# Cours de réseaux

**M1 Informatique**

**Faculté Jean Perrin**

# Plan du cours

Partie 1 : Introduction

Partie 2 : Couche physique

Partie 3 : Couche liaison des données

Partie 4 : Couche Réseau / IPv4

Partie 5 : Couche Réseau / Routage

Partie 6 : Couche Réseau / IPv4, IPv6

Partie 7 : Couche transport : TCP et UDP

Partie 8 : Couche application

Partie 9 : Couche application / Etude de protocoles

Partie 10 : Notions d'attaques et de sécurité

# Partie 9
# Couche application
# Etudes de protocoles

# Vous êtes ici

| Modèle OSI | |
|---|---|
| 7 | Application |
| 6 | Présentation |
| 5 | Session |
| 4 | Transport |
| 3 | Réseau |
| 2 | Liaison |
| 1 | Physique |

| TCP/IP |
|---|
| *Applications*<br>*Services Internet* |
| *Transport (TCP)* |
| *Internet (IP)* |
| *Accès au Réseau* |

# Services et protocole Telnet

Telnet date du début des années 70

Il est l'un des plus anciens protocoles de la couche application de la suite TCP/IP.

Permettait aux utilisateurs d'accéder à distance aux systèmes informatiques comme ils le faisaient avec les terminaux texte directement connectés.

Telnet est un protocole de base sur lequel s'appuient d'autres protocoles (FTP, SMTP, POP3,…)

