services Réseau TCPv4

Configuration de TCP/IP

La configuration TCP/IP sous Solaris s'effectue en éditant plusieurs fichiers :

Fichier	**Description**
/etc/resolv.conf	Spécifie les serveurs DNS
/etc/hosts	Spécifie des binômes nom-adresse IP pour la résolution locale
/etc/nsswitch.conf	Spécifie l'ordre des services de résolution de noms
/etc/nodename	Nom de la machine
/etc/hostname.nic	Nom ou adresse IP associé(e) à la carte *nic*
/etc/defaultdomain	Nom du domaine NIS / NIS+
/etc/defaultrouter	Spécifie le routeur par défaut
/etc/inet/netmasks	Spécifie le masque réseau

Le configuration TCP/IP peut être effectuée en utilisant la commande **sys-unconfig**. Cette commande redémarre le système et permet de configurer les éléments suivants :

- Nom et adresse de la machine
- L'utilisation de NIS ou pas
- Le service de résolution de noms à utiliser
- Le mot de passe de root
- La date, l'heure et le lieu du système.

Important - Tapez la commande **sys-unconfig** pour configurer votre système en DHCP puis en adresse fixe. Notez que le FQDN de votre système doit être **solaris.i2tch.loc** et que l'adresse IP est la **10.0.2.15**. L'adresse de la passerelle est la **10.0.2.2** tandis que les DNS sont **8.8.8.8**, **8.8.4.4** et **10.0.2.3**.

Configuration manuelle de l'adresse IP

DHCP

Pour configurer un poste en DHCP, il convient de :

- supprimer le fichier **/etc/nodename** s'il existe,
- supprimer le fichier **/etc/defaultrouter** s'il existe,
- créer le fichier **vide /etc/hostname.nic** où *nic* est le nom de la carte réseau obtenu en utilisant la commande **netstat - i**,
- éditer le fichier **/etc/hosts** en y inscrivant **uniquement** l'adresse loopback.

Par exemple :

```
# rm -f /etc/nodename
# rm -f /etc/defaultrouter
# netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      353    0     353    0     0      0
e1000g0 1500 unknown       unknown        3157   0     2641   0     0      0
# rm -f /etc/hostname.e1000g0
# touch /etc/hostname.e1000g0
# cat  /etc/hostname.e1000g0
#
# echo "127.0.0.1       localhost" > /etc/hosts
# cat /etc/hosts
127.0.0.1       localhost
```

```
# reboot
```

Adresse IP en fixe

Afin d'affecter une adresse IP fixe à votre système, il est nécessaire d'éditer les fichiers suivants :

- /etc/nodename
  - ce fichier contient le FQDN du système
- /etc/hostname.nic
  - ce fichier contient l'adresse IP fixe de la carte réseau
- /etc/hosts
  - ce fichier contient la correspondance adresse IP - FQDN
- /etc/defaultdomain
  - ce fichier contient le FQDN du système
- /etc/defaultrouter
  - ce fichier contient l'adresse IP de la passerelle par défault
- /etc/resolv.conf
  - ce fichier contient les adresses IP des serveurs DNS
- /etc/nsswitch.conf
  - ce fichier stipule l'ordre de recherche des services de noms

Par exemple :

```
# vi /etc/nodename
# cat /etc/nodename
solaris.i2tch.loc
# vi /etc/hostname.e1000g0
# vi /etc/hostname.e1000g0
10.0.2.15
# vi /etc/hosts
# cat /etc/hosts
127.0.0.1       localhost unknown unknown.
10.0.2.15       solaris.i2tch.loc
```

```
# cat /etc/defaultdomain
solaris.i2tch.loc
# vi /etc/defaultrouter
# cat /etc/defaultrouter
10.0.2.2
# cat /etc/resolv.conf
nameserver 10.0.2.3
# grep '^hosts:' /etc/nsswitch.conf
hosts:      files dns
# grep '10' /etc/inet/netmasks
10.0.2.0     255.255.255.0
# reboot
```

Résolution d'adresses IP

La configuration DNS est stockée dans le fichier **/etc/resolv.conf** :

/etc/resolv.conf

```
# cat /etc/resolv.conf
nameserver 10.0.2.3
```

L'ordre de recherche des services de noms est stocké dans le fichier **/etc/nsswitch.conf**. Pour connaître l'ordre, saisissez la commande suivante :

```
# grep '^hosts:' /etc/nsswitch.conf
hosts: files dns # Added by DHCP
```

Le mot **files** fait référence au fichier **/etc/hosts** :

/etc/hosts

Le fichier /etc/hosts est un lien vers le fichier **/etc/inet/hosts**.

```
# cat /etc/hosts
127.0.0.1       localhost
10.0.2.15       solaris.i2tch.loc       # Added by DHCP
```

Pour tester le serveur DNS, deux commandes sont possibles :

```
# nslookup www.ittraining.center
Server:         10.0.2.3
Address:        10.0.2.3#53

Non-authoritative answer:
Name:   www.ittraining.center
Address: 217.160.0.225

# dig www.ittraining.center

; <<>> DiG 9.6-ESV-R8 <<>> www.ittraining.center
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50638
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.ittraining.center.         IN      A

;; ANSWER SECTION:
www.ittraining.center.  3564    IN      A       217.160.0.225

;; Query time: 6 msec
```

```
;; SERVER: 10.0.2.3#53(10.0.2.3)
;; WHEN: Thu Jan 16 11:11:09 2020
;; MSG SIZE  rcvd: 55
```

Services réseaux

Quand un client émet une demande de connexion vers une application réseau sur un serveur, il utilise un socket attaché à un port local **supérieur à 1023**, alloué d'une manière dynamique. La requête contient le port de destination sur le serveur.

inetd

Historiquement les serveurs étaient gérés par le service **inetd**. Lors de l'utilisation d'inetd, les serveurs gérés ne sont lancés qu'en cas de besoin, c'est-à-dire, lors d'une connexion. Le fichier de configuration d'inetd est **/etc/inetd.conf** :

```
# cat /etc/inetd.conf
#
# Copyright 2004 Sun Microsystems, Inc.  All rights reserved.
# Use is subject to license terms.
#
#ident  "@(#)inetd.conf 1.56    04/10/21 SMI"
#
# Legacy configuration file for inetd(1M).  See inetd.conf(4).
#
# This file is no longer directly used to configure inetd.
# The Solaris services which were formerly configured using this file
# are now configured in the Service Management Facility (see smf(5))
# using inetadm(1M).
#
# Any records remaining in this file after installation or upgrade,
# or later created by installing additional software, must be converted
# to smf(5) services and imported into the smf repository using
```

```
# inetconv(1M), otherwise the service will not be available.  Once
# a service has been converted using inetconv, further changes made to
# its entry here are not reflected in the service.
#
#
# CacheFS daemon.  Provided only as a basis for conversion by inetconv(1M).
#
100235/1 tli rpc/ticotsord wait root /usr/lib/fs/cachefs/cachefsd cachefsd
# TFTPD - tftp server (primarily used for booting)
#tftp   dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
```

Les lignes de configuration des serveurs ressemblaient à :

```
#tftp   dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
```

Notez que cette ligne est précédée par le caractère **#** indiquant que le serveur concerné est désactivé. Pour activer le serveur, il convient d'enlever le caractère **#**, de sauvegarder le fichier et de relancer le service inetd.

Le premier champs de la ligne identifie le nom du port qui identifie l'application. Inetd lit ensuite le fichier **/etc/services** pour déduire le numéro de port concerné et se met à l'écoute de celui-ci.

Le deuxième et le troisième champs définissent le type de protocole, à savoir:

- stream tcp pour le tcp
- dgram udp pour l'udp

Le quatrième champs prend un de deux valeurs:

- nowait
  - indique qu'il y aura un serveur par client
- wait
  - indique qu'il y aura un seul serveur pour l'ensemble des clients

Le cinquième champs indique l'identité sous laquelle sera exécuté le serveur.

Le sixième champs indique l'exécutable. Dans ce cas c'est **/usr/sbin/in.tftpd**.

Le septième champs indique les arguments de l'application serveur dont l'argument **0** est le nom de l'application.

TCP Wrapper

Lors de l'utilisation d'inetd, **TCP Wrapper** pouvait être utilisé pour contrôler l'accès à des services réseaux grâce à des **ACL**. Par exemple la ligne dans le fichier /etc/inetd.conf :

```
tftp   dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
```

était écrite de la façon suivante pour pouvoir utiliser TCP Wrapper :

```
tftp   dgram   udp6    wait    root    /usr/sbin/tcpd      in.tftpd -s /tftpboot
```

quand une requête arrive pour le serveur tftp, inetd activait le wrapper **tcpd** au lieu d'activer **in.tftpd** directement.

tcpd mettait à jour un journal et vérifiait si le client avait le droit d'utiliser le service concerné. Les ACL se trouvent dans deux fichiers:

- **/etc/hosts.allow**
- **/etc/host.deny**

Il faut noter que si ces fichiers n'existent pas ou sont vides, il n'y a pas de contrôle d'accès.

Le format d'une ligne dans un de ces deux fichiers est:

```
démon : liste_de_clients
```

Par exemple dans le cas de notre serveur tftp, on verrait une ligne dans le fichier **/etc/hosts.allow** similaire à:

```
in.tftpd : 192.168.1.10, .fenestros.com
```

ce qui implique que la machine dont le numéro IP est le 192.168.1.10 ainsi que les machines du domaine **fenestros.com** sont autorisées à utiliser le

service.

Le mot clef **ALL** pouvait être utilisé pour indiquer tout. Par exemple, **ALL:ALL** dans le fichier **/etc/host.deny** bloque effectivement toute tentative de connexion à un service inetd sauf pour les ACL inclus dans le fichier /etc/host.allow.

SMF

Depuis Solaris 10, les fonctionnalités d'inetd sont comprises dans SMF. Par défaut, TCP Wrappers est désactivé :

```
# svcprop -p defaults inetd
defaults/bind_addr astring ""
defaults/bind_fail_interval integer -1
defaults/bind_fail_max integer -1
defaults/con_rate_offline integer -1
defaults/connection_backlog integer 10
defaults/failrate_cnt integer 40
defaults/failrate_interval integer 60
defaults/inherit_env boolean true
defaults/max_con_rate integer -1
defaults/max_copies integer -1
defaults/stability astring Evolving
defaults/tcp_trace boolean false
defaults/tcp_wrappers boolean false
defaults/value_authorization astring solaris.smf.value.inetd
```

Pour activer TCP Wrappers, il convient de modifier les propriétés de ce service :

```
# svccfg -s inetd setprop defaults/tcp_wrappers=true
# svcadm refresh inetd
# svcprop -p defaults inetd | grep tcp_wrappers
defaults/tcp_wrappers boolean true
```

Vous pouvez maintenant vérifier si les services, tel **telnet** et **ftp**, gérés par SMF inetd sont configurés pour utiliser TCP Wrappers :

```
# inetadm -l telnet | grep tcp_wrappers
default  tcp_wrappers=TRUE
# inetadm -l ftp | grep tcp_wrappers
default  tcp_wrappers=TRUE
```

Pour intégrer le serveur tftpd dans inetd/SMF, il convient d'abord de vérifier si le fichier /etc/services contient une ligne tftp :

```
#  cat /etc/services | grep tftp
tftp            69/udp
```

Ensuite, il est nécessaire de créer un fichier au format des lignes inetd classiques et de l'importer dans SMF grâce à la commande **inetconv** :

```
# echo "tftp   dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot" > /inetd.tftp
# cat /inetd.tftp
tftp   dgram   udp6    wait    root    /usr/sbin/in.tftpd      in.tftpd -s /tftpboot
# inetconv  -i  /inetd.tftp
tftp -> /var/svc/manifest/network/tftp-udp6.xml
Importation de tftp-udp6.xml ...Terminé
```

A l'examen du fichier manifest pour tftp, vous constaterez que le service est **network/tftp/udp6** et que le service qui le contrôle est **svc:/network/inetd:default** :

```
# cat /var/svc/manifest/network/tftp-udp6.xml
<?xml version='1.0'?>
<!DOCTYPE service_bundle SYSTEM '/usr/share/lib/xml/dtd/service_bundle.dtd.1'>
<!--
    Service manifest for the tftp service.

    Generated by inetconv(1M) from inetd.conf(4).
-->

<service_bundle type='manifest' name='inetconv:tftp'>
```

```
<service
        name='network/tftp/udp6'
        type='service'
        version='1'>

        <create_default_instance enabled='true'/>

        <restarter>
                <service_fmri value='svc:/network/inetd:default' />
        </restarter>

        <!--
            Set a timeout of 0 to signify to inetd that we don't want to
            timeout this service, since the forked process is the one that
            does the service's work. This is the case for most/all legacy
            inetd services; for services written to take advantage of SMF
            capabilities, the start method should fork off a process to
            handle the request and return a success code.
        -->
        <exec_method
                type='method'
                name='inetd_start'
                exec='/usr/sbin/in.tftpd -s /tftpboot'
                timeout_seconds='0'>
                <method_context>
                        <method_credential user='root' group='root' />
                </method_context>
        </exec_method>

        <!--
            Use inetd's built-in kill support to disable services.
        -->
        <exec_method
                type='method'
```

```
                name='inetd_disable'
                exec=':kill'
                timeout_seconds='0'>
        </exec_method>

        <!--
            Use inetd's built-in process kill support to offline wait type
            services.
        -->
        <exec_method
                type='method'
                name='inetd_offline'
                exec=':kill_process'
                timeout_seconds='0'>
        </exec_method>

        <!--
            This property group is used to record information about
            how this manifest was created.  It is an implementation
            detail which should not be modified or deleted.
        -->
        <property_group name='inetconv' type='framework'>
                <propval name='converted' type='boolean' value='true' />
                <propval name='version' type='integer' value='1' />
                <propval name='source_line' type='astring' value=
'tftp dgram udp6 wait root /usr/sbin/in.tftpd      in.tftpd -s /tftpboot'
                />
        </property_group>

        <property_group name='inetd' type='framework'>
                <propval name='name' type='astring' value='tftp' />
                <propval name='endpoint_type' type='astring' value='dgram' />
                <propval name='proto' type='astring' value='udp6' />
                <propval name='wait' type='boolean' value='true' />
```

```
                <propval name='isrpc' type='boolean' value='false' />
        </property_group>

        <stability value='External' />

        <template>
                <common_name>
                        <loctext xml:lang='C'>
tftp
                        </loctext>
                </common_name>
        </template>
</service>

</service_bundle>
```

Activez maintenant le service tftp au sein d'inetd grâce à la commande **inetadm** :

```
# inetadm -e svc:/network/tftp/udp6:default
```

et constatez ses propriétés :

```
# inetadm   -l   svc:/network/tftp/udp6;default
SCOPE    NAME=VALUE
         name="tftp"
         endpoint_type="dgram"
         proto="udp6"
         isrpc=FALSE
         wait=TRUE
         exec="/usr/sbin/in.tftpd -s /tftpboot"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
default  bind_fail_interval=-1
```

```
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=FALSE
default  tcp_wrappers=TRUE
default  connection_backlog=10
default: not found
```

Commandes de base

ifconfig

Pour afficher la configuration IP de la machine il faut saisir la commande suivante :

```
#ifconfig -a [Entrée]
```

Vous obtiendrez une fenêtre similaire à celle-ci :

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
        inet 10.0.2.15 netmask ffffff00 broadcast 10.0.2.255
        ether 8:0:27:79:12:5b
```

hostname

Pour afficher la configuration nom d'hôte de la machine, deux commandes peuvent être utilisées :

```
# hostname
solaris.i2tch.loc
# uname -a
SunOS solaris.i2tch.loc 5.10 Generic_147148-26 i86pc i386 i86pc
```

ping

Pour tester l'accessibilité d'une machine, vous devez utiliser la commande ping :

```
# ping www.ittraining.center
www.ittraining.center is alive
```

Pour rendre la commande verbose, utilisez l'option **-s** :

```
# ping -s www.ittraining.center
PING www.ittraining.center: 56 data bytes
64 bytes from 217-160-0-225.elastic-ssl.ui-r.com (217.160.0.225): icmp_seq=0. time=28.7 ms
64 bytes from 217-160-0-225.elastic-ssl.ui-r.com (217.160.0.225): icmp_seq=1. time=43.1 ms
64 bytes from 217-160-0-225.elastic-ssl.ui-r.com (217.160.0.225): icmp_seq=2. time=35.2 ms
64 bytes from 217-160-0-225.elastic-ssl.ui-r.com (217.160.0.225): icmp_seq=3. time=41.7 ms
64 bytes from 217-160-0-225.elastic-ssl.ui-r.com (217.160.0.225): icmp_seq=4. time=38.5 ms
^C
----www.ittraining.center PING Statistics----
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 28.7/37.4/43.1/5.8
```

netstat -i

Pour visualiser les statistiques réseaux, vous disposez de la commande **netstat** :

```
# netstat -i
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      244    0     244    0     0      0
e1000g0 1500 solaris.i2tch.loc solaris       659    0     451    0     0      0
```

Connexion à Distance

Telnet

La commande **telnet** est utilisée pour établir une connexion à distance avec un serveur telnet. Le service telnet revient à une redirection des canaux standards d'entrée et de sortie. Notez que la connexion n'est **pas** sécurisée. Pour fermer la connexion, il faut saisir la commande **exit**. La commande telnet n'offre pas de services de transfert de fichiers. Pour cela, il convient d'utiliser la command **ftp**.

Sous Solaris, le serveur telnet est démarré par défaut :

```
# svcs telnet
STATE          STIME    FMRI
online         11:13:42 svc:/network/telnet:default
```

Vérifiez que votre serveur fonctionne en vous connectant à localhost :

```
# telnet localhost
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
login: user2
Password: tra@inee
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user2
$ exit
```

```
Connection to localhost closed by foreign host.
```

ssh

La commande **ssh** est le successeur et la remplaçante de la commande **rlogin**. Il permet d'établir des connexions sécurisées avec une machine distante. SSH comporte cinq acteurs :

- Le **serveur SSH**
  - le démon sshd, qui s'occupe des authentifications et autorisations des clients,
- Le **client SSH**
  - ssh ou scp, qui assure la connexion et le dialogue avec le serveur,
- La **session** qui représente la connexion courante et qui commence juste après l'authentification réussie,
- Les **clefs**
  - **Couple de clef utilisateur asymétriques** et persistantes qui assurent l'identité d'un utilisateur et qui sont stockés sur disque dur,
  - **Clef hôte asymétrique et persistante** garantissant l'identité du serveur er qui est conservé sur disque dur
  - **Clef serveur asymétrique et temporaire** utilisée par le protocole SSH1 qui sert au chiffrement de la clé de session,
  - **Clef de session symétrique qui est générée aléatoirement** et qui permet le chiiffrement de la communication entre le client et le serveur. Elle est détruite en fin de session. SSH-1 utilise une seule clef tandis que SSH-2 utilise une clef par direction de la communication,
- La **base de données des hôtes connus** qui stocke les clés des connexions précédentes.

SSH fonctionne de la manière suivante pour la la mise en place d'un canal sécurisé:

- Le client contacte le serveur sur son port 22,
- Les client et le serveur échangent leur version de SSH. En cas de non-compatibilité de versions, l'un des deux met fin au processus,
- Le serveur SSH s'identifie auprès du client en lui fournissant :
  - Sa clé hôte,
  - Sa clé serveur,
  - Une séquence aléatoire de huit octets à inclure dans les futures réponses du client,
  - Une liste de méthodes de chiffrage, compression et authentification,
- Le client et le serveur produisent un identifiant identique, un haché MD5 long de 128 bits contenant la clé hôte, la clé serveur et la séquence aléatoire,
- Le client génère sa clé de session symétrique et la chiffre deux fois de suite, une fois avec la clé hôte du serveur et la deuxième fois avec la clé serveur. Le client envoie cette clé au serveur accompagnée de la séquence aléatoire et un choix d'algorithmes supportés,

- Le serveur déchiffre la clé de session,
- Le client et le serveur mettent en place le canal sécurisé.

SSH-1

SSH-1 utilise une paire de clefs de type RSA1. Il assure l'intégrité des données par une **Contrôle de Redondance Cyclique** (CRC) et est un bloc dit **monolithique**.

Afin de s'identifier, le client essaie chacune des six méthodes suivantes :

- **Kerberos**,
- **Rhosts**,
- **RhostsRSA**,
- Par **clef asymétrique**,
- **TIS**,
- Par **mot de passe**.

SSH-2

SSH-2 utilise **DSA** ou **RSA**. Il assure l'intégrité des données par l'algorithme **HMAC**. SSH-2 est organisé en trois **couches** :

- **SSH-TRANS** – Transport Layer Protocol,
- **SSH-AUTH** – Authentification Protocol,
- **SSH-CONN** – Connection Protocol.

SSH-2 diffère de SSH-1 essentiellement dans la phase authentification.

Trois méthodes d'authentification :

- Par **clef asymétrique**,
  - Identique à SSH-1 sauf avec l'algorithme DSA,
- **RhostsRSA**,
- Par **mot de passe**.

L'authentification par mot de passe

Connectez-vous à votre propre machine en utilisant SSH et le compte **user2** :

```
# ssh -l user2 localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 44:8f:9b:56:4d:48:44:e3:d8:df:bd:0a:f5:ef:1d:4e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Password: tra@inee
Last login: Thu Jan 16 11:50:37 2020 from localhost
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user2
```

Pour fermer la connexion, utilisez la commande **exit** :

```
$ exit
Connection to localhost closed.
```

L'authentification par clef asymétrique

- Le **client** envoie au serveur une requête d'authentification par clé asymétrique qui contient le module de la clé à utiliser,
- Le **serveur** recherche une correspondance pour ce module dans le fichier des clés autorisés **~/.ssh/authorized_keys**,
  - Dans le cas où une correspondance n'est pas trouvée, le serveur met fin à la communication,
  - Dans le cas contraire le serveur génère une chaîne aléatoire de 256 bits appelée un **challenge** et la chiffre avec la **clé publique du client**,
- Le **client** reçoit le challenge et le décrypte avec la partie privée de sa clé. Il combine le challenge avec l'identifiant de session et chiffre le résultat. Ensuite il envoie le résultat chiffré au serveur.
- Le **serveur** génère le même haché et le compare avec celui reçu du client. Si les deux hachés sont identiques, l'authentification est réussie.

Installation

Pour mettre en place ce système il convient d'utiliser la commande **ssh-keygen** :

```
# su - user2
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user2
$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/export/home/user2/.ssh/id_rsa):
Created directory '/export/home/user2/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /export/home/user2/.ssh/id_rsa.
Your public key has been saved in /export/home/user2/.ssh/id_rsa.pub.
The key fingerprint is:
d2:3c:5c:2b:98:50:e8:b5:a2:0e:f6:b6:ff:7f:b0:2f user2@solaris.i2tch.loc
```

Lors de la question concernant la passphrase, notez qu'il faut entrer une passphrase **vide**.

Il convient ensuite de se connecter sur le serveur en utilisant ssh :

```
$ssh -l nom_de_compte numero_ip (nom_de_machine) [Entrée]
```

Par exemple :

```
$ ssh -l user2 localhost
The authenticity of host 'localhost (127.0.0.1)' can't be established.
RSA key fingerprint is 44:8f:9b:56:4d:48:44:e3:d8:df:bd:0a:f5:ef:1d:4e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (RSA) to the list of known hosts.
Password: tra@inee
```

```
Last login: Thu Jan 16 11:58:47 2020 from localhost
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user2
$ ls -la | grep .ssh
drwx------   2 user2    groupe2      512 Jan 16 12:01 .ssh
$ exit
Connection to localhost closed.
```

Si le dossier distant .ssh n'existe pas dans le répertoire personnel de l'utilisateur connecté, il faut le créer avec des permissions de 700 :

scp

Ensuite, il convient de transférer le fichier local **.ssh/id_rsa.pub** de la **machine cliente** vers le serveur en le renommant en **authorized_keys** :

```
$ scp /export/home/user2/.ssh/id_rsa.pub user2@localhost:/export/home/user2/.ssh/authorized_keys
Password:
id_rsa.pub           100% |*****************************|   233       00:00
```

Lors de la connexion suivante au serveur en tant qu'user1, l'authentification utilise le couple de clefs asymétrique et aucun mot de passe n'est requis :

```
$ ssh localhost
Last login: Thu Jan 16 12:04:48 2020 from localhost
Oracle Corporation      SunOS 5.10      Generic Patch   January 2005
$ pwd
/export/home/user2
$ exit
Connection to localhost closed.
```

Pour une protection supplémentaire du fichier **authorized_keys**, modifiez les permissions sur ce fichier à 600.

Transfert de Fichiers

wget

Si nécessaire, installez **pkgutil** :

```
# pkgadd -d http://get.opencsw.org/now

## Downloading...
.............25%.............50%.............75%.............100%
## Download Complete


The following packages are available:
  1  CSWpkgutil     pkgutil - Installs Solaris packages easily
                    (all) 2.6.7,REV=2014.10.16

Select package(s) you wish to process (or 'all' to process
all packages). (default: all) [?,??,q]: 1

Processing package instance <CSWpkgutil> from <http://get.opencsw.org/now>

pkgutil - Installs Solaris packages easily(all) 2.6.7,REV=2014.10.16
Please see /opt/csw/share/doc/pkgutil/license for license information.
## Processing package information.
## Processing system information.
## Verifying package dependencies.
## Verifying disk space requirements.
## Checking for conflicts with packages already installed.
## Checking for setuid/setgid programs.

This package contains scripts which will be executed with super-user
```

```
permission during the process of installing this package.

Do you want to continue with the installation of <CSWpkgutil> [y,n,?] y

Installing pkgutil - Installs Solaris packages easily as <CSWpkgutil>

## Installing part 1 of 1.
/etc/opt/csw/pkgutil.conf.CSW
/etc/opt/csw <implied directory>
/opt/csw/bin/pkgutil
/opt/csw <implied directory>
/opt/csw/bin <implied directory>
/opt/csw/etc/pkgutil.conf.CSW
/opt/csw/etc <implied directory>
/opt/csw/libexec/pkgutil/wget-i386
/opt/csw/libexec/pkgutil/wget-sparc
/opt/csw/share/doc/pkgutil/license
/opt/csw/share/doc/pkgutil/readme
/opt/csw/share/man/man1/pkgutil.1
/opt/csw/var/pkgutil/admin.CSW
[ verifying class <none> ]
## Executing postinstall script.

Copying sample pkgutil.conf to /opt/csw/etc.
Copying sample pkgutil.conf to /etc/opt/csw.
Copying sample admin from /opt/csw/var/pkgutil to /var/opt/csw/pkgutil.

NOTE!
NOTE! Make sure to check out any changes in /etc/opt/csw/pkgutil.conf.CSW.
NOTE!


Installation of <CSWpkgutil> was successful.
```

Pour installer wget, utilisez la commande **pkgutil** :

```
# /opt/csw/bin/pkgutil --install wget
=> Fetching new catalog and descriptions (http://mirror.opencsw.org/opencsw/testing/i386/5.10) if available ...
Solving needed dependencies ...
Solving dependency order ...
Install 25 NEW packages:
        CSWcas-migrateconf-1.50,REV=2015.01.17 (opencsw/testing)
        CSWcas-preserveconf-1.50,REV=2015.01.17 (opencsw/testing)
        CSWcas-texinfo-1.50,REV=2015.01.17 (opencsw/testing)
        CSWcommon-1.5,REV=2010.12.11 (opencsw/testing)
        CSWggettext-data-0.19.8,REV=2016.09.08 (opencsw/testing)
        CSWiconv-1.14,REV=2011.08.08 (opencsw/testing)
        CSWlibassuan0-2.4.1,REV=2015.11.24 (opencsw/testing)
        CSWlibcharset1-1.14,REV=2011.08.07 (opencsw/testing)
        CSWlibexpat1-2.2.7,REV=2019.06.28 (opencsw/testing)
        CSWlibgcc-s1-5.5.0,REV=2017.10.23 (opencsw/testing)
        CSWlibgpg-error0-1.20,REV=2015.11.20 (opencsw/testing)
        CSWlibgpgme11-1.6.0,REV=2016.07.06 (opencsw/testing)
        CSWlibiconv2-1.14,REV=2011.08.07 (opencsw/testing)
        CSWlibidn2-0-2.0.4,REV=2018.01.16 (opencsw/testing)
        CSWlibintl8-0.18.1.1,p,REV=2011.03.15 (opencsw/testing)
        CSWlibintl9-0.19.8,REV=2016.09.08 (opencsw/testing)
        CSWlibmetalink3-0.1.3,REV=2015.06.27 (opencsw/testing)
        CSWlibpcre1-8.38,REV=2015.11.23 (opencsw/testing)
        CSWlibpsl5-0.21.0,REV=2019.08.16 (opencsw/testing)
        CSWlibssl1-0-0-1.0.2t,REV=2019.09.11 (opencsw/testing)
        CSWlibunistring2-0.9.9,REV=2018.03.11 (opencsw/testing)
        CSWlibuuid1-1.0.2,REV=2014.08.12 (opencsw/testing)
        CSWlibz1-1.2.8,REV=2013.09.23 (opencsw/testing)
        CSWpublic-suffix-list-20190816,REV=2019.08.16 (opencsw/testing)
        CSWwget-1.20.3,REV=2019.11.28 (opencsw/testing)
Total size: 9.6 MB
25 packages to fetch. Do you want to continue? ([y],n,auto) y
```

```
...
The following files are already installed on the system and are being
used by another package:
* /etc/opt/csw <attribute change only>
* /opt/csw <attribute change only>
* /opt/csw/bin <attribute change only>
* /opt/csw/etc <attribute change only>
* /opt/csw/share <attribute change only>
* /opt/csw/share/doc <attribute change only>
* /opt/csw/share/man <attribute change only>
* /opt/csw/var <attribute change only>
* /var/opt/csw <attribute change only>

* - conflict with a file which does not belong to any package.

Do you want to install these conflicting files [y,n,?,q] y
...
Installation of <CSWwget> was successful.
```

La commande **wget** est utilisée pour récupérer un fichier via http ou ftp.

```
# /opt/csw/bin/wget http://ittraining.center/wget.txt
--2020-01-16 12:23:18--  http://ittraining.center/wget.txt
Resolving ittraining.center (ittraining.center)... 217.160.0.225, 2001:8d8:100f:f000::2c4
Connecting to ittraining.center (ittraining.center)|217.160.0.225|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 0 [text/plain]
Saving to: 'wget.txt'

wget.txt                [ <=>                 ]       0  --.-KB/s    in 0s

2020-01-16 12:23:18 (0.00 B/s) - 'wget.txt' saved [0/0]
```

ftp

Avant de commencer, placez-vous dasn le répertoire **/tmp** :

```
# cd /tmp
# pwd
/tmp
```

La commande **ftp** est utilisée pour le transfert de fichiers :

```
# ftp localhost
Connected to localhost.
220 solaris.i2tch.loc FTP server ready.
Name (localhost:root): user1
331 Password required for user1.
Password:tra@inee
230 User user1 logged in.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Une fois connecté, il convient d'utiliser la commande **help** pour afficher la liste des commandes disponibles :

```
ftp> help
Commands may be abbreviated.  Commands are:

!               cr              mdir            protect         safe
$               delete          mechanism       proxy           send
account         debug           mget            put             sendport
append          dir             mkdir           pwd             site
ascii           disconnect      mls             quit            status
bell            form            mode            quote           struct
binary          get             mput            recv            sunique
```

```
bye             glob            nlist           reget           tcpwindow
case            hash            nmap            remotehelp      tenex
ccc             help            ntrans          rename          trace
cd              lcd             open            reset           type
cdup            ls              passive         restart         user
clear           macdef          private         rmdir           verbose
close           mdelete         prompt          runique         ?
```

Parmi les commandes listées, on peut remarquer les commandes :

```
ftp> pwd
257 "/export/home/user1" is current directory.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
local.cshrc
local.login
local.profile
226 Transfer complete.
41 bytes received in 0.00014 seconds (283.62 Kbytes/s)
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
total 8
-rw-r--r--   1 user1    groupe1      144 Jan 14 12:48 .profile
-rw-r--r--   1 user1    groupe1      136 Jan 14 12:48 local.cshrc
-rw-r--r--   1 user1    groupe1      157 Jan 14 12:48 local.login
-rw-r--r--   1 user1    groupe1      174 Jan 14 12:48 local.profile
226 Transfer complete.
276 bytes received in 3.1e-05 seconds (8822.05 Kbytes/s)
```

Par contre notez que le serveur FTP de SUN fourni en standard n'est **pas** dans un **chroot** :

```
ftp> cd /
```

```
250 CWD command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
Desktop
Documents
bin
boot
core
core.882
dev
devices
etc
export
home
inetd.tftp
kernel
lib
lost+found
mnt
net
opt
platform
proc
pwd.txt
rep
sbin
system
tmp
usr
var
vol
226 Transfer complete.
196 bytes received in 0.0012 seconds (153.31 Kbytes/s)
```

Une commande ftp très utile est **!**. Cette commande permet d'exécuter une commande sur la machine cliente. Dans notre cas, nous saisissons la commande suivante :

```
ftp> !pwd
/tmp
```

Pour transférer un fichier vers le serveur, il convient d'utiliser la commande **put** :

```
ftp> !ls
hsperfdata_noaccess  hsperfdata_root      ntp.conf
ftp> cd /export/home/user1
250 CWD command successful.
ftp> put ntp.conf
200 PORT command successful.
150 Opening BINARY mode data connection for ntp.conf.
226 Transfer complete.
local: ntp.conf remote: ntp.conf
454 bytes sent in 0.00039 seconds (1127.48 Kbytes/s)
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mput**. Créez d'abord les fichiers **ntp1.conf** et **ntp2.conf** :

```
ftp> !cp ntp.conf ntp1.conf
ftp> !cp ntp.conf ntp2.conf
ftp> !ls
hsperfdata_noaccess  hsperfdata_root      ntp.conf             ntp1.conf            ntp2.conf
```

Ensuite téléversez les fichiers :

```
ftp> mput ntp*
mput ntp.conf? y
200 PORT command successful.
150 Opening BINARY mode data connection for ntp.conf.
226 Transfer complete.
local: ntp.conf remote: ntp.conf
```

```
454 bytes sent in 0.00046 seconds (972.30 Kbytes/s)
mput ntp1.conf? y
200 PORT command successful.
150 Opening BINARY mode data connection for ntp1.conf.
226 Transfer complete.
local: ntp1.conf remote: ntp1.conf
454 bytes sent in 0.00033 seconds (1344.94 Kbytes/s)
mput ntp2.conf? y
200 PORT command successful.
150 Opening BINARY mode data connection for ntp2.conf.
226 Transfer complete.
local: ntp2.conf remote: ntp2.conf
454 bytes sent in 0.00026 seconds (1711.30 Kbytes/s)
```

Important - Notez l'utilisation du joker * dans la ligne de commande ftp. Vous remarquerez aussi que lors de chaque transfert, le serveur vous demande de le valider.

Dans le cas où on ne souhaite pas la demande de confirmation lors des transferts multiples, il convient de saisir la commande **prompt** :