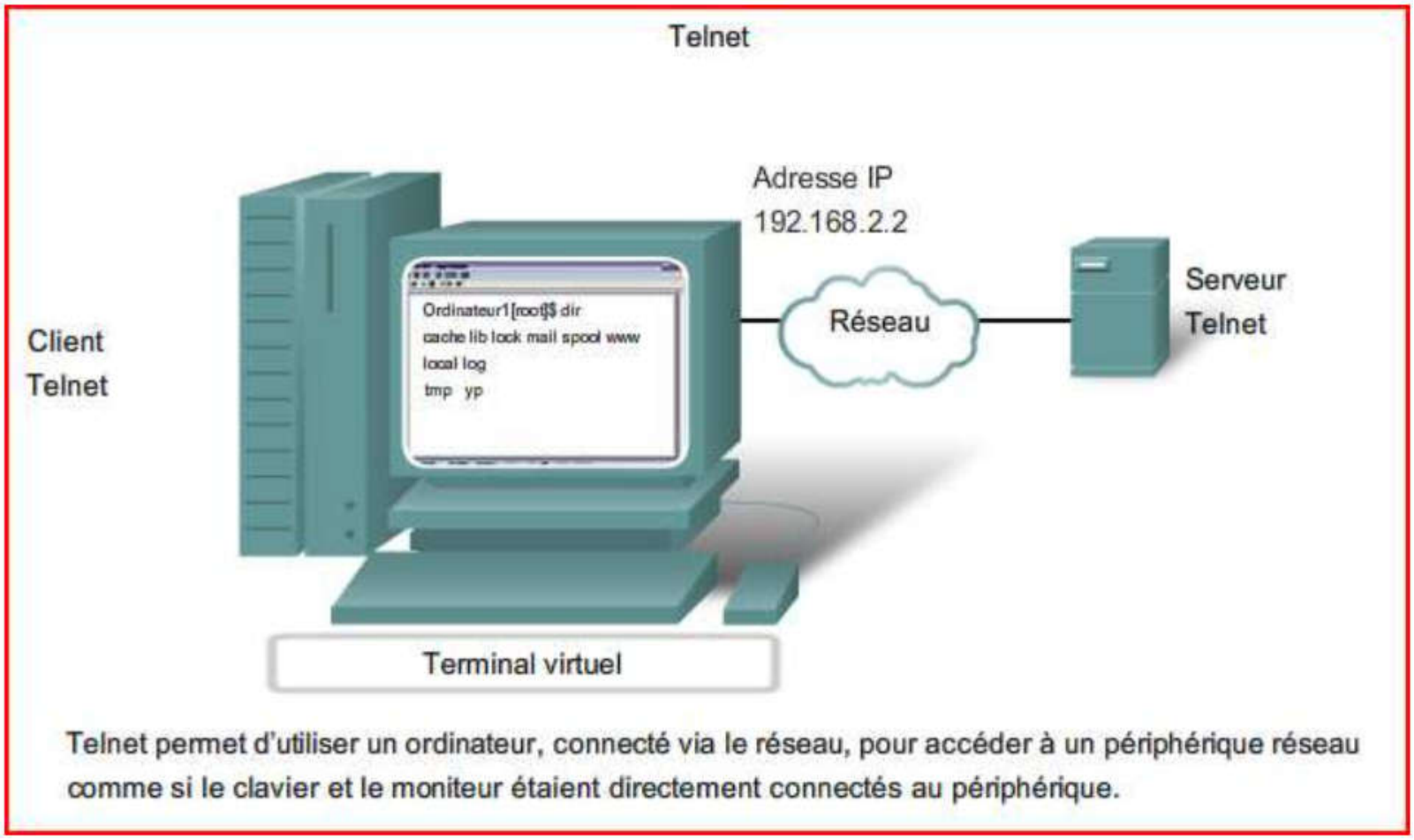
# Services et protocole Telnet

Telnet offre une méthode standard permettant d'émuler les périphériques terminaux texte via le réseau.

Telnet désigne à la fois le protocole et le logiciel client.

Une connexion utilisant Telnet est appelée connexion ou **session VTY** : Virtual teletype

# Services et protocole Telnet




Telnet permet d'utiliser un ordinateur, connecté via le réseau, pour accéder à un périphérique réseau comme si le clavier et le moniteur étaient directement connectés au périphérique.

# Services et protocole Telnet

## Protocole Telnet

Protocole du type client/serveur s'appuyant sur une connexion TCP pour envoyer des données au format ASCII (codés sur 1 octet)

Il définit la manière dont une session VTY s'établit et prend fin (syntaxe et ordre des commandes).

Fournit la liste des commandes exécutables pendant une session.

Se base sur des principes d'options négociées et des règles de négociation.

# Services et protocole Telnet

## Protocole Telnet

Chaque **commande** Telnet est constituée de 2 octets.

Le premier octet est un caractère spécial nommé IAC (Interpret As Command) : il introduit l'octet suivant en tant que commande plutôt que texte.

# Services et protocole Telnet

## Sécurité du protocole Telnet

Le protocole prend en charge l'authentification de l'utilisateur.

Il ne prend pas en charge le transport des données chiffrées : les données échangées sont transportées en clair sur le réseau.

Cela peut poser problème avec les r-commandes BSD (rlogin, rsh, rexec) : login et mot de passe circulent alors en clair…

Le protocole SSH (Secure Shell) fournit une méthode alternative sécurisée avec **authentification plus forte et transport chiffré**.