```
ftp> prompt
Interactive mode off.
ftp> mput ntp*
200 PORT command successful.
150 Opening BINARY mode data connection for ntp.conf.
226 Transfer complete.
local: ntp.conf remote: ntp.conf
454 bytes sent in 0.00023 seconds (1929.68 Kbytes/s)
200 PORT command successful.
150 Opening BINARY mode data connection for ntp1.conf.
226 Transfer complete.
local: ntp1.conf remote: ntp1.conf
```

```
454 bytes sent in 0.00031 seconds (1438.13 Kbytes/s)
200 PORT command successful.
150 Opening BINARY mode data connection for ntp2.conf.
226 Transfer complete.
local: ntp2.conf remote: ntp2.conf
454 bytes sent in 0.00026 seconds (1720.71 Kbytes/s)
```

> **Important** - Dans ce cas, aucune confirmation ne vous êtes demandée, **y compris pour les fichiers déjà présents** sur le serveur.

Pour transférer un fichier du serveur, il convient d'utiliser la commande **get** :

```
ftp> pwd
257 "/export/home/user1" is current directory.
ftp> !pwd
/tmp
ftp> !rm ntp*
ftp> !ls
hsperfdata_noaccess  hsperfdata_root
ftp> get ntp.conf
200 PORT command successful.
150 Opening BINARY mode data connection for ntp.conf (454 bytes).
226 Transfer complete.
local: ntp.conf remote: ntp.conf
454 bytes received in 3.6e-05 seconds (12284.49 Kbytes/s)
ftp> !ls
hsperfdata_noaccess  hsperfdata_root      ntp.conf
```

Vous pouvez également transférer plusieurs fichiers à la fois grâce à la commande **mget** (voir la commande **mput** ci dessus).

Pour supprimer un fichier sur le serveur, il convient d'utiliser la commande **del** :

```
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
local.cshrc
local.login
local.profile
ntp.conf
ntp1.conf
ntp2.conf
226 Transfer complete.
73 bytes received in 0.00015 seconds (467.95 Kbytes/s)
ftp> del ntp.conf
250 DELE command successful.
ftp> del ntp1.conf
250 DELE command successful.
ftp> del ntp2.conf
250 DELE command successful.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
local.cshrc
local.login
local.profile
226 Transfer complete.
41 bytes received in 8.7e-05 seconds (458.65 Kbytes/s)
```

Pour fermer la session, il convient d'utiliser la commande **quit** :

```
ftp> quit
421 Timeout (180 seconds): closing control connection.
```

Routage

Routage statique

Pour afficher la table de routage de la machine, il convient d'utiliser la commande netstat :

```
# netstat -nr

Routing Table: IPv4
  Destination           Gateway           Flags  Ref     Use     Interface
-------------------- -------------------- ----- ----- ---------- ---------
default              10.0.2.2             UG        1          7 e1000g0
10.0.2.0             10.0.2.15            U         1         10 e1000g0
224.0.0.0            10.0.2.15            U         1          0 e1000g0
127.0.0.1            127.0.0.1            UH        4        173 lo0
```

La table issue de la commande **route** indique les informations suivantes:

- La destination qui peut être un hôte ou un réseau et est identifiée par le champs **Destination**,
- La route à prendre identifiée par les champs **Gateway** et **Interface**. Dans le cas d'une valeur de **default** ceci spécifie une route directe. La valeur d'Interface spécifie la carte à utiliser,
- Le champ **Flags** qui peux prendre un ou plusieurs de svaleurs suivantes:
  - U - **Up** - la route est active
  - H - **Host** - la route conduit à un hôte
  - G - **Gateways** - la route passe par une passerelle
- Le champ **Ref** indique le nombre de références à cette route. Ce champs est utilisé par le Noyau de Linux,
- Le champ **Use** indique le nombre de recherches associés à cette route,

La commande traceroute

La commande ping est à la base de la commande **traceroute**. Cette commande sert à découvrir la route empruntée pour accéder à un site donné :

```
#traceroute nom_de_machine [Entrée]
```

```
# traceroute www.ittraining.center
traceroute to www.ittraining.center (217.160.0.225), 30 hops max, 40 byte packets
 1  2.2.0.10.rev.sfr.net (10.0.2.2)  0.575 ms  0.001 ms  4.499 ms
 2  192.168.0.1 (192.168.0.1)  0.843 ms  0.589 ms  0.001 ms
 3  * * *
 4  10.4.2.13 (10.4.2.13)  25.509 ms 10.4.1.13 (10.4.1.13)  21.429 ms  36.168 ms
 5  172.16.88.251 (172.16.88.251)  32.603 ms  26.348 ms  29.515 ms
 6  93.230.154.77.rev.sfr.net (77.154.230.93)  30.286 ms  20.785 ms  19.468 ms
 7  129.10.136.77.rev.sfr.net (77.136.10.129)  20.625 ms  28.720 ms  33.833 ms
 8  129.10.136.77.rev.sfr.net (77.136.10.129)  32.237 ms  21.467 ms  19.066 ms
 9  oneandone.par.franceix.net (37.49.236.42)  28.966 ms  25.429 ms  30.333 ms
10  ae-8-0.bb-a.bap.rhr.de.oneandone.net (212.227.120.42)  25.951 ms  28.079 ms  37.076 ms
11  port-channel-3.gw-distd-sh-1.bap.rhr.de.oneandone.net (212.227.122.3)  28.975 ms  36.476 ms  40.521 ms
12  * * *
13  * * *
14  * * *
15  * * *
16  * * *
17  * *^C#
```

La commande route

La commande **route** permet de paramétrer le routage indirect. **Par exemple** (veuillez ne PAS saisir les deux commandes qui suivent) :

```
# route add -net 10.2.0.0 netmask 255.255.0.0 10.3.0.1
```

ou bien

```
# route add default 10.1.2.250
```

La commande route permet aussi de se renseigner sur une route particulière :

```
# route get default
   route to: default
destination: default
       mask: default
    gateway: 2.2.0.10.rev.sfr.net
  interface: e1000g0
      flags: <UP,GATEWAY,DONE,STATIC>
 recvpipe  sendpipe  ssthresh    rtt,ms rttvar,ms  hopcount      mtu     expire
       0         0         0         0         0         0      1500         0
```

Activer/désactiver le routage sur le serveur

Si le fichier **/etc/defaultrouter** est vide, le démon **in.routed** utilise **RIP** pour mettre à jour les tables de routage.

Pour désactiver le routage sur le serveur, il convient de créer le fichier **/etc/notrouter**.

Routage dynamique

Créez d'abord l'interface nic:1 avec une adresse IP de 192.168.1.1/24.

```
# echo "192.168.1.1" > /etc/hostname.e1000g0:1
# ifconfig e1000g0:1 plumb
# ifconfig e1000g0:1 192.168.1.1
# ifconfig e1000g0:1 netmask 255.255.255.0
# ifconfig e1000g0:1 up
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
e1000g0: flags=1004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4> mtu 1500 index 2
```

```
        inet 10.0.2.15 netmask ffffff00 broadcast 10.0.2.255
        ether 8:0:27:e1:88:45
e1000g0:1: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index 2
        inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
```

RouteD

Le démon **routed** gère le protocole **RIP** (Routing Information Protocol). Le démon in.routed est désactivé par défaut :

```
# svcs route
STATE          STIME    FMRI
disabled       11:02:15 svc:/network/routing/route:default
```

Il faut donc démarré le service:

```
# svcadm enable route
# svcs route
STATE          STIME    FMRI
online         14:21:53 svc:/network/routing/route:default
```

Les démons **routed** présents sur chaque passerelle vont construire la table de routage.

Annexe #1 - Comprendre les Réseaux

Présentation des Réseaux

La définition d'un réseau peut être résumé ainsi :

- un ensemble d'**Equipements** (systèmes et périphériques) communiquant entre eux,
- une entité destinée au transport de données dans différents environnements.

Pour que la communication soit efficace, elle doit respecter les critères suivants :

- présenter des informations compréhensibles par tous les participants,
- être compatible avec un maximum d'interlocuteurs différents (dans le cas d'un réseau, les interlocuteurs sont des équipements : imprimantes, ordinateurs, clients, serveurs, téléphones...),
- si l'interlocuteur n'est pas disponible, les informations ne doivent pas se perdre,
- permettre une réduction des coûts (par ex. interconnexion à bas coût),
- permettre une productivité accrue (par ex. interconnexion à haut débit),
- être sécurisée si les informations à transmettre sont dites sensibles,
- garantir l'**unicité** et de l'**universalité** de l'**accès à l'information**.

On peut distinguer deux familles d'**Equipements** - les **Eléments Passifs** et les **Eléments Actifs**.

Les **Eléments Passifs** transmettent le signal d'un point à un autre :

- **Les Infrastructures ou Supports** - des câbles, de l'atmosphère ou des fibres optiques permettant de relier **physiquement** des équipements,
- **La Topologie** - l'architecture d'un réseau définissant les connexions entre les **Equipements** et, éventuellement, la hiérarchie entre eux.

Les **Eléments Actifs** sont des équipements qui consomment de l'énergie en traitant ou en interprétant le signal. Les **Equipements** sont classés selon leurs fonctions :

- **Equipement de Distribution Interne au Réseau** - Répartiteur (Hub, Switch, Commutateur etc.), Borne d'accès (Hotspot), Convertisseur de signal (Transciever), Amplificateur (Répéteur) ...,
- **Equipement d'Interconnexion de Réseaux** - Routeurs, Ponts ...,
- **Nœuds** et **Interfaces Réseaux** - postes informatiques, équipements en réseau ....

Un **Nœud** est une extrémité de connexion qui peut être une intersection de plusieurs connexions ou de plusieurs **Equipements**.

Une **Interface Réseau** est une prise ou élément d'un **Equipement Actif** faisant la connexion vers d'autres **Equipements** réseaux et qui reçoit et émet des données.

> Dans le cas d'un mélange d'**Equipements** non-homogènes en termes de performances au sein du même réseau, c'est la loi du plus faible qui emporte.

Tous les **Equipements** connectés au même support doivent respecter un ensemble de règles appelé une **Protocole de Communication**.

Les **Protocoles de Communication** définissent de façon formelle et interopérable la manière dont les informations sont échangées entre les **Equipements**.

Des **Logiciels**, dédiés à la gestion de ces **Protocoles de Communication**, sont installés sur des **Equipements d'Interconnexion** afin de fournir des fonctions de contrôle permettant une communication entre les **Equipements**.

Se basant sur des **Protocoles de Communication**, des **Services** fournissent des fonctionnalités accessibles aux utilisateurs ou d'autres programmes.

L'ensemble des **Equipements**, **Logiciels** et **Protocoles de Communication** constitue l'**Architecture Réseau**.

Classification des Réseaux

Les réseaux peuvent être classifiés de trois façon différentes :

- par **Mode de Transmission**,
- par **Topologie**,
- par **Étendue**.

Classification par Mode de Transmission

Il existe deux **Classes** de réseaux dans cette classification :

- les **Réseaux en Mode de Diffusion**,
  - utilise un seul support de transmission,
  - le message est envoyé sur tout le réseau à l'adresse d'**un** destinataire,
- les **Réseaux en Mode Point à Point**,
  - une seule liaison entre deux équipements,
  - les nœuds permettent de choisir la route en fonction de l'adresse du destinataire,
  - quand deux nœuds non directement connectés entre eux veulent communiquer ils le font par l'intermédiaire des autres noeuds du réseau.

Classification par Topologie

La **Topologie Physique** d'un réseau décrit l'organisation de ce dernier en termes de câblage. La **Topologie Logique** d'un réseau décrit comment les données circulent sur le réseau. En effet c'est le choix des concentrateurs ainsi que les connections des câbles qui déterminent la topologie logique.

La Topologie Physique

Il existe 6 topologies physiques de réseau :

- La Topologie en Ligne,
- La Topologie en Bus,
- La Topologie en Etoile,
- La Topologie en Anneau,
- La Topologie en Arbre,
- La Topologie Maillée.

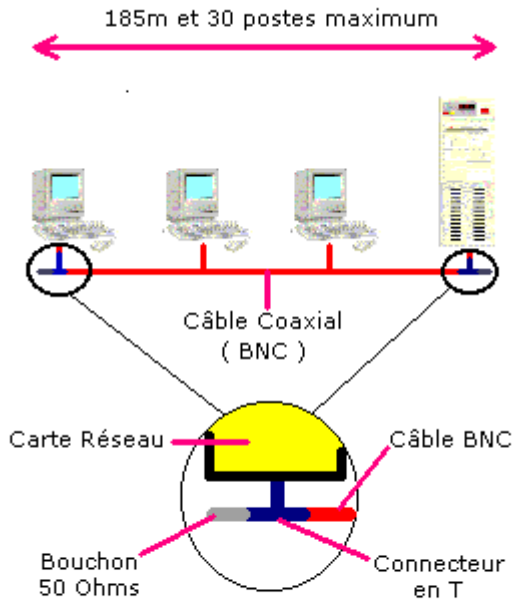
La Topologie en Ligne

Tous les nœuds sont connectés à un seul support. L'inconvénient de cette topologie est que dans le cas d'une défaillance d'une station, le réseau se trouve coupé en deux sous-réseaux.

La Topologie en Bus

Tous les nœuds sont connectés à un seul support (un câble BNC en T) avec des bouchons à chaque extrémité. La longueur du bus est limitée à **185m**. Le nombre de stations de travail est limité à **30**. Les Stations sont reliées au Bus par des 'T'. Les bouchons sont des terminateurs qui sont des résistances de **50 Ohms**. Quand le support tombe en panne, le réseau ne fonctionne plus. Quand une station tombe en panne, elle ne perturbe pas le

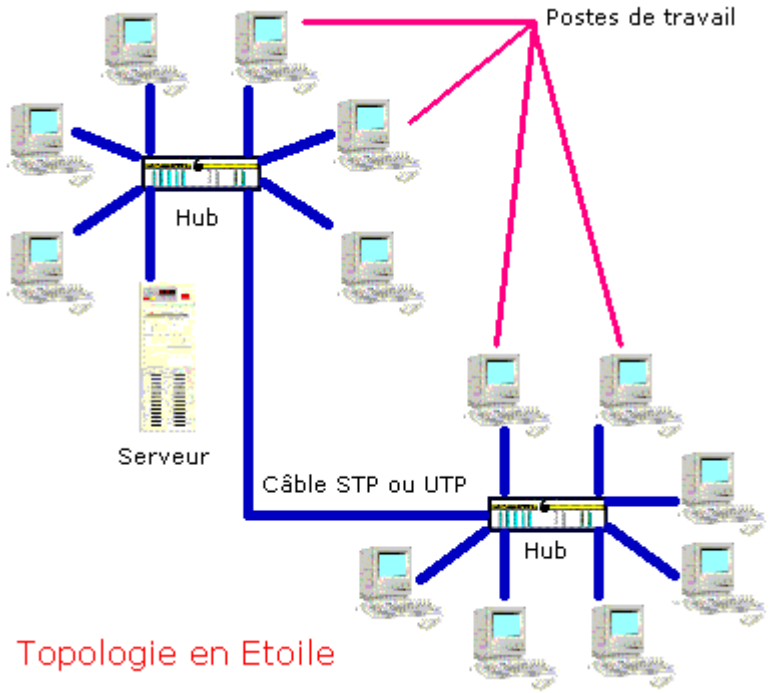
fonctionnement de l'ensemble du réseau. Les Stations étant reliés à un suel support, ce type de topologie necessite un **Protocole d'Accès** pour gérer le tour de parole des Stations afin d'éviter des conflits.



Topologie en Bus

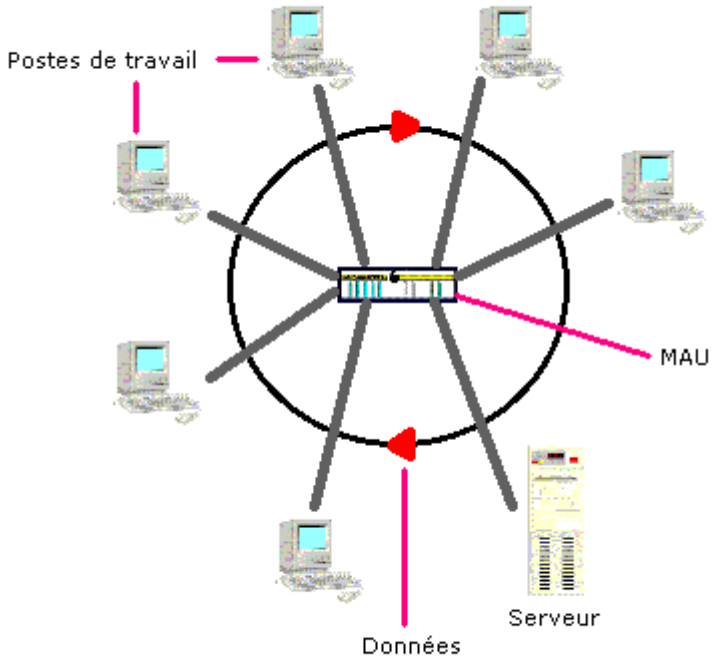
La Topologie en Étoile

Chaque nœud est connecté à un périphérique central appelé un **Hub** (**Concentrateur**) ou un **Switch** (**Commutateur**). Un Hub ou un Switch est prévu pour 4, 8, 16, 32 ... stations. En cas d'un réseau d'un plus grand nombre de stations, plusieurs Hubs ou Switches sont connectés ensemble. Quand une station tombe en panne, elle ne perturbe pas le fonctionnement de l'ensemble du réseau. Le point faible de cette topologie est l'équipement central.



La Topologie en Anneau

Chaque nœud est relié directement à ses deux voisins dans une topologie logique de cercle ininterrompu et une topologie physique en étoile car les stations sont reliées à un type de hub spécial, appelé un **Multistation Access Unit** (MAU).



Topologie en Anneau

Les stations sont reliées à la MAU par un câble 'IBM' munie d'une prise **AUI** du côté de la carte et une prise **Hermaphrodite** du coté de la MAU. Les données sont échangées dans un sens unidirectionnel. Une trame, appelée un **jeton**, circule en permanence. Si l'anneau est brisé, l'ensemble du réseau s'arrête. Pour cette raison, il est courant de voir deux anneaux contre-rotatifs.

La Topologie en Arbre

La Topologie en Arbre est utilisée dans un réseau hierarchique où le sommet, aussi appelé la **racine**, est connecté à plusieurs noeuds de niveau inférieur. Ces neouds peuvent à leur tour être connectés à d'autres noeuds inférieurs. L'ensemble forme une arborescence. Le point faible de cette topologie est sa racine. En cas de défaillance, le réseau est coupé en deux.

La Topologie Maillée

Cette Topologie est utilisée pour des grands réseaux de distribution tels Internet ou le WIFI. Chaque noeud à tous les autres via des liaisons point à point. Le nombre de liaisons devient très rapidement important en cas d'un grand nombre de noeuds. Par exemple dans le cas de 100 Stations (N), le nombre de liaisons est obtenu par la formule suivante :

```
N(N-1)/2 = 100(100-1)/2 = 4 950
```

La **Topologie Physique** la plus répandue est la **Topologie en Etoile**.

Classification par Etendue

La classification par étendue nous fournit 4 réseaux principaux :

Nom	Description	Traduction	Taille Approximative (M)
PAN	Personal Area Network	Réseau Personnel	1 -10
LAN	Local Area Network	Réseau Local Entreprise (RLE)	5 - 1 200
MAN	Métropolitain Area Network	Réseau Urbain	900 - 100 000
WAN	Wide Area Network	Réseau Long Distance (RLD)	50 000 et au delà

Cependant, d'autres classification existent :

CAN	Campus Area Network	Réseau de Campus
GAN	Global Area Network	Réseau Global
TAN	Tiny Area Network	Réseau Minuscule
FAN	Family Area Network	Réseau Familial
SAN	Storage Area Network	Réseau de Stockage

Etant donné que les WANs sont gérés par des opérateurs de télécommunications qui doivent demander une licence à l'état mais que les LANs ont été historiquement mis en oeuvre dans les entreprises, ces derniers sont en majorité issus du monde informatique.

Les Types de LAN

Il existe deux types de LAN :

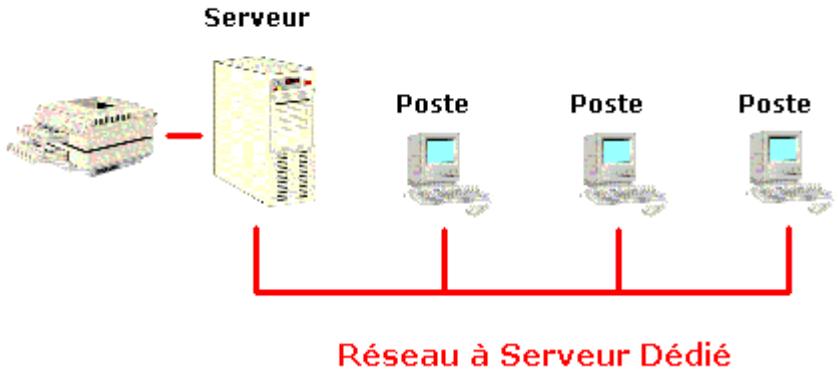
- le réseau à serveur dédié,
- le réseau poste à poste.

Réseau à Serveur Dédié

Le réseau à serveur dédié est caractérisé par le fait que toutes les ressources (imprimantes, applications, lecteurs etc.) sont gérées par le serveur. Les autres micro-ordinateurs ne jouent le rôle de client.

Des exemples des systèmes d'exploitation du réseau à serveur dédié sont :

- Windows NT Server,
- Windows 2000 Server,
- Windows 2003 Server,
- Windows 2008 Server,
- Linux,
- Unix.



Réseau Poste-à-Poste

Le réseau poste à poste est caractérisé par le fait que tous les ordinateurs peuvent jouer le rôle de client et de serveur :

- Windows 95,
- Windows 98,
- Windows NT Workstation.

Le Modèle Client/Serveur

Le modèle Client/Serveur est une des modalités des architectures informatiques distribuées. Dans ce modèle un serveur est tout **Logiciel** fournissant un **Service**.

Le serveur est aussi :

- passif, c'est-à-dire en attente permenante d'une demande, appelée une requête d'un client,
- capable de traiter plusieurs requêtes simultanément en utilisant le **multi-threading**,
- garant de l'intégrité globale.

Le client est, par contre **actif**, étant à l'origine des requêtes.

Il existe trois types de modèle client/serveur :

- **Plat** - tous les clients communiques avec un seul serveur,
- **Hiérarchique** - les clients n'ont de contact qu'avec les serveurs de plus haut niveau qu'eux,
- **Peer-to-Peer** - les équipements sont à la fois client **et** serveur en même temps.

Modèles de Communication

Les réseaux sont bâtis sur des technologies et des modèles. Le modèle **théorique** le plus important est le modèle **O**pen **S**ystem **Interconnection** créé par l'**I**nternational **Organization** for **S**tandardization tandis que le modèle pratique le plus important est le modèle **TCP/IP**.

Le modèle OSI

Le modèle OSI qui a été proposé par l'ISO est devenu le standard en termes de modèle pour décrire l'échange de données entre ordinateurs. Cette norme se repose sur sept couches, de la une - la Couche Physique, à la sept - la Couche d'Application, appelés des services. La communication entre les différentes couches est synchronisée entre le poste émetteur et le poste récepteur grâce à ce que l'on appelle un protocole.

Ce modèle repose sur trois termes :

- Les **Couches**,
- Les **Protocoles**,
- Les **Interfaces**.

Les Couches

Des sept couches :

- Les couches 1 à 3 sont les **Couches Basses** orientées **Transmission**,
- La couche 4 est la **Couche Charnière** entre les **Couches Basses** et les **Couches Hautes**,

- Les couches 5 à 7 sont les **Couches Hautes** orientées **Traitement**.

La couche du même niveau du système **A** parle avec son homologue du système **B**.

- **La Couche Physique** (Couche 1) est responsable :
  - du transfert de données binaires sur le câble physique ou virtuel
  - de la définition de tout aspect physique allant du connecteur jusqu'au câble en passant par la carte réseau, y compris l'organisation même du réseau
  - de la définition des tensions électriques sur le câble pour obtenir le 0 et le 1 binaires
- **La Couche de Liaison** (Couche 2) est responsable :
  - de la réception des données de la couche physique
  - de l'organisation des données en fragments, appelés des trames qui ont un format différent selon s'il s'agit d'un réseau basé sur la technologie Ethernet ou la technologie Token-Ring
  - de la préparation, émission et réception des trames
  - de la gestion de l'accès au réseau
  - de la communication nœud à nœud
  - de la gestion des erreurs
    - avant la transmission, le nœud émetteur calcule un code appelé un CRC et l'incorpore dans les données envoyées
    - le nœud récepteur recalcule un CRC en fonction du contenu de la trame reçue et le compare à celui incorporé avec l'envoi
    - en cas de deux CRC identique, le nœud récepteur envoie un accusé de réception au nœud émetteur
  - de la réception de l'accusé de réception
  - éventuellement de le ré-émission des données
  - En prenant ce modèle, l'IEEE (Institute of Electrical and Eletronics Engineers) l'a étendu avec le Modèle IEEE (802).
    - Dans ce modèle la Couche de Liaison est divisée en deux sous-couches importantes :
      - La **Sous-Couche LLC** (Logical Link Control) qui :
        - gère les accusés de réception
        - gère le flux de trames
      - La **Sous-Couche MAC** (Media Access Control) qui :
        - gère la méthode d'accès au réseau
        - le CSMA/CD dans un réseau basé sur la technologie Ethernet
        - l'accès au jeton dans un réseau basé sur la technologie Token-Ring
        - gère les erreurs
- **La Couche de Réseau** (Couche 3) est responsable de la gestion de la bonne distribution des différentes informations aux bonnes adresses en :

    - identifiant le chemin à emprunter d'un nœud donné à un autre
    - appliquant une conversion des adresses logiques (des noms) en adresses physiques
    - ajoutant des information adressage aux envois
    - détectant des paquets trop volumineux avant l'envoi et en les divisant en trames de données de tailles autorisées

- **La Couche de Transport** (Couche 4) est responsable de veiller à ce que les données soient envoyées correctement en :
    - constituant des paquets de données corrects
    - les envoyant dans le bon ordre
    - vérifiant que les données sont traités dans le même ordre que l'ordre d'émission
    - permettant à un processus sur un nœud de communiquer avec un autre nœud et d'échanger des messages avec lui

- **La Couche de Session** (Couche 5) est responsable :
    - de l'établissement, du maintien, et de la mise à fin de la communication entre deux noeuds distants, c'est-à-dire, de la session
    - de la conversation entre deux processus de vérification de la réception des messages envoyés en séquences, c'est-à-dire, le point de contrôle

- de la sécurité lors de l'ouverture de la session, c'est-à-dire, les droits d'utilisateurs etc.

- **La Couche de Présentation** (Couche 6) est responsable :
    - du formatage et de la mise en forme des données
    - des conversions de données telles le cryptage/décryptage

- **La Couche d'Application** (Couche 7) est responsable :
    - du dialogue homme/machine via des messages affichés
    - du partage des ressources
    - de la messagerie

Les Protocoles

Un **protocole** est un langage commun utilisé par dexu entités en communication pour pouvoir se comprendre. La nature du Protocole dépends directement de la nature de la communication. Cette bature dépend du **paradigme** de communication que l'application nécessite. Le paradigme est un modèle abstrait d'un problème ou d'une situation. Dans le paradigme de la diffusion, l'émetteur envoie dans informations au récepteur sans se soucier de ce que le récepteur va en faire. C'est la responsabilité du récepteur de comprendre et d'utiliser les informations.

Les Interfaces

Chaque couche rend des **services** à la couche immédiatement supérieure et utilise les services de la couche immédiatement inférieure. L'ensemble des services s'appelle une **Interface**. Les services sont composés de **S**ervice **D**ata **U**nits et sont disponibles par un **S**service **A**ccess **P**oint.

Protocol Data Units

L'**Unité de Données** ou *Protocol Data Unit* pour chaque couche comporte un nom spécifique :

- **Application Protocol Data Units** pour la couche **Application**,
- **Présentation Protocol Data Units** pour la couche **Présentation**,
- **Session Protocol Data Units** pour la couche **Session**,
- **Transport Protocol Data Units** pour la couche **Transport**.

Or, pour les **Couches Basses** on parle de :

- **Paquets** pour la couche **Réseau**,
- **Trames** pour la couche **Liaison**,
- **Bits** pouyr la couche **Physique**.

Encapsulation et Désencapsulation

Lorque les données sont communiqueés par le système A au système B, celles-ci commencent au niveau de la couche d'Application. Le couche d'Application ajoute une en-tête à l'unité de données qui contient des **informations de contrôle du protocole**. Au passage de chaque couche, celle-ci ajoute sa propre en-tête. De cette façon, lors de sa descente vers la couche physique, les données et l'entête de la couche supérieure sont encapulsulées :

Couche Système A	**Encapsulation**
Application	Application Header (AH) + Unité de Données (UD)
Présentation	Présentation Header (PH) + AH + UD
Session	Session Header (SH) + PH + AH + UD

Couche Système A	Encapsulation
Transport	Transport Header (TH) + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD

Lors de son voyage de la couche Physique vers la couche Application dans le système B, les en-têtes sont supprimées par chaque couche correspondante. On parle alors de **désencapsulation** :

Couche Système B	Encapsulation
Liaison	Liaison Header (DH) + NH + TH + SH + PH + AH + UD
Réseau	Network Header (NH) + TH + SH + PH + AH + UD
Transport	Transport Header (TH) + SH + PH + AH + UD
Session	Session Header (SH) + PH + AH + UD
Présentation	Présentation Header (PH) + AH + UD
Application	Application Header (AH) + Unité de Données (UD)

Spécification NDIS et le Modèle ODI

<note tip> **Cliquez ici pour ouvrir le schéma Simplifié du Modèle OSI incluant la spécification NDIS** </note>

La spécification NDIS (Network Driver Interface Specification) a été introduite conjointement par les sociétés Microsoft et 3Com. Cette spécification ainsi que son homologue, le modèle ODI (Open Datalink Interface) introduit conjointement par les sociétés Novell et Apple à la même époque, définit des standards pour les pilotes de cartes réseau afin qu'ils puissent être indépendants des protocoles utilisées et les systèmes d'exploitation sur les machines. Des deux 'standards', la spécification NDIS est le plus répandu, intervenant a niveau de la sous-couche MAC et l a couche de liaison. Elle spécifie :

- l'interface pilote-matériel
- l'interface pilote-protocole
- l'interface pilote - système d'exploitation

Le modèle TCP/IP

<note tip> **Cliquez ici pour voir le modèle OSI incluant la suite des protocoles et services TCP/IP** </note>

La suite des protocoles TCP/IP (Transmission Control Protocol / Internet Protocol) est issu de la DOD (Dept. Américain de la Défense) et le travail de l'ARPA (Advanced Research Project Agency).

- La suite des protocoles TCP/IP
  - a été introduite en 1974
  - a été utilisée dans l'ARPAnet en 1975
  - permet la communication entre des réseaux à base de systèmes d'exploitation, architectures et technologies différents
  - est très proche du modèle OSI en termes d'architecture et se place au niveau de la couche d'Application jusqu'à la couche Réseau.
  - est, en réalité, une suite de protocoles et de services :
    - **IP** (Internet Protocol)
      - le protocole IP s'intègre dans la couche Réseau du modèle OSI en assurant la communication entre les systèmes. Bien qu'il puisse découper des messages en fragments ou datagrammes et les reconstituer dans le bon ordre à l'arrivée, il ne garantit pas la réception.
    - **ICMP** (Internet Control Message Protocol)
      - le protocole ICMP produit des messages de contrôle aidant à synchroniser le réseau. Un exemple de ceci est la commande ping.
    - **TCP** (Transmission Control Protocol)
      - le protocole TCP se trouve au niveau de la couche de Transport du modèle OSI et s'occupe de la transmission des données entre noeuds.
    - **UDP** (User Datagram Protocol)
      - le protocole UDP n'est pas orienté connexion. Il est utilisé pour la transmission rapide de messages entre nœuds sans garantir leur acheminement.
    - **Telnet**
      - le protocole Telnet est utilisé pour établir une connexion de terminal à distance. Il se trouve dans la couche d'Application du modèle OSI.
    - **Ftp** (File Transfer Protocol)
      - le protocole ftp est utilisé pour le transfert de fichiers. Il se trouve dans la couche d'Application du modèle OSI.
    - **SMTP** (Simple Message Transfer Protocol)
      - le service SMTP est utilisé pour le transfert de courrier électronique. Il se trouve dans la couche d'Application du modèle OSI.
    - **DNS** (Domain Name Service)
      - le service DNS est utilisé pour le résolution de noms en adresses IP. Il se trouve dans la couche d'Application du modèle OSI.
    - **SNMP** (Simple Network Management Protocol)

        - le protocole SNMP est composé d'un agent et un gestionnaire. L'agent SNMP collecte des informations sur les périphériques, les configurations et les performances tandis que le gestionnaire SNMP reçois ses informations et réagit en conséquence.
    - **NFS** (Network File System)
        - le NFS a été mis au point par Sun Microsystems
        - le NFS génère un lien virtuel entre les lecteurs et les disques durs permettant de monter dans un disque virtuel local un disque distant
    - et aussi POP3, NNTP, IMAP etc ...

<note tip> **Cliquez ici pour voir les modèles TCP/IP et OSI** </note>

Le modèle TCP/IP est composé de 4 couches :

- La couche d'Accès Réseau
    - Cette couche spécifie la forme sous laquelle les données doivent être acheminées, quelque soit le type de réseau utilisé.
- La couche Internet
    - Cette couche est chargée de fournir le paquet de données.
- La couche de Transport
    - Cette couche assure l'acheminement des données et se charge des mécanismes permettant de connaître l'état de la transmission.
- La couche d'Application
    - Cette couche englobe les applications standards de réseau telles ftp, telnet, ssh, etc..

Les noms des Unités de Données sont différents selon le protocole utilisé et la couche du modèle TCP/IP :

Couche	**TCP**	**UDP**
Application	Stream	Message
Transport	Segment	Packet
Internet	Datagram	Datagram
Réseau	Frame	Frame

Les Raccordements

Les Modes de Transmission

On peut distinguer 3 modes de transmission :

- La **Liaison Simplex**,
  - Les données ne circulent que dans un **seul** sens de l'émetteur ver le récepteur,
  - La liaison nécessite deux canaux de transmissions,
- La **Liaison Half-Duplex** aussi appelée la **Liaison à l'Alternat** ou encore la **Liaison Semi-Duplex**,
  - Les données circulent dans un sens ou l'autre mais jamais dans les deux sens en même temps. Chaque extrémité émet donc à son tour,
  - La liaison permet d'avoir une liaison bi-directionnelle qui utilise la totalité de la banse passante,
- La **Liaison Full-Duplex** dans les deux sens en **même** temps. Chaque extrémité peut émettre et recevoir simultanément,
  - La liaison est caractérisée par une bande passante divisée par deux pour chaque sens des émissions.

Les Câbles

Le Câble Coaxial

En partant de l'extérieur, le câble coaxial est composé :

- d'une **Gaine** en caoutchouc, PVC ou Téflon pour protéger le câble,
- d'un **Blindage** en métal pour diminuer le bruit du aux parasites,
- d'un **Isolant** (diélectrique) pour éviter le contact entre le blindage et l'âme et ainsi éviter des courts-circuits,
- d'un **Âme** en cuivre ou torsadés pour transporter les données.

Avantages :

- **Peux coûteux**,
- Facilement **manipulable**,
- Peut être utilisé pour de **longues distances**,
- A un débit de 10 Mbit/s dans un LAN et 100 Mbit/s dans un WAN.

Inconvénients :

- Fragile,
- Instable,

- Vulnérable aux interférences,
- Half-Duplex.

Le Câble Paire Torsadée

Ce câble existe sous deux formes selon son utilisation :

- **Monobrin** pour du câblage **horizontal** (**Capillaire**),
  - chaque fil est composé d'un seul conducteur en cuivre,
  - la distance ne doit pas dépassée 90m.
- **Multibrin** pour des **cordons de brassage** :
  - chaque fil est composé de plusieurs brins en cuivre,
  - câble souple.

Avantages :

- Un débit de 10 Mbit/s à 10 GBit/s,
- A une bande passante plus large,
- Pas d'interruption par coupure du câble,
- Permet le **câblage universel** (téléphonie, fax, données ...),
- Full-Duplex.

Inconvénients :

- Nombre de câbles > câble coaxial,
- Plus cher,
- Plus encombrant dans les gaines techniques.

Catagories de Blindage

Il existe trois catagories de blindage :

- **Twisted** ou Torsadé,
- **Foiled** ou Entouré,
- **Shielded** ou Avec Ecran.

De ce fait, il existe 5 catagories de câbles Paire Torsadée :

Nom anglais ^ Appelation Ancienne ^ Nouvelle Appelation ^

Unshielded Twisted Pair	UTP	U/UTP
Foiled Twisted Pair	FTP	F/UTP
Shield Twisted Pair	STP	S/UTP
Shield Foiled Twisted Pair	SFTP	SF/UTP
Shield Shield Twisted Pair	S/STP	SS/STP3

Ces catégories donnent lieu à des **Classes** :

Classe	**Débit**	**Nombre de Paires Torsadées**	**Connecteur**	**Commentaires**
3	10 Mbit/s	4	RJ11	
4	16 Mbit/s	4	S/O	Non-utilisée de nos jours
5	100 Mbit/s	4	RJ45	Obsolète
5e/D	1 Gbit/s sur 100m	4	RJ45	S/O
6/E	2.5 Gbit/s sur 100m ou 10 Gbit/s sur 25m à 55m	4	Idéal pour PoE	
7/F	10 Gbit/s sur 100m	4	GG45 ou Tera	Paires individuellement et collectivement blindées. Problème de compatibilité avec les classes precédentes due au connecteur.

La Prise RJ45

Une prise RJ45 comporte 8 broches. Un câble peut être **droit** quand la broche 1 d'une extremité est connectée à la broche 1 de la prise RJ45 à l'autre extrémité, la broche 2 d'une extremité est connectée à la broche 2 de la prise RJ45 à l'autre extrémité et ainsi de suite ou bien **croisé** quand le brochage est inversé.

Les câbles croisés sont utilisés lors du branchement de deux équipements identiques (PC à PC, Hub à Hub, Routeur à Routeur).

Channel Link et Basic Link

Le **Channel Link** ou **Canal** est l'ensemble du **Basic Link** ou **Lien** de base et les cordons de brassage et de raccordement des équipements qui sont limités en distance à 10m.

Le **Basic Link** est le lien entre la prise RJ45 murale et la baie de brassage. Il est limité à 90m en classe 5D.

La Fibre Optique

La **Fibre Optique** est un fil de **Silice** permettant le transfert de la lumière. De ce fait elle est caractérisée par :

- des meilleures performances que le cuivre,
- de plus de communications simultanément,
- de la capacité de relier de plus grandes distances,
- une insensibilité aux perturbations,
- une résistance à la corrosion.

Qui plus est, elle ne produit aucune perturbation.

Elle est composée :

- d'un coeur de 10, de 50/125 ou de 62.50 micron,
- d'une gaine de 125 micron,
- d'une protection de 230 micron.

Il existe deux types de fibres, la **Fibre Monomode** et la **Fibre Multimodes**.

La Fibre Monomode :

- a un coeur de 8 à 10 Microns,
- est divisée en sous-catégories de distance,

    - 10 Km,
    - 15 Km,
    - 20 Km,
    - 50 Km,
    - 80 Km,
    - 100 Km.

La Fibre Multimode :

- a un coeur de 62,50 micron ou de 50/125 micron avec une gaine orange,
- permet plusieurs trajets lumineux appelés **modes** en même temps en Full Duplex,
- est utilisée pour de bas débits ou de courtes distances,
    - 2 Km pour 100 Mbit/s,
    - 500 m pour 1 Gbit/s.

Les Réseaux sans Fils

Les réseaux sans fils sans basés sur une liaison qui utilise des ondes radio-électriques (radio et infra-rouges).