# Protocole FTP

File Transfert Protocol

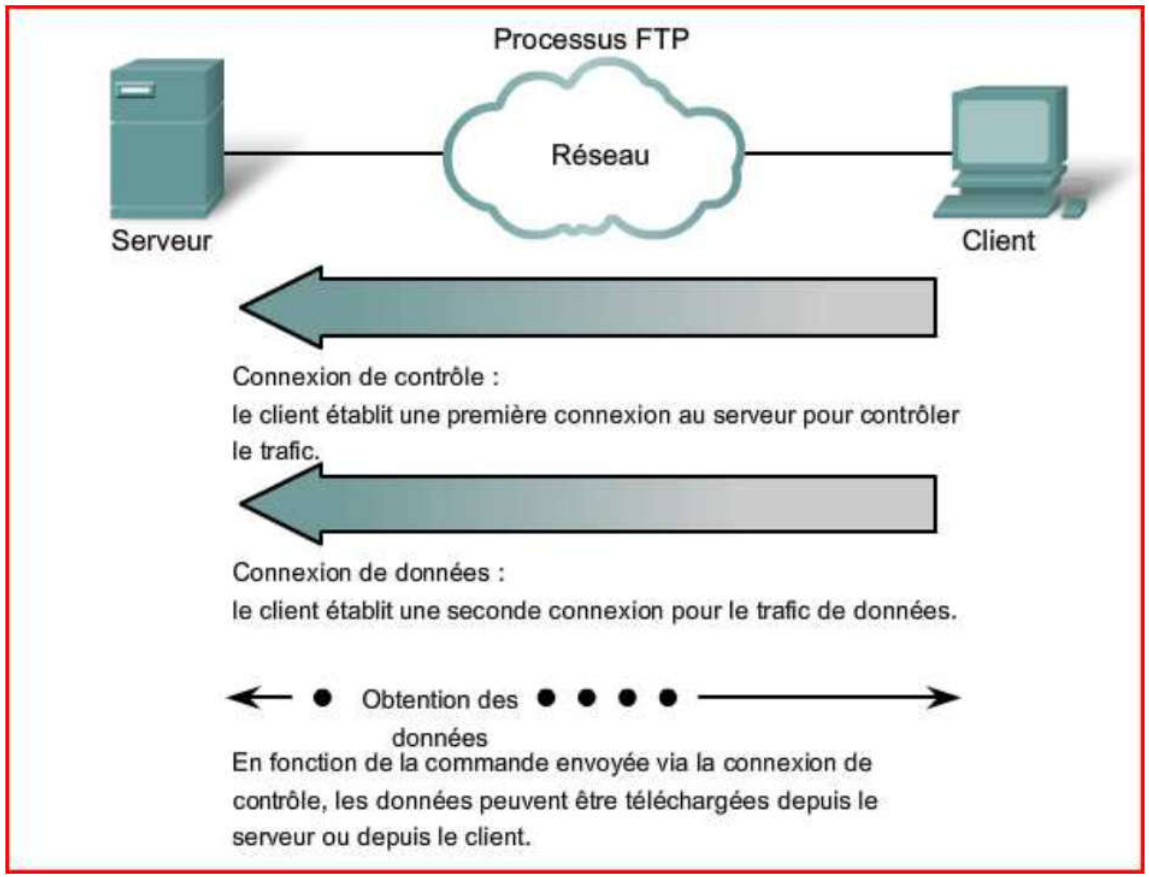
Le client FTP est une application utilisée pour récupérer des fichiers sur un serveur FTP (démon FTPd)

Nécessite deux connexions entre client et serveur :

- Une pour l'échange des commandes et des réponses (port TCP 21)
- L'autre pour le transfert de fichiers (port TCP 20)

Le transfert de fichiers peut s'effectuer dans les deux directions.