Il existe des technologies différentes en fonction de la fréquence utilisée et de la portée des transmissions :

- Réseaux Personnels sans Fils - Bluetooth, HomeRF,
- Réseaux Locaux sans Fils - LiFI, WiFI,
- Réseaux Métropolitains sans Fil - wImax,
- Réseaux Etendus sans Fils - GSM, GPRS, UMTS.

Les principales ondes utilisées pour la transmission des données sont :

- Ondes GSM - Ondes Hertziennes repeosant sur des micro-ondes à basse fréquence avec une portée d'une dizaine de kilomètres,
- Ondes Wi-Fi - Ondes Hertziennes reposant sur des micro-ondes à haute fréquence avec une portée de 20 à 50 mètres,
- Ondes Satellitaires - Ondes Hertziennes longues portées.

Le Courant Porteur en Ligne

Le CPL utilise le réseau électrique domestique, le réseau moyenne et basse tension pour transmettre des informations numériques.

Le CPL superpose un signal à plus haute fréquence au signal électrique.

Seuls donc, les fils conducteurs transportent les signaux CPL.

Le coupleur intégré en entrée des boîtiers CPL élimine les composants basses fréquences pour isoler le signal CPL.

Le CPL utilise la phase électrique et le neutre. De ce fait, une installation triphasée fournit 3 réseaux CPL différents.

Le signal CPL ne s'arrête pas necéssairement aux limites de l'installation électrique. En effet en cas de compteurs non-numériques le signal les traversent.

Les normes CPL sont :

Norme	Débit Théorique	Débit Pratique	Temps pour copier 1 Go
Homeplug 1.01	14 Mbps	5.4 Mbps	25m 20s
Homeplug 1.1	85 Mbps	12 Mbps	11m 20s
PréUPA 200	200 Mbps	30 Mbps	4m 30s

Technologies

Il existe plusieurs technologies de réseau :

- Ethernet,
- Token-Ring,
- ARCnet,
- etc..

Nous détaillerons ici les deux technologies les plus répandues, à savoir Ethernet et Token-Ring.

Ethernet

La technologie Ethernet se repose sur :

- une topologie logique de bus,
- une topologie physique de bus ou étoile.

L'accès au bus utilise le **CSMA/CD**, Carrier Sense Multiple Access / Collision Detection (Accès Multiple à Détection de Porteuse / Détection de Collisions).

Il faut noter que :

- les données sont transmises à chaque nœud - c'est la méthode d'**accès multiple**,
- chaque nœud qui veut émettre écoute le réseau - c'est la **détection de porteuse**,
- quand le réseau est silencieux une trame est émise dans laquelle se trouvent les données ainsi que l'adresse du destinataire,
- le système est dit donc **aléatoire** ou **non-déterministe**,
- quand deux nœuds émettent en même temps, il y a **collision de données**,
- les deux nœuds vont donc cesser d'émettre, se mettant en attente jusqu'à ce qu'ils commencent à émettre de nouveau.

Token-Ring

La technologie Token-Ring se repose sur :

- une topologie logique en anneau,
- une topologie physique en étoile.

Token-Ring se traduit par **Anneau à Jeton**. Il n'est pas aussi répandu que l'Ethernet pour des raisons de coûts. En effet le rajout d'un nœud en Token-Ring peut coûter jusqu'à **4 fois plus cher qu'en Ethernet**.

Il faut noter que :

- les données sont transmises dans le réseau par un système appelé **méthode de passage de jeton**,
- le jeton est une **trame numérique vide** de données qui tourne en permanence dans l'anneau,
- quand un nœud souhaite émettre, il saisit le jeton, y dépose des données avec l'adresse du destinataire et ensuite laisse poursuivre son chemin

jusqu'à sa destination,
- pendant son voyage, aucun autre nœud ne peut émettre,
- une fois arrivé à sa destination, le jeton dépose ses données et retourne à l'émetteur pour confirmer la livraison,
- ce système est appelé **déterministe**.

L'intérêt de la technologie Token-Ring se trouve dans le fait :

- qu'il **évite des collisions**,
- qu'il est **possible de déterminer avec exactitude le temps que prenne l'acheminement des données**.

La technologie Token-Ring est donc idéale, voire obligatoire, dans des installations où chaque nœud doit disposer d'une opportunité à intervalle fixe d'émettre des données.

Périphériques Réseaux Spéciaux

En plus du câblage, les périphériques de réseau spéciaux sont des éléments primordiaux tant au niveau de la topologie physique que la topologie logique.

Les périphériques de réseau spéciaux sont :

- les Concentrateurs ou *Hubs*,
- les Répéteurs ou *Repeaters*,
- les Ponts ou *Bridges*,
- les Commutateurs ou *Switches*,
- les Routeurs ou *Routers*,
- les Passerelles ou *Gateways*.

L'objectif ici est de vous permettre de comprendre le rôle de chaque périphérique.

Les Concentrateurs

Les Concentrateurs permettent une connectivité entre les nœuds en topologie en étoile. Selon leur configuration, la topologie logique peut être en

étoile, en bus ou en anneau. Il existe de multiples types de Concentrateurs allant du plus simple au Concentrateur intelligent.

- **Le Concentrateur Simple**
  - est une boîte de raccordement centrale,
  - joue le rôle de récepteur et du réémetteur des signaux sans accélération ni gestion de ceux-ci,
  - est un périphérique utilisé pour des groupes de travail.
- **Le Concentrateur Évolué**
  - est un Concentrateur simple qui offre en plus l'amplification des signaux, la gestion du type de topologie logique grâce à des capacités d'être configurés à l'aide d'un logiciel ainsi que l'homogénéisation du réseau en offrant des ports pour un câblage différent. Par exemple, 8 ports en paire torsadée non-blindée et un port BNC.
- **Le Concentrateur Intelligent**
  - est un Concentrateur évolué qui offre en plus la détection automatique des pannes, la connectique avec un Pont ou un Routeur ainsi que le diagnostic et la génération de rapports.

Les Répéteurs

Un Répéteur est un périphérique réseau simple. Il est utilisé pour amplifier le signal quand :

- la longueur du câble dépasse la limite autorisée,
- le câble passe par une zone ou les interférences sont importantes.

Éventuellement, et uniquement dans le cas où le Répéteur serait muni d'une telle fonction, celui-ci peut être utiliser pour connecter deux réseaux ayant un câblage différent.

Les Ponts

Un Pont est **Répéteur intelligent**. Outre sa capacité d'amplifier les signaux, le Pont analyse le trafic qui passe par lui et met à jour une liste d'adresses des cartes réseau, appelée **une table de routage**, n'autorisant que les transmissions destinées à d'autres segments du réseau.

Les **diffusions** sont néanmoins autorisées.

Comme un Pont doit être intelligent, on utilise souvent un micro-ordinateur comme Pont. Forcément équipé de 2 cartes réseau, le Pont peut également jouer le rôle de serveur de fichiers.

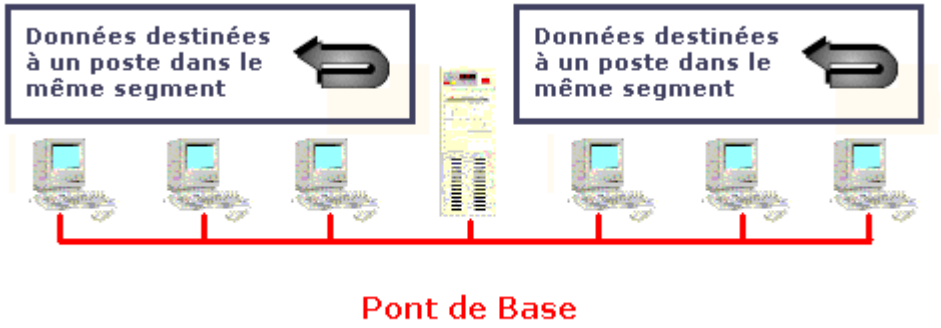
Le Pont sert donc à isoler des segments du réseau pour des raisons de :

- **sécurité** afin d'éviter à ce que des données sensibles soient propagées sur tout le réseau,
- **performance** afin qu'une partie du réseau trop chargée ralentisse le réseau entier,
- **fiabilité** afin par exemple qu'une carte en panne ne gène pas le reste du réseau avec une diffusion.

Il existe trois types de configuration de Ponts

Le Pont de Base

Le Pont de Base est utilisé très rarement pour isoler deux segments.

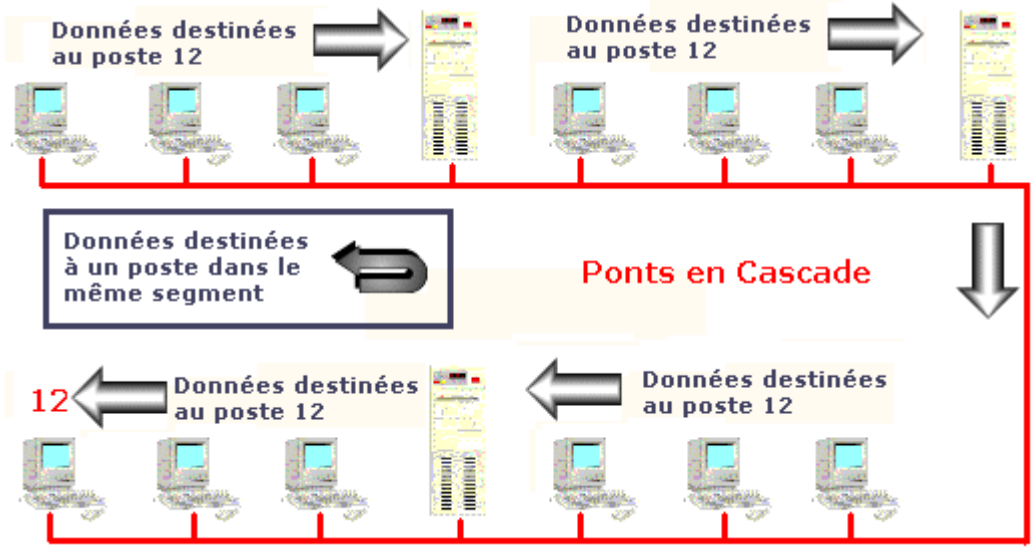


Pont de Base

;#; ;#;

Le Pont en Cascade

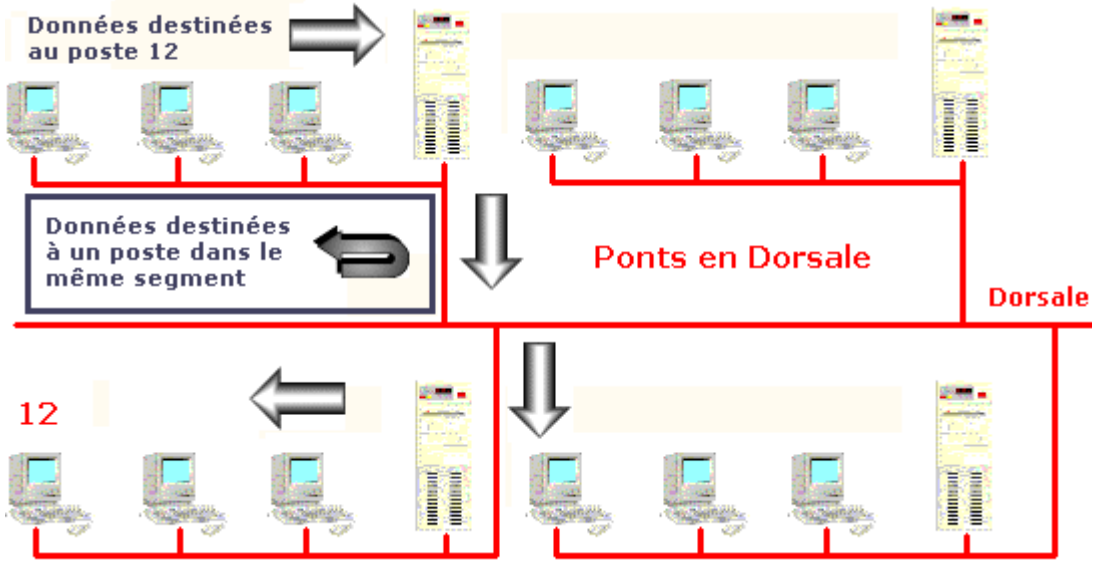
Le Pont en Cascade est à éviter car les données en provenance d'un segment doivent passer par plusieurs Ponts. Ceci a pour conséquence de ralentir la transmission des données, voire même de créer un trafic superflu en cas de rémission par le nœud



;#; ;#;

Le Pont en Dorsale

Le Pont en Dorsale coûte plus chère que la configuration précédente car il faut un nombre de Ponts équivalent au nombre de segments + 1. Par contre elle réduit les problèmes précédemment cités puisque les données ne transitent que par deux Ponts.



;#; ;#;

Les Commutateurs

Un Commutateur peut être considéré comme un Concentrateur intelligent et un Pont. Ils sont gérés souvent par des logiciels. La topologie physique d'un réseau commuté est en étoile. Par contre la topologie logique est spéciale, elle s'appelle une topologie commutée.

Lors de la communication de données entre deux nœuds, le Commutateur ouvre une connexion temporaire virtuelle en fermant les autres ports. De cette façon la bande passante totale est disponible pour cette transmission et les risques de collision sont minimisés.

Certains Commutateurs haut de gamme sont équipés d'un système anti-catastrophe qui leur permet d'isoler une partie d'un réseau en panne afin que les autres parties puissent continuer à fonctionner sans problème.

Les Routeurs

Un Routeur est un Pont sophistiqué capable :

- d'assurer l'interconnexion entre des segments,
- de filtrer le trafic,
- d'isoler une partie du réseau,
- d' explorer les informations d'adressage pour trouver le chemin le plus approprié et le plus rentable pour la transmission des données.

Les Routeurs utilisent une table de routage pour stocker les informations sur :

- les adresses du réseau,
- les solutions de connexion vers d'autres réseaux,
- l'efficacité des différentes routes.

Il existe deux types de Routeur :

- le **Routeur Statique**
  - la table de routage est éditer manuellement,
  - les routes empruntées pour la transmission des données sont toujours les mêmes,
  - il n'y a pas de recherche d'efficacité.
- le **Routeur Dynamique**
  - découvre automatiquement les routes à emprunter dans un réseau.

Les Passerelles

Ce périphérique, souvent un logiciel, sert à faire une conversion de données :

- entre deux technologies différentes (Ethernet - Token-Ring),
- entre deux protocoles différents,
- entre des formats de données différents.

Annexe #2 - Comprendre TCP Version 4

En-tête TCP

L'en-tête TCP est codée sur 4 octets soit 32 bits :

<table>
<tr><th>1er octet</th><th>2ème octet</th><th>3ème octet</th><th>4 ème octet</th></tr>
<tr><td colspan="2">Port source</td><td colspan="2">Port destination</td></tr>
<tr><td colspan="4">Numéro de séquence</td></tr>
<tr><td colspan="4">Numéro d'acquittement</td></tr>
<tr><td>Offset</td><td>Flags</td><td colspan="2">Fenêtre</td></tr>
<tr><td colspan="2">Checksum</td><td colspan="2">Pointeur Urgent</td></tr>
<tr><td colspan="3">Options</td><td>Padding</td></tr>
<tr><td colspan="4">Données</td></tr>
</table>

Vous noterez que les numéros de ports sont codés sur 16 bits. Cette information nous permet de calculer le nombres de ports maximum en IPv4, soit 2^{16} ports ou 65 535.

L'**Offset** contient la taille de l'en-tête.

Les **Flags** sont :

- URG - Si la valeur est 1 le pointeur urgent est utilisé. Le numéro de séquence et le pointeur urgent indique un octet spécifique.
- ACK - Si la valeur est 1, le paquet est un accusé de réception
- PSH - Si la valeur est 1, les données sont immédiatement présentées à l'application
- RST - Si la valeur est 1, la communication comporte un problème et la connexion est réinitialisée
- SYN - Si la valeur est 1, le paquet est un paquet de synchronisation
- FIN - Si la valeur est 1, le paquet indique la fin de la connexion

La **Fenêtre** est codée sur 16 bits. La Fenêtre est une donnée liée au fonctionnement d'expédition de données appelé le **sliding window** ou la **fenêtre glissante**. Puisque il serait impossible, pour des raisons de performance, d'attendre l'accusé de réception de chaque paquet envoyé, l'expéditeur envoie des paquets par groupe. La taille de cette groupe s'appelle la Fenêtre. Dans le cas d'un problème de réception d'une partie de la Fenêtre, toute la Fenêtre est ré-expédiée.

Le **Checksum** est une façon de calculer si le paquet est complet.

Le **Padding** est un champ pouvant être rempli de valeurs nulles de façon à ce que la taille de l'en-tête soit un multiple de 32

En-tête UDP

L'en-tête UDP est codée sur 4 octets soit 32 bits :

1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Port source		Port destination	
longueur		Checksum	
Données			

L'en-tête UDP a une longueur de 8 octets.

Fragmentation et Ré-encapsulation

La taille limite d'un paquet TCP, l'en-tête comprise, ne peut pas dépasser **65 535 octets**. Cependant chaque réseau est qualifié par son MTU (Maximum Tranfer Unit). Cette valeur est la taille maximum d'un paquet autorisée. L'unité est en **octets**. Pour un réseau Ethernet sa valeur est de 1 500. Quand un paquet doit être expédié sur un réseau ayant un MTU inférieur à sa propre taille, le paquet doit être **fractionné**. A la sortie du réseau, le paquet est reconstitué. Cette reconstitution s'appelle **ré-encapsulation**.

Adressage

L'adressage IP requière que chaque périphérique sur le réseau possède une adresse IP unique de 4 octets, soit 32 bits au format XXX.XXX.XXX.XXX De cette façon le nombre total d'adresses est de 2^{32} = 4.3 Milliards.

Les adresses IP sont divisées en 5 classes, de A à E. Les 4 octets des classes A à C sont divisés en deux, une partie qui s'appelle le **Net ID** qui identifie le réseau et une partie qui s'appelle le **Host ID** qui identifie le hôte :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
A	Net ID	Host ID		

	1er octet	2ème octet	3ème octet	4 ème octet
B	Net ID		Host ID	
C	Net ID			Host ID
D	Multicast			
E	Réservé			

L'attribution d'une classe dépend du nombre de hôtes à connecter. Chaque classe est identifié par un **Class ID** composé de 1 à 3 bits :

Classe	Bits ID Classe	Valeur ID Classe	Bits ID Réseau	Nb. de Réseaux	Bits ID hôtes	Nb. d'adresses	Octet de Départ
A	1	0	7	2^7=128	24	2^{24}=16 777 216	1 - 126
B	2	10	14	2^{14}=16 834	16	2^{16}=65 535	128 - 191
C	3	110	21	2^{21}=2 097 152	8	2^8=256	192 - 223

Dans chaque classe, certaines adresses sont réservées pour un usage privé :

Classe	IP de Départ	IP de Fin
A	10.0.0.0	10.255.255.255
B	172.16.0.0	172.31.255.255
C	192.168.0.0	192.168.255.255

Il existe des adresses particulières ne pouvant pas être utilisées pour identifier un hôte :

Adresse Particulière	Description
169.254.0.0 à 169.254.255.255	Automatic Private IP Addressing de Microsoft
Hôte du réseau courant	Tous les bits du Net ID sont à 0
Adresse de réseau	Tous les bits du Host ID sont à 0
Adresse de diffusion	Tous les bits du Host ID sont à 1

L'adresse de réseau identifie le **segment** du réseau entier tandis que l'adresse de diffusion identifie tous les hôtes sur le segment de réseau.

Afin de mieux comprendre l'adresse de réseau et l'adresse de diffusion, prenons le cas de l'adresse 192.168.10.1 en classe C :

	1er octet	2ème octet	3ème octet	4 ème octet

	1er octet	2ème octet	3ème octet	4 ème octet

	Net ID			Host ID
Adresse IP	192	168	10	1
Binaire	11000000	10101000	000001010	00000001
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	000001010	**00000000**
Adresse réseau	192	168	10	0
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	000001010	**11111111**
Adresse de diffusion	192	168	10	255

Masques de sous-réseaux

Tout comme l'adresse IP, le masque de sous-réseau compte 4 octets ou 32 bits. Les masques de sous-réseaux permettent d'identifer le Net ID et le Host ID :

Classe	Masque	Notation CIDR
A	255.0.0.0	/8
B	255.255.0.0	/16
C	255.255.255.0	/24

Le terme **CIDR** veut dire **Classless InterDomain Routing**. Le terme Notation CIDR correspond au nombre de bits d'une valeur de 1 dans le masque de sous-réseau.

Quand un hôte souhaite émettre il procède d'abord à l'identification de sa propre adresse réseau par un calcul AND (ET) appliqué à sa propre adresse et son masque de sous-réseau qui stipule :

- 1 x 1 = 1
- 0 x 1 = 0
- 1 x 0 = 0
- 0 x 0 = 0

Prenons le cas de l'adresse IP 192.168.10.1 ayant un masque de 255.255.255.0 :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	10	1
Binaire	11000000	10101000	00001010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Cet hôte essaie de communiquer avec un hôte ayant une adresse IP de 192.168.10.10. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	10	10
Binaire	11000000	10101000	00001010	00001010
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00001010	00000000
Adresse réseau	192	168	10	0

Puisque l'adresse réseau est identique dans les deux cas, l'hôte émetteur présume que l'hôte de destination se trouve sur son réseau et envoie les paquets directement sur le réseau sans s'adresser à sa passerelle par défaut.

L'hôte émetteur essaie maintenant de communiquer avec avec un hôte ayant une adresse IP de 192.168.2.1. Il procède donc au même calcul en appliquant **son propre masque de sous-réseau** à l'adresse IP de l'hôte destinataire :

	1er octet	**2ème octet**	**3ème octet**	**4 ème octet**
Adresse IP	192	168	2	1
Binaire	11000000	10101000	00000010	00000001
Masque de sous-réseau				
Binaire	11111111	11111111	11111111	00000000
Calcul AND	11000000	10101000	00000010	00000000

	1er octet	2ème octet	3ème octet	4 ème octet
Adresse réseau	192	168	2	0

Dans ce cas, l'hôte émetteur constate que le réseau de destination 192.168.2.0 n'est pas identique à son propre réseau 192.168.10.0. Il adresse donc les paquets à la passerelle par défaut.

VLSM

Puisque le stock de réseaux disponibles sous IPv4 est presque épuisé, une solution a du être trouvée pour créer des sous-réseaux en attendant l'introduction de l'IPv6. Cette solution s'appelle le VLSM ou Variable Length Subnet Masks. Le VLSM exprime les masques de sous-réseaux au format CIDR.

Son principe est simple. Afin de créer des réseaux différents à partir d'une adresse réseau d'une classe donnée, il convient de réduire le nombre d'hôtes. De cette façon les bits 'libérés' du Host ID peuvent être utilisés pour identifier les sous-réseaux.

Pour illustrer ceci, prenons l'exemple d'un réseau 192.168.1.0. Sur ce réseau, nous pouvons mettre 2^8-2 soit 254 hôtes entre 192.168.1.1 au 192.168.1.254.

Supposons que nous souhaiterions diviser notre réseau en 2 sous-réseaux. Pour coder 2 sous-réseaux, il faut que l'on libère 2 bits du Host ID. Les deux bits libérés auront les valeurs binaires suivantes :

- 00
- 01
- 10
- 11

Les valeurs binaires du quatrième octet de nos adresses de sous-réseaux seront donc :

- 192.168.1.00XXXXXX
- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX
- 192.168.1.11XXXXXX

où les XXXXXX représentent les bits que nous réservons pour décrire les hôtes dans chacun des sous-réseaux.

Nous ne pouvons pas utiliser les deux sous-réseaux suivants :

- 192.168.1.00XXXXXX
- 192.168.1.11XXXXXX

car ceux-ci correspondent aux débuts de l'adresse réseau 192.168.1.0 et de l'adresse de diffusion 192.168.1.255.

Nous pouvons utiliser les deux sous-réseaux suivants :

- 192.168.1.01XXXXXX
- 192.168.1.10XXXXXX

Pour le premier sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #1	192	168	1	01XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	01**000000**
Adresse réseau	192	168	1	**64**
Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	01**111111**
Adresse de diffusion	192	168	1	**127**

- L'adresse CIDR du réseau est donc 192.168.1.64/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.65 à 192.168.1.126

Pour le deuxième sous-réseau l'adresse réseau et l'adresse de diffusion sont :

Sous-réseau #2	192	168	1	10XXXXXX
Calcul de l'adresse de réseau				
Binaire	11000000	10101000	00000001	10**000000**
Adresse réseau	192	168	1	**128**

Calcul de l'adresse de diffusion				
Binaire	11000000	10101000	00000001	10**111111**
Adresse de diffusion	192	168	1	**191**

- L'adresse CIDR du réseau est donc 192.168.1.128/26 car le Net ID est codé sur 24+2 bits.
- Le masque de sous-réseau est donc le 11111111.11111111.11111111.11000000 ou le 255.255.255.192
- Nous pouvons avoir 2^6-2 soit 62 hôtes.
- La plage valide d'adresses IP est de 192.168.1.129 à 192.168.1.190

La valeur qui sépare les sous-réseaux est 64. Cette valeur comporte le nom **incrément**.

Ports et sockets

Afin que les données arrivent aux applications que les attendent, TCP utilise des numéros de ports sur la couche transport. Le numéros de ports sont divisés en trois groupes :

- **Well Known Ports**
  - De 1 à 1023
- **Registered Ports**
  - De 1024 à 49151
- **Dynamic** et/ou **Private Ports**
  - De 49152 à 65535

Le couple **numéro IP:numéro de port** s'appelle un **socket**.

/etc/services

Les ports les plus utilisés sont détaillés dans le fichier **/etc/services** :

```
[root@centos7 ~]# more /etc/services
```

```
# /etc/services:
# $Id: services,v 1.55 2013/04/14 ovasik Exp $
#
# Network services, Internet style
# IANA services version: last updated 2013-04-10
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
# Updated from RFC 1700, ``Assigned Numbers'' (October 1994).  Not all ports
# are included, only the more common ones.
#
# The latest IANA port assignments can be gotten from
#       http://www.iana.org/assignments/port-numbers
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# Each line describes one service, and is of the form:
#
# service-name  port/protocol  [aliases ...]   [# comment]

tcpmux          1/tcp                           # TCP port service multiplexer
tcpmux          1/udp                           # TCP port service multiplexer
rje             5/tcp                           # Remote Job Entry
rje             5/udp                           # Remote Job Entry
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
systat          11/udp          users
daytime         13/tcp
--More--(0%)
```

Notez que les ports sont listés par deux :

- le port TCP
- le port UDP

La liste la plus complète peut être consultée à l'adresse suivante **https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml**.

Pour connaitre la liste des sockets ouverts sur l'ordinateur, saisissez la commande suivante :

```
[root@centos7 ~]# netstat -an | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:7127          0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:52284         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:49669         0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:52284         127.0.0.1:46641         ESTABLISHED
tcp        0      0 10.0.2.15:22            10.0.2.2:47261          ESTABLISHED
tcp        0      0 127.0.0.1:46641         127.0.0.1:52284         ESTABLISHED
tcp6       0      0 :::22                   :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
udp        0      0 10.0.2.15:49309         10.0.2.3:53             ESTABLISHED
udp        0      0 0.0.0.0:42155           0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 127.0.0.1:323           0.0.0.0:*
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:14451           0.0.0.0:*
udp        0      0 10.0.2.15:37244         212.83.184.186:123      ESTABLISHED
udp6       0      0 ::1:323                 :::*
udp6       0      0 :::35912                :::*
raw6       0      0 :::58                   :::*                    7
```

```
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     20224    public/pickup
unix  2      [ ACC ]     STREAM     LISTENING     20228    public/cleanup
unix  2      [ ACC ]     STREAM     LISTENING     20231    public/qmgr
unix  2      [ ACC ]     STREAM     LISTENING     11278    /run/lvm/lvmpolld.socket
unix  2      [ ACC ]     STREAM     LISTENING     13838    /var/run/dbus/system_bus_s
ocket
unix  2      [ ACC ]     STREAM     LISTENING     20253    public/flush
unix  2      [ ACC ]     STREAM     LISTENING     20268    public/showq
--More--
```

Pour connaitre la liste des applications ayant ouvert un port sur l'ordinateur, saisissez la commande suivante :

```
[root@centos7 ~]# netstat -anp | more
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      855/sshd
tcp        0      0 127.0.0.1:7127          0.0.0.0:*               LISTEN      3275/Remote Access
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN      854/cupsd
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN      2214/master
tcp        0      0 127.0.0.1:52284         0.0.0.0:*               LISTEN      3389/Remote Access
tcp        0      0 127.0.0.1:49669         0.0.0.0:*               LISTEN      3275/Remote Access
tcp        0      0 127.0.0.1:52284         127.0.0.1:46641         ESTABLISHED 3389/Remote Access
tcp        0      0 10.0.2.15:22            10.0.2.2:47261          ESTABLISHED 4557/sshd: trainee
tcp        0      1 10.0.2.15:55144         86.241.135.118:443      SYN_SENT    3275/Remote Access
tcp        0      0 127.0.0.1:46641         127.0.0.1:52284         ESTABLISHED 3275/Remote Access
tcp6       0      0 :::22                   :::*                    LISTEN      855/sshd
tcp6       0      0 ::1:631                 :::*                    LISTEN      854/cupsd
udp        0      0 0.0.0.0:42155           0.0.0.0:*                           525/avahi-daemon: r
udp        0      0 0.0.0.0:5353            0.0.0.0:*                           525/avahi-daemon: r
udp        0      0 127.0.0.1:323           0.0.0.0:*                           556/chronyd
udp        0      0 0.0.0.0:68              0.0.0.0:*                           4501/dhclient
udp        0      0 0.0.0.0:14451           0.0.0.0:*                           4501/dhclient
```