# Protocole FTP



# Services et protocoles SMTP/POP

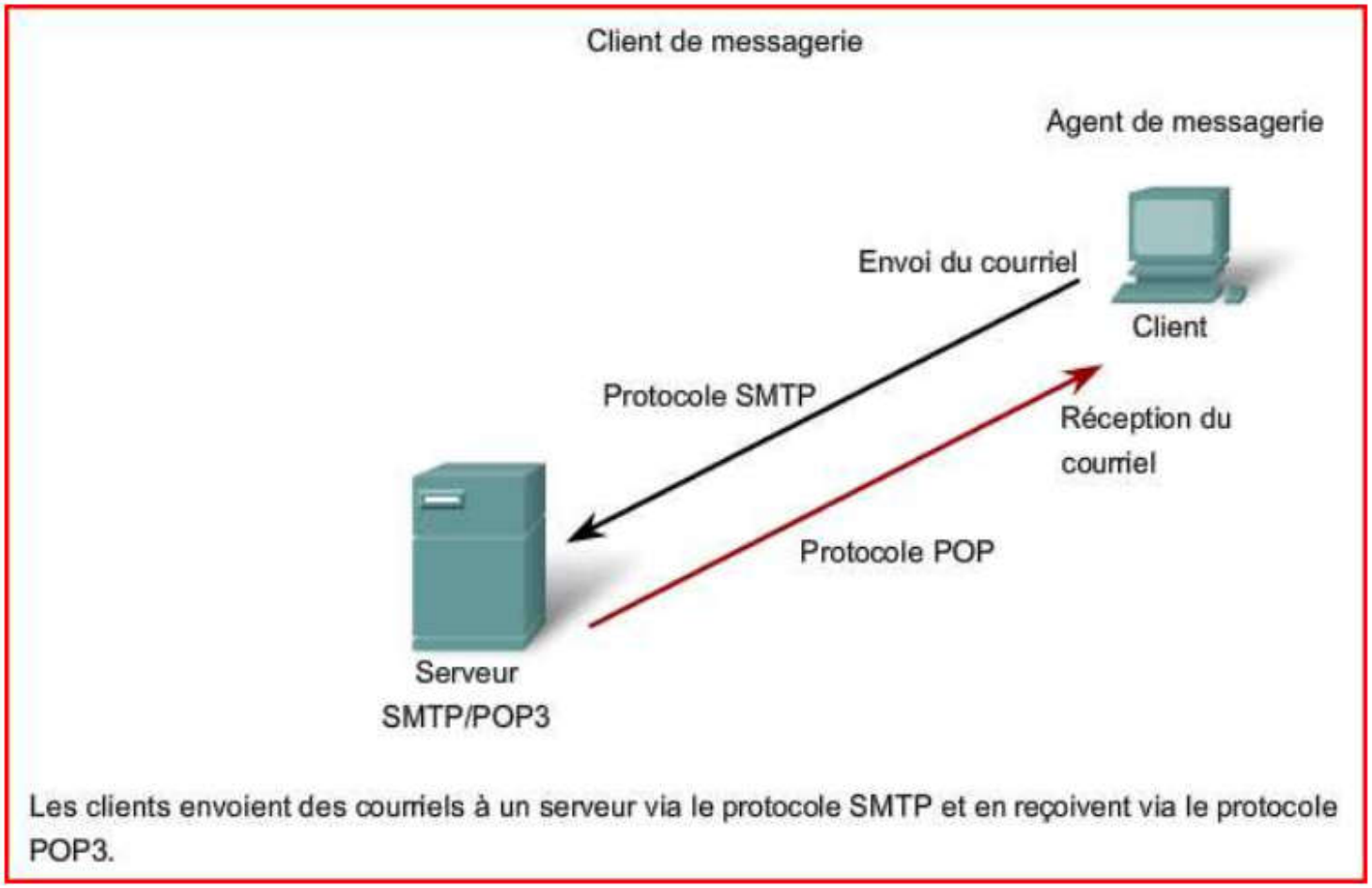
Dédiés à la messagerie électronique

**SMTP** (Simple Mail Transfer Protocol) s'occupe de la partie envoi de mails.

**POP** (Post Office Protocol) s'occupe de la partie réception de mails.

Ces deux protocoles définissent des processus client/serveur.

# Services et protocoles SMTP/POP




Les clients envoient des courriels à un serveur via le protocole SMTP et en reçoivent via le protocole POP3.

# Services et protocoles SMTP/POP

## Client de messagerie

Le **MUA** (Mail User Agent ) fait le lien avec assure le serveur pour relever le courrier entrant ou transférer le courrier sortant.

On trouve le MUA sur les **clients de messagerie** lourds (outlook, thunderbird, Eudora…) ou sur les applications de messagerie WEB appelées **WebMAIL**.

Un client de messagerie fournit généralement les fonctionnalités de plusieurs protocoles au sein d'une même application.

# Services et protocoles SMTP/POP

## Processus de serveur de messagerie

Le serveur de messagerie dispose de deux processus distincts :

- Agent de transfert des messages : **MTA** (Mail Transport Agent)
- Agent de remise des messages : **MDA** (Mail Delivery Agent)

# Services et protocoles SMTP/POP

## Processus MTA (Mail Transport Agent)

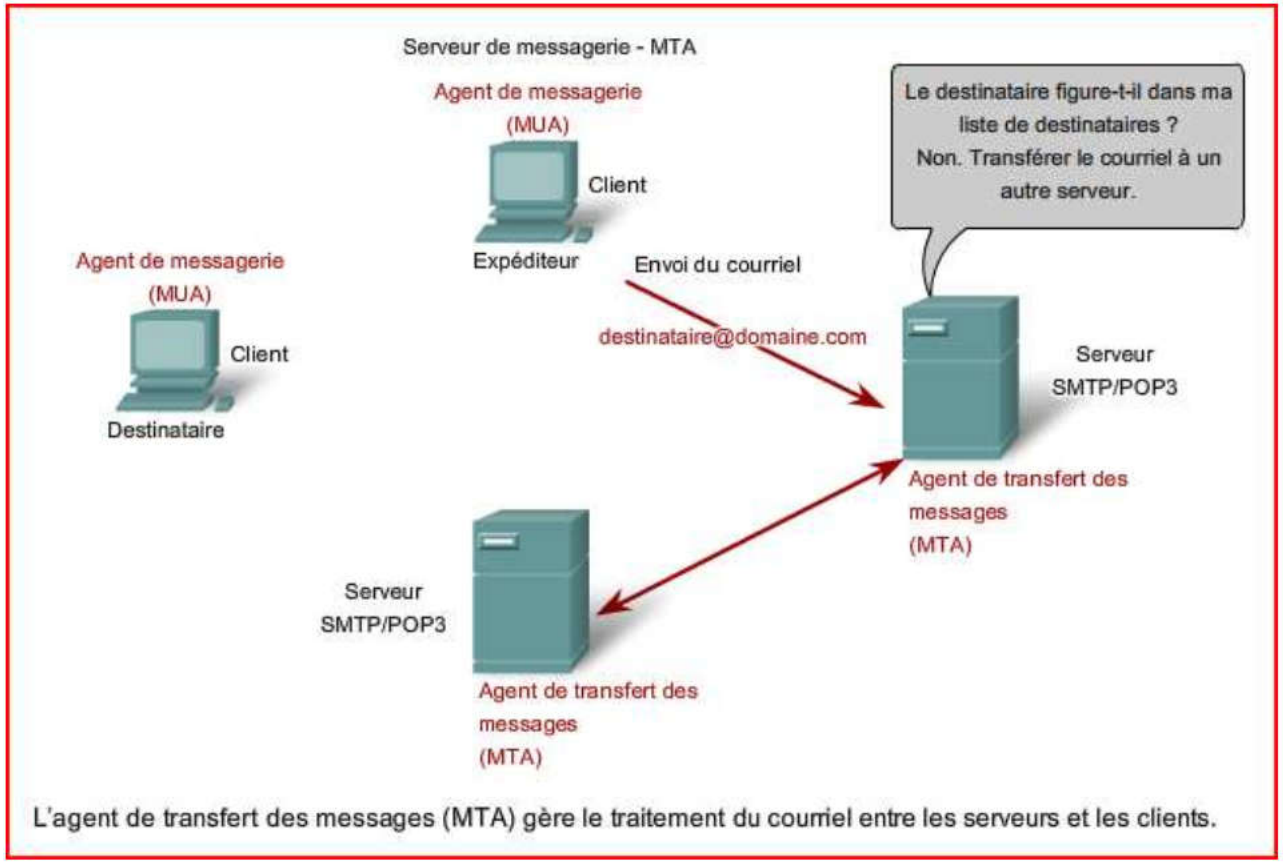
Les MTA discutent entre eux en utilisant le protocole SMTP.

A la réception d'un message, l'agent détermine (depuis l'entête du message) comment le transférer :

- La bal du destinataire réside sur le serveur local, le message est transmis à l'agent MDA
- Sinon le message est acheminé vers le MTA du serveur approprié

# Services et protocoles SMTP/POP

## Processus MTA



# Services et protocoles SMTP/POP

## Processus MDA (Mail Delivery Agent)

Accepte les messages d'un agent de transfert et procède à leur remise effective.

Les message sont placés dans la boîte aux lettres des utilisateurs.

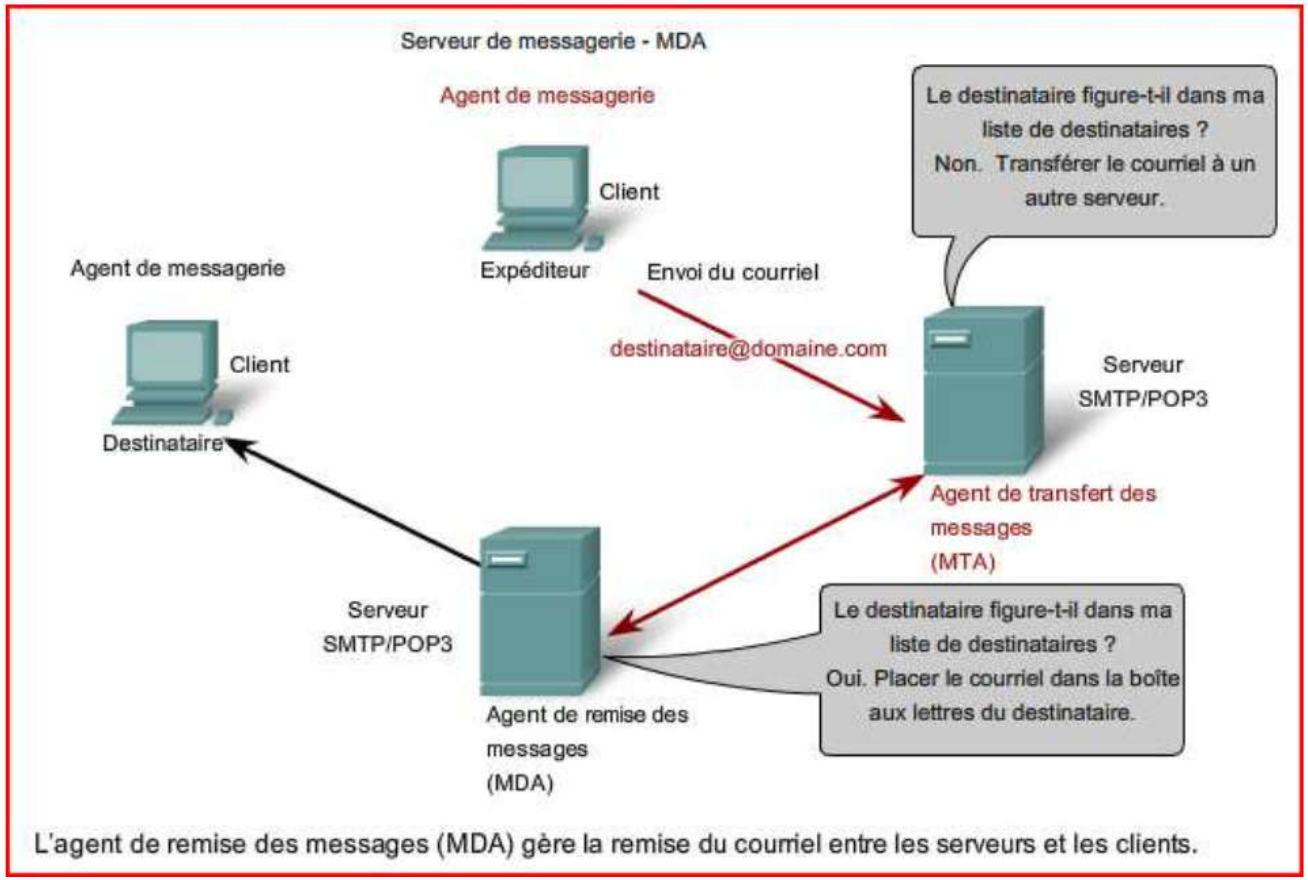
Utilise par exemple POP pour discuter avec le client de messagerie

MDA peut également traiter les derniers aspects liés à la remise tels que

- Analyse antivirus
- Filtrage des courriers indésirables
- Gestion des reçus

# Services et protocoles SMTP/POP

## Processus MDA




L'agent de remise des messages (MDA) gère la remise du courriel entre les serveurs et les clients.

# Services et protocoles SMTP/POP

## Protocole SMTP

Basé sur un ensemble rigide de commandes et de réponses.

Ces commandes prennent en charge les différentes phases telle que l'ouverture de session, le transfert de mail, la vérification des noms, etc…

Exemples de commandes

HELO, EHLO, MAIL FROM, RCPT TO…

# MDA : POP3 vs IMAP

| Feature | POP3 | IMAP |
|---|---|---|
| Where is protocol defined | RFC 1939 | RFC 2060 |
| TCP port used | 110 | 143 |
| Where is e-mail stored | User's PC | Server |
| Where is e-mail read | Off-line | On-line |
| Connect time required | Little | Much |
| Use of server resources | Minimal | Extensive |
| Multiple mailboxes | No | Yes |
| Who backs up mailboxes | User | ISP |
| Good for mobile users | No | Yes |
| User control over downloading | Little | Great |
| Partial message downloads | No | Yes |
| Are disk quotas a problem | No | Could be in time |
| Simple to implement | Yes | No |
| Widespread support | Yes | Growing |

# Services WWW et HTTP

**WWW** : World Wide Web (toile mondiale).

Service internet permettant de consulter des pages depuis des sites au moyen d'un navigateur.

Inventé au CERN à Genève (Tim Berners-Lee et Robert Cailliau) au début des années 90.

Basé sur un système hypertexte et hyperlien.

Autres noms : Le Web, la toile,…