```
udp        0      0 10.0.2.15:37244         212.83.184.186:123      ESTABLISHED 556/chronyd
udp6       0      0 ::1:323                 :::*                                556/chronyd
udp6       0      0 :::35912                :::*                                4501/dhclient
raw6       0      0 :::58                   :::*                    7           653/NetworkManager
Active UNIX domain sockets (servers and established)
Proto RefCnt Flags       Type       State         I-Node   PID/Program name     Path
unix  2      [ ACC ]     STREAM     LISTENING     20224    2214/master          public/pickup
unix  2      [ ACC ]     STREAM     LISTENING     20228    2214/master          public/cleanup
unix  2      [ ACC ]     STREAM     LISTENING     20231    2214/master          public/qmgr
unix  2      [ ACC ]     STREAM     LISTENING     11278    1/systemd            /run/lvm/lvmpolld.socket
unix  2      [ ACC ]     STREAM     LISTENING     13838    1/systemd            /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     STREAM     LISTENING     20253    2214/master          public/flush
unix  2      [ ACC ]     STREAM     LISTENING     20268    2214/master          public/showq
unix  2      [ ACC ]     STREAM     LISTENING     13859    1/systemd            /var/run/rpcbind.sock
--More--
```

Résolution d'adresses Ethernet

Chaque protocole peut être encapsulé dans une **trame** Ethernet. Lorsque la trame doit être transportée de l'expéditeur au destinataire, ce premier doit connaitre l'adresse Ethernet du dernier. L'adresse Ethernet est aussi appelée l'adresse **Physique** ou l'adresse **MAC**.

Pour connaître l'adresse Ethernet du destinataire, l'expéditeur fait appel au protocol **ARP**. Les informations reçues sont stockées dans une table. Pour visualiser ces informations, il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# arp -a
? (10.0.2.3) at 52:54:00:12:35:03 [ether] on enp0s3
gateway (10.0.2.2) at 52:54:00:12:35:02 [ether] on enp0s3
```

Options de la commande

Les options de cette commande sont :

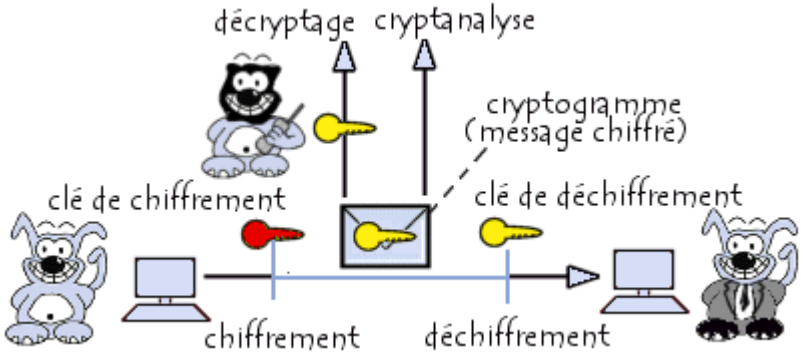
```
[root@centos7 ~]# arp --help
Usage:
  arp [-vn]  [<HW>] [-i <if>] [-a] [<hostname>]             <-Display ARP cache
  arp [-v]          [-i <if>] -d  <host> [pub]               <-Delete ARP entry
  arp [-vnD] [<HW>] [-i <if>] -f  [<filename>]            <-Add entry from file
  arp [-v]   [<HW>] [-i <if>] -s  <host> <hwaddr> [temp]            <-Add entry
  arp [-v]   [<HW>] [-i <if>] -Ds <host> <if> [netmask <nm>] pub          <-''-

        -a                      display (all) hosts in alternative (BSD) style
        -e                      display (all) hosts in default (Linux) style
        -s, --set               set a new ARP entry
        -d, --delete            delete a specified entry
        -v, --verbose           be verbose
        -n, --numeric           don't resolve names
        -i, --device            specify network interface (e.g. eth0)
        -D, --use-device        read <hwaddr> from given device
        -A, -p, --protocol      specify protocol family
        -f, --file              read new entries from file or from /etc/ethers

  <HW>=Use '-H <hw>' to specify hardware address type. Default: ether
  List of possible hardware types (which support ARP):
    ash (Ash) ether (Ethernet) ax25 (AMPR AX.25)
    netrom (AMPR NET/ROM) rose (AMPR ROSE) arcnet (ARCnet)
    dlci (Frame Relay DLCI) fddi (Fiber Distributed Data Interface) hippi (HIPPI)
    irda (IrLAP) x25 (generic X.25) infiniband (InfiniBand)
    eui64 (Generic EUI-64)
```

Annexe #3 - Comprendre le Chiffrement

Introduction à la cryptologie

Définitions

- **La Cryptologie**
  - La science qui étudie les aspects scientifiques de ces techniques, c'est-à-dire qu'elle englobe la cryptographie et la cryptanalyse.
- **La Cryptanalyse**
  - Lorsque la clef de déchiffrement n'est pas connue de l'attaquant on parle alors de cryptanalyse ou cryptoanalyse (on entend souvent aussi le terme plus familier de cassage).
- **La Cryptographie**
  - Un terme générique désignant l'ensemble des techniques permettant de chiffrer des messages, c'est-à-dire permettant de les rendre inintelligibles sans une action spécifique. Les verbes crypter et chiffrer sont utilisés.
- **Le Décryptement ou Décryptage**
  - Est le fait d'essayer de déchiffrer illégitimement le message (que la clé de déchiffrement soit connue ou non de l'attaquant).



La Cryptographie

La cryptographie apporte quatre points clefs:

- La confidentialité
  - consiste à rendre l'information inintelligible à d'autres personnes que les acteurs de la transaction.
- L'intégrité
  - consiste à déterminer si les données n'ont pas été altérées durant la communication (de manière fortuite ou intentionnelle).
- L'authentification

    - consiste à assurer l'identité d'un utilisateur.
- La non-répudiation
    - est la garantie qu'aucun des correspondants ne pourra nier la transaction.

La cryptographie est basée sur l'arithmétique. Il s'agit, dans le cas d'un texte, de transformer les lettres qui composent le message en une succession de chiffres (sous forme de bits dans le cas de l'informatique), puis ensuite de faire des calculs sur ces chiffres pour:

- Procéder au chiffrement
    - Le résultat de cette modification (le message chiffré) est appelé cryptogramme (Ciphertext) par opposition au message initial, appelé message en clair (Plaintext)
- Procéder au déchiffrement

Le chiffrement se fait à l'aide d'une clef de chiffrement. Le déchiffrement nécessite une clef de déchiffrement.

On distingue deux types de clefs:

- Les clés symétriques:
    - des clés utilisées pour le chiffrement ainsi que pour le déchiffrement. On parle alors de chiffrement symétrique ou de chiffrement à clé secrète.
- Les clés asymétriques:
    - des clés utilisées dans le cas du chiffrement asymétrique (aussi appelé chiffrement à clé publique). Dans ce cas, une clé différente est utilisée pour le chiffrement et pour le déchiffrement.

Le Chiffrement par Substitution

Le chiffrement par substitution consiste à remplacer dans un message une ou plusieurs entités (généralement des lettres) par une ou plusieurs autres entités. On distingue généralement plusieurs types de cryptosystèmes par substitution :

- La substitution **monoalphabétique**
    - consiste à remplacer chaque lettre du message par une autre lettre de l'alphabet
- La substitution **polyalphabétique**
    - consiste à utiliser une suite de chiffres monoalphabétique réutilisée périodiquement
- La substitution **homophonique**
    - permet de faire correspondre à chaque lettre du message en clair un ensemble possible d'autres caractères

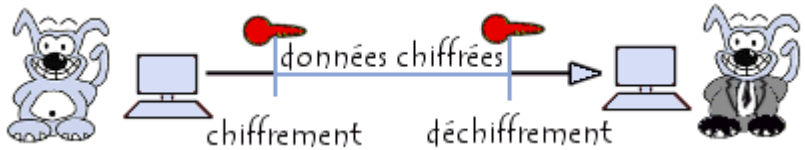
- La substitution de **polygrammes**
  - consiste à substituer un groupe de caractères (polygramme) dans le message par un autre groupe de caractères

Algorithmes à clé secrète

Le Chiffrement Symétrique

Ce système est aussi appelé le système à **Clef Secrète** ou à **clef privée**.

Ce système consiste à effectuer une opération de chiffrement par algorithme mais comporte un inconvénient, à savoir qu'il nécessite un canal sécurisé pour la transmission de la clef de chiffrement/déchiffrement.



> Le système de Méthode du Masque Jetable (One Time Pad) fût mis au point dans les années 1920. Il utilisait une clef générée aléatoirement à usage unique.

Les algorithmes de chiffrement symétrique couramment utilisés en informatique sont:

- **Data Encryption Standard** (DES),
- **Triple DES** (3DES),
- **RC2**,
- **Blowfish**,
- **International Data Encryption Algorithm** (IDEA),
- **Advanced Encryption Standard** (AES).

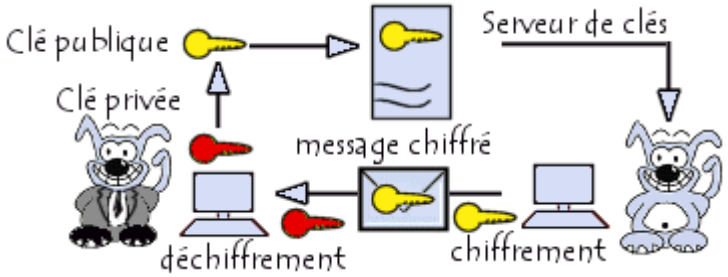
Algorithmes à clef publique

Le Chiffrement Asymétrique

Ce système est aussi appelé **Système à Clef Publique**.

Ce système consiste à avoir deux clefs appelées des **bi-clefs**:

- Une clef **publique** pour le chiffrement
- Une clef **secrète** ou **privée** pour le déchiffrement



- L'utilisateur A (celui qui déchiffre) choisit une clef privée.
- A partir de cette clef il génère plusieurs clefs publiques grâce à un algorithme.
- L'utilisateur B (celui qui chiffre) choisit une des clefs publiques à travers un canal non-sécurisé pour chiffrer les données à l'attention de l'utilisateur A.

Ce système est basé sur ce que l'on appelle une **fonction à trappe à sens unique** ou **one-way trap door**.

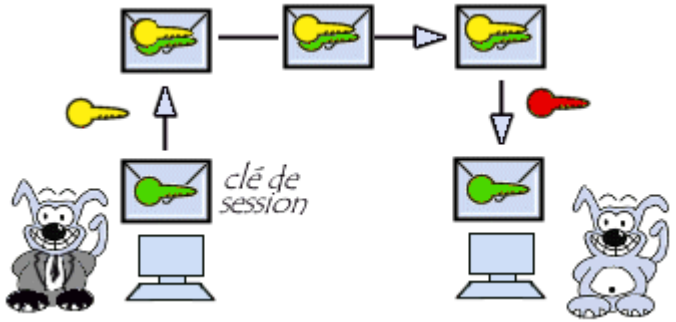
Il existe toutefois un problème – s'assurer que la clef publique récupérée est bien celle qui correspond au destinataire !

Les algorithmes de chiffrement asymétrique couramment utilisés en informatique sont:

- **Digital Signature Algorithm** (DSA)
- **Rivest, Shamir, Adleman** (RSA)

La Clef de Session

Ce système est un compromis entre le système symétrique et le système asymétrique. Il permet l'envoie de données chiffrées à l'aide d'un algorithme de chiffrement symétrique par un canal non-sécurisé et a été mis au point pour palier au problème de lenteur de déchiffrement du système asymétrique.

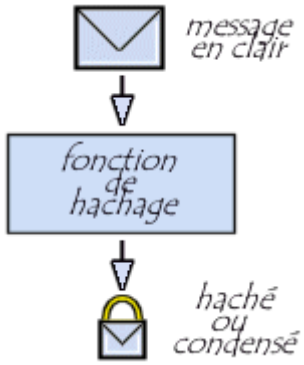


Ce système fonctionne de la façon suivante :

- L'utilisateur A chiffre une clef privée générée aléatoirement, appelée une « clef de session », en utilisant une des clefs publiques de l'utilisateur B.
- L'utilisateur A chiffre les données avec la clef de session.
- L'utilisateur B déchiffre la clef de session en utilisant sa propre clef privée.
- L'utilisateur B déchiffre les données en utilisant la clef de session.

Fonctions de Hachage

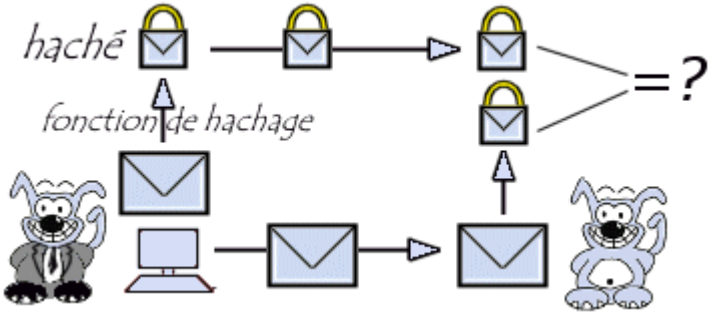
La fonction de **hachage**, aussi appelée une fonction de **condensation**, est à **sens unique** (one way function). Il « condense » un message en clair et produit un haché unique.



Les deux algorithmes de hachage utilisés sont:

- **Message Digest 5** (MD5)
- **Secure Hash Algorithm** (SHA)

Lors de son envoie, le message est accompagné de son haché et il est donc possible de garantir son intégrité:

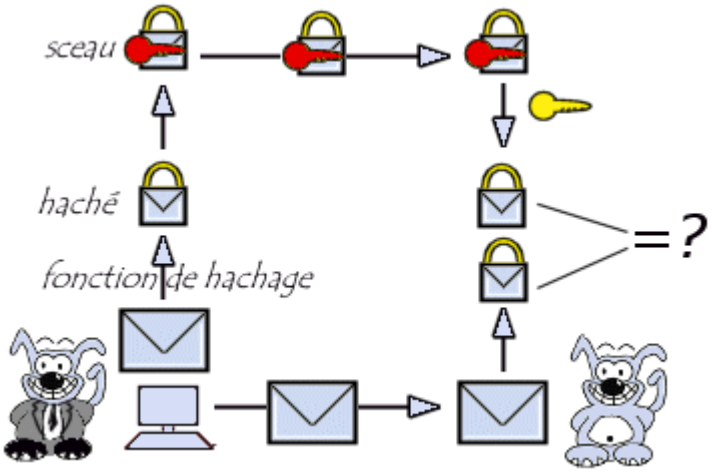


- A la réception du message, le destinataire ou l'utilisateur B calcule le haché du message reçu et le compare avec le haché accompagnant le document.
- Si le message ou le haché a été falsifié durant la communication, les deux empreintes ne correspondront pas.

> Ce système permet de vérifier que l'empreinte correspond bien au message reçu, mais ne permet pas de prouver que le message a bien été envoyé par l'utilisateur A.

Signature Numérique

Pour garantir l'authentification du message l'utilisateur A va chiffrer ou **signer** le haché à l'aide de sa clé privée. Le haché signé est appelé un **sceau**.



- L'utilisateur A envoie le sceau au destinataire.
- A la réception du message L'utilisateur B déchiffre le sceau avec la clé publique de l'utilisateur A.
- Il compare le haché obtenu au haché reçu en pièce jointe.

Ce mécanisme de création de sceau est appelé **scellement**.

Ce mécanisme est identique au procédé utilisé par SSH lors d'une connexion

Utilisation de GnuPG

Présentation

GNU Privacy Guard permet aux utilisateurs de transférer des messages chiffrés et/ou signés.

Installation

Sous RHEL/CentOS 7, le paquet gnupg est installé par défaut :

```
[root@centos7 ~]# whereis gpg
gpg: /usr/bin/gpg /usr/share/man/man1/gpg.1.gz
```

Configuration

Pour initialiser GnuPG, saisissez la commande suivante :

```
[root@centos7 ~]# gpg
gpg: directory `/root/.gnupg' created
gpg: new configuration file `/root/.gnupg/gpg.conf' created
gpg: WARNING: options in `/root/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring `/root/.gnupg/secring.gpg' created
gpg: keyring `/root/.gnupg/pubring.gpg' created
gpg: Go ahead and type your message ...
^C
gpg: signal Interrupt caught ... exiting
```

Pour générer les clefs, saisissez la commande suivante :

```
[root@centos7 ~]# gpg --gen-key
gpg (GnuPG) 2.0.22; Copyright (C) 2013 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
```

```
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0)
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: I2TCH
Email address: infos@i2tch.eu
Comment: Test Key
You selected this USER-ID:
    "I2TCH (Test Key) <infos@i2tch.eu>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
```

```
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key F6A5B400 marked as ultimately trusted
public and secret key created and signed.

gpg: checking the trustdb
gpg: 3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0  valid:   1  signed:   0  trust: 0-, 0q, 0n, 0m, 0f, 1u
pub   2048R/F6A5B400 2016-08-10
      Key fingerprint = CA95 0CB9 859B 2F80 B8AF  8C07 5365 C618 F6A5 B400
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/700F1CD5 2016-08-10
```

La liste de clefs peut être visualisée avec la commande suivante :

```
[root@centos7 ~]# gpg --list-keys
/root/.gnupg/pubring.gpg
------------------------
pub   2048R/F6A5B400 2016-08-10
uid                  I2TCH (Test Key) <infos@i2tch.eu>
sub   2048R/700F1CD5 2016-08-10
```

Pour importer la clef d'un correspondant dans sa trousse de clefs il convient d'utiliser la commande suivante :

```
# gpg --import la-clef.asc
```

Pour exporter sa clef publique, il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# gpg --export --armor I2TCH > ~/I2TCH.asc
[root@centos7 ~]# cat I2TCH.asc
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.22 (GNU/Linux)
```

```
mQENBFeqtJkBCACgQWXgjwnOMvsI1KDgyCRrg3MZmjcvU9SBh+lEEEWfqY7MKXjs
PsXN/MHjQIWSptVe0OjVQhQWDicIEiVQ6nxV+jqx5Q9fqxmjsOrzIQFadsF5LS/9
LR9KcXdBoSORGXMIsRKMzfS6oyK3yi5jM65P6jpH6PnyaX1PvlgjASh7F8OGoU8y
46WW/nW4DEN3MsX2FBtwY6aMcD5+Nvb8tJrQGz/DCrd045DRAR15mA8SVPPHSRMY
v3UzxXeRKxF/NDlecGubP1xfvvqkg/+hxUuybAc6MBhDQKvwL/ZpCoBlUCiouFPA
T9bYfagX2LHVBanY/mtIVhYDygtdc37hovEzABEBAAG0IUkyVENIIChUZXN0IEtl
eSkgPGluZm9zQGkydGNoLmV1PokBOQQTAQIAIwUCV6q0mQIbAwcLCQgHAwIBBhUI
AgkKCwQWAgMBAh4BAheAAAoJEFNlxhj2pbQAYW8H/iEg51IkqWYFSDBDWUljK3Sv
vvVGdQhqc7UptgYyqCWfegngdBH+2ScB8kbM8QVGX8kJ+xT4SeiV/VJdN6sHIkHV
sHBj5x77E5QVaU2Oz0cOlvg9cyGuQ5Y5sBN0MYEp7Q+mHpCTFhaCj8zwQ1/ZHdlK
Qk/8nNk/k2A94BJCwyQqITRWlOTYj0JGzgZV8yU5ROISsk+L6Pi4IHJw0+ZXlPXv
b0bG4p4mEmnhzZcfJ3runLJlCNrMRrWvBkkkGK/djuXDIR7HStQQlreSPG/m1JRF
hP/SARssyIWxyABSYOjR214jVLhC/pQvnbuVguYJoR6tdYqiui7oq+HBHXkP5pe5
AQ0EV6q0mQEIAOSMChRKnaZgOLzh0FN7jCJT8z8xqj1Qze8F4Vz6nS+X0Rk38lQt
ICRSMJFm6CzyPbQamAAIPshT256brXF0jPp9vpirJn/bPsiT+rtu0dl5QyWJqcc/
fE+/HaMVOuQ9HrplxV/heBqyPSA8BCppbAMti8i2DK7pNqe1JJ7CRxG0nakDSEgK
QbyrGjZYm0q2c9zb9QSbzxg/aKX2D9dlHUpNl4dhJ231d0OhBMQoW6psJjIrzfHd
dpgAYycgennv7Ik8+CIOjgb+Gl5AewLYCMFKCx/mV6/UkzXhmWw/o/POKsRFQJM+
glXAr6ddQRhk6L482R3qkWTlQHx62KAr7BMAEQEAAYkBHwQYAQIACQUCV6q0mQIb
DAAKCRBTZcYY9qW0AF60B/9tpW1Bq8GyDN6kpj+of/b8xu37A4v6Ws43feuT7cNc
EuCi+EB6wyQ3dGSgQ9BUrO2KbF0OtjYxak3FWrFkmoFucvraxC2xQZuoqN+Qtydf
utmC6V4dCOsp3NWkuCBlUN4axI+m2M2tgTn06iDDkW0ZTISxMqapHjzIi43ufJrD
1RBPjl8BBvxSqcceZsybqNre8u9xka2fXW8xMTJr14xeYITd/YJbJ5UkpU0xmzJz
hR6B8Nji4yDplTZJtz8yJ+vOlg4p7TBN6O/BCvf83q9DfmhtlE8sYsoQ1dHhPNPR
VjdGSFRo2huGFd2KNCleilRVI3xcnrR9S7ziRJu9KG9H
=9R5l
-----END PGP PUBLIC KEY BLOCK-----
```

Cette clef peut ensuite être jointe à des messages électroniques ou bien déposée sur un serveur de clefs tel http://www.keyserver.net.

Signer un message

Créez maintenant un message à signer :

```
[root@centos7 ~]# vi ~/message.txt
[root@centos7 ~]# cat ~/message.txt
This is a test message for gpg
```

Pour signer ce message en format binaire, il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# gpg --default-key I2TCH --detach-sign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID F6A5B400, created 2016-08-10

[root@centos7 ~]# ls -l | grep message
-rw-r--r--. 1 root    root          31 Aug 10 07:14 message.txt
-rw-r--r--. 1 root    root         287 Aug 10 07:16 message.txt.sig

[root@centos7 ~]# cat message.txt.sig
□□□□□□□□W□□□
    □Se□□□□□,□|□□D=□□□X□@N□□□□□|□k!□M[□□□yh□□□p□□□□□□□}(C$□□□□□Y□□d:□□E□i^-□□□W{S□
□□2□;□□□`□yj□□9□□□]□b□{□□`□□□□□□□□□□□□□□□|*□%□□R[root@centos7 ~]# □□rt□□□□□□;□_S9z□□□□HK□□□Hq□□!□G□$□□□□2□=□Scc=
```

Pour signer ce message en format ascii, il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# gpg --default-key I2TCH --armor --detach-sign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID F6A5B400, created 2016-08-10

[root@centos7 ~]# ls -l | grep message
-rw-r--r--. 1 root    root          31 Aug 10 07:14 message.txt
-rw-r--r--. 1 root    root         490 Aug 10 07:17 message.txt.asc
-rw-r--r--. 1 root    root         287 Aug 10 07:16 message.txt.sig
```

```
[root@centos7 ~]# cat message.txt.asc
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.22 (GNU/Linux)

iQEcBAABAgAGBQJXqrkDAAoJEFNlxhj2pbQATwoH/0oAm1YkXH3cfnlZef+qAc3X
AfFZz8lfwSlrBYwgDA/vd98tJNLv8VxYLRu02JcRBHqjV/LYqfCACoLKyYWCUzT/
NZiOPZDVaEpOx1vLIbmBxGclfFtbvSiZj/eBrReE2tAnnTBSTPLH58kPAMEmVgM6
Io8BPSnZvOlNhYQrPsGd046SLPRu8hTozwtB47Do6B6RazzpGLG7zOD1JZP56eD7
oo3+1HxYdv4arVgjb/bfyCNtvyPyQm+sTZPYL3vfAjkHfBQAaoAVSPRKRLgUMPD0
xrlhOU0PhK1+0pF8nVf/jt+SgCiGU8Jg1zEKCqSBuIT2Acs/kZIMo75Qo9zE2C4=
=rO70
-----END PGP SIGNATURE-----
```

Pour vérifier la signature d'un message signé en mode ascii, il convient d'utiliser la commande :

```
[root@centos7 ~]# gpg --verify message.txt.asc
gpg: Signature made Wed 10 Aug 2016 07:43:49 CEST using RSA key ID F6A5B400
gpg: Good signature from "I2TCH (Test Key) <infos@i2tch.eu>"
```

> Pour vérifier la signature d'un message signé en mode ascii et produit en dehors du message lui-même, il convient d'utiliser la commande :
>
> ```
> # gpg --verify message.txt.asc message.txt
> ```

Pour signer ce message **dans le message lui-même** en format ascii, il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# gpg --default-key I2TCH --clearsign message.txt

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID F6A5B400, created 2016-08-10
```

```
File `message.txt.asc' exists. Overwrite? (y/N) y

[root@centos7 ~]# ls -l | grep message
-rw-r--r--. 1 root    root          31 Aug 10 07:14 message.txt
-rw-r--r--. 1 root    root         568 Aug 10 07:43 message.txt.asc
-rw-r--r--. 1 root    root         287 Aug 10 07:16 message.txt.sig

[root@centos7 ~]# cat message.txt.asc
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

This is a test message for gpg
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v2.0.22 (GNU/Linux)

iQEcBAEBAgAGBQJXqr8VAAoJEFNlxhj2pbQAQ3cH+wemHfA6SoMOakxzno0iJ5ry
yROrwTm2clnEbN2zJ7rWzwRT6YtPU4mFgDyjL6G7TzOU5o9AI1dfm2iZ3kdJKmgQ
ug1F9SfhtGjltnPB46keYnzthaHNzwLwNJtv2pgxcfhO9gbFWH4FCjMRAGm9S4Hl
okF/xKVVoQzK/n/OyelUJJ6GzfNnoS75bv1WbFlie2+KlTs1MEZGZK4HiZKeXUM5
8Z4wPBKy3AlcQlZdW9rScbyHjAeyQ/yFR8Bnax6m1MK7fJv3XoaDgRegENrGwvRN
YHV7kmFU3X/ew8l85FW3q1URjKxAZLqzYRXjNRoFs67yZYTNGqcZvP3BWlefpTw=
=JTVm
-----END PGP SIGNATURE-----
```

Chiffrer un message

Pour chiffrer un message, il faut disposer de la clef publique du destinataire du message. Ce dernier utilisera ensuite sa clef privée pour déchiffrer le message. Il convient de préciser le destinataire du message, ou plus précisément la clef publique à utiliser, lors d'un chiffrement :

```
  gpg --recipient <destinataire> --encrypt <message>
```

- <*destinataire*> représente toute information permettant de distinguer sans ambigüité une clef publique dans votre trousseau. Cette information peut-être le nom ou l'adresse email associé à la clef publique que vous voulez utiliser,

- <*message*> représente le message à chiffrer.

Par exemple pour chiffrer un message en mode binaire, il convient de saisir la commande suivante :

```
[root@centos7 ~]# gpg --recipient I2TCH --encrypt message.txt

[root@centos7 ~]# ls -l | grep message
-rw-r--r--. 1 root    root          31 Aug 10 07:14 message.txt
-rw-r--r--. 1 root    root         568 Aug 10 07:43 message.txt.asc
-rw-r--r--. 1 root    root         368 Aug 10 07:47 message.txt.gpg
-rw-r--r--. 1 root    root         287 Aug 10 07:16 message.txt.sig

[root@centos7 ~]# cat message.txt.gpg
□
  □□q3p□□□□□?□j□*
□□□□X□□VL□□_~7□□□_*□u□□□BD□□R□□E□%□□!□j□□□□X□$□□aL□F0[□)\N□□$&□S□1□□□i□c□□□□□Pt□□ìH#□□9□□□l□Re□□□M□vX9Wi□K□□□h□□
ě□WK□S5□i□H□□□U□□□/"□□c~,□□Mm□□□□`□□□□(□□_□□E□□|Q□5;□□e□□L"□cLQ□□+□□□□/fhrS□E□□F□□□3□□□□&□□□□□□□Pk□*□□$H□□5␦□□R]
□□□□rM□□□□□Mt9□*□
```

Et pour chiffrer un message en mode ascii, il convient de saisir la commande suivante :

```
[root@centos7 ~]# gpg --recipient I2TCH --armor --encrypt message.txt
File `message.txt.asc' exists. Overwrite? (y/N) y

[root@centos7 ~]# ls -l | grep message
-rw-r--r--. 1 root    root          31 Aug 10 07:14 message.txt
-rw-r--r--. 1 root    root         596 Aug 10 07:49 message.txt.asc
-rw-r--r--. 1 root    root         368 Aug 10 07:47 message.txt.gpg
-rw-r--r--. 1 root    root         287 Aug 10 07:16 message.txt.sig

[root@centos7 ~]# cat message.txt.asc
-----BEGIN PGP MESSAGE-----
Version: GnuPG v2.0.22 (GNU/Linux)
```

```
hQEMA8ZxMwBwDxzVAQf/TAyZ0QI5NKvhQTQC5bAuALrxIXnX0t7yL5ARJ1A4qeE9
vzKPBj7IJHANmW5t9Is+zq1fjdmNVBl7rDw9fLEHGXVARhWlyhMUPHdw2XPSE+VT
0Vzg89w/g5G6eirmKsvDDZq3jm3c/k1w0BrAH6nowAsNuQwoesDr2fazOYVZH+OA
BHr8aslUp06VEOC7dy9gXy7o0Q5Ycb94uM7wC/ByqP2a4sJG10MMUxdSw7vk53/n
qdiIw0oCdhxNcirjSer3ZzHmqeSeQp6Sl424WuV1VZLnQXvmm084h3Z73kfBbeQc
BJfGqDWIv0pNb/5hn+LOdYn+8JZFguKu+H6ah//ogtJeAbg4kocR6zQzdMp1m8lY
p3h4HgfllK85X+WCQBcTgVaY7t0FHEkfQTrF3oYJI5kkRRnBvHKsKSN1fltKauBc
tmT2G6lZTHO+YRUItKjlAti21hVuRw1gUierqy97Rg==
=NeW+
-----END PGP MESSAGE-----
```

Pour décrypter un message il convient d'utiliser la commande suivante :

```
[root@centos7 ~]# gpg --decrypt message.txt.asc

You need a passphrase to unlock the secret key for
user: "I2TCH (Test Key) <infos@i2tch.eu>"
2048-bit RSA key, ID 700F1CD5, created 2016-08-10 (main key ID F6A5B400)

gpg: encrypted with 2048-bit RSA key, ID 700F1CD5, created 2016-08-10
      "I2TCH (Test Key) <infos@i2tch.eu>"
This is a test message for gpg
```

PKI

On appelle **PKI** (Public Key Infrastucture, ou en français **infrastructure à clé publique (ICP)**, parfois **infrastructure de gestion de clés (IGC)**) l'ensemble des solutions techniques basées sur la cryptographie à clé publique.

Les cryptosystèmes à clés publiques permettent de s'affranchir de la nécessité d'avoir recours systématiquement à un canal sécurisé pour s'échanger les clés. En revanche, la publication de la clé publique à grande échelle doit se faire en toute confiance pour assurer que :

- La clé publique est bien celle de son propriétaire ;
- Le propriétaire de la clé est digne de confiance ;

- La clé est toujours valide.

Ainsi, il est nécessaire d'associer au bi-clé (ensemble clé publique / clé privée) un certificat délivré par un **tiers de confiance** : l'infrastructure de gestion de clés.

Le tiers de confiance est une entité appelée communément autorité de certification (ou en anglais Certification authority, abrégé CA) chargée d'assurer la véracité des informations contenues dans le certificat de clé publique et de sa validité.

Pour ce faire, l'autorité signe le certificat de clé publique à l'aide de sa propre clé en utilisant le principe de signature numérique.

Le rôle de l'infrastructure de clés publiques est multiple et couvre notamment les champs suivants :

- enregistrer des demandes de clés en vérifiant l'identité des demandeurs ;
- générer les paires de clés (clé privée / clé publique) ;
- garantir la confidentialité des clés privées correspondant aux clés publiques ;
- certifier l'association entre chaque utilisateurs et sa clé publique ;
- révoquer des clés (en cas de perte par son propriétaire, d'expiration de sa date de validité ou de compromission).

Une infrastructure à clé publique est en règle générale composée de trois entités distinctes :

- L'autorité d'enregistrement (AE ou RA pour Recording authority), chargée des formalité administratives telles que la vérification de l'identité des demandeurs, le suivi et la gestion des demandes, etc.) ;
- L'autorité de certification (AC ou CA pour Certification Authority), chargée des tâches techniques de création de certificats. L'autorité de certification est ainsi chargée de la signature des demandes de certificat (CSR pour Certificate Signing Request, parfois appelées PKCS#10, nom du format correspondant). L'autorité de certification a également pour mission la signature des listes de révocations (CRL pour Certificate Revocation List) ;
- L'Autorité de dépôt (Repository) dont la mission est de conserver en sécurité les certificats.

Certificats X509

Pour palier aux problèmes liés à des clefs publiques piratées, un système de certificats a été mis en place.

Le certificat permet d’associer la clef publique à une entité ou une personne. Les certificats sont délivrés par des Organismes de Certification.

Les certificats sont des fichiers divisés en deux parties :

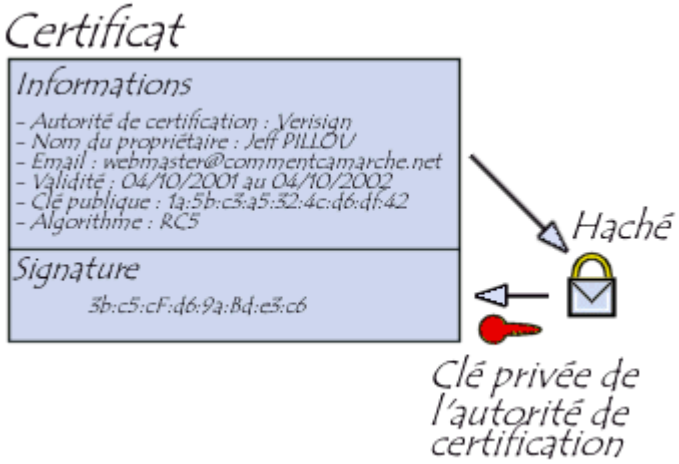
- La partie contenant les informations
- La partie contenant la signature de l'autorité de certification

La structure des certificats est normalisée par le standard **X.509** de l'**Union internationale des télécommunications**.

Elle contient :

- Le nom de l'autorité de certification
- Le nom du propriétaire du certificat
- La date de validité du certificat
- L'algorithme de chiffrement utilisé
- La clé publique du propriétaire

Le Certificat est signé par l'autorité de certification:



La vérification se passe ainsi: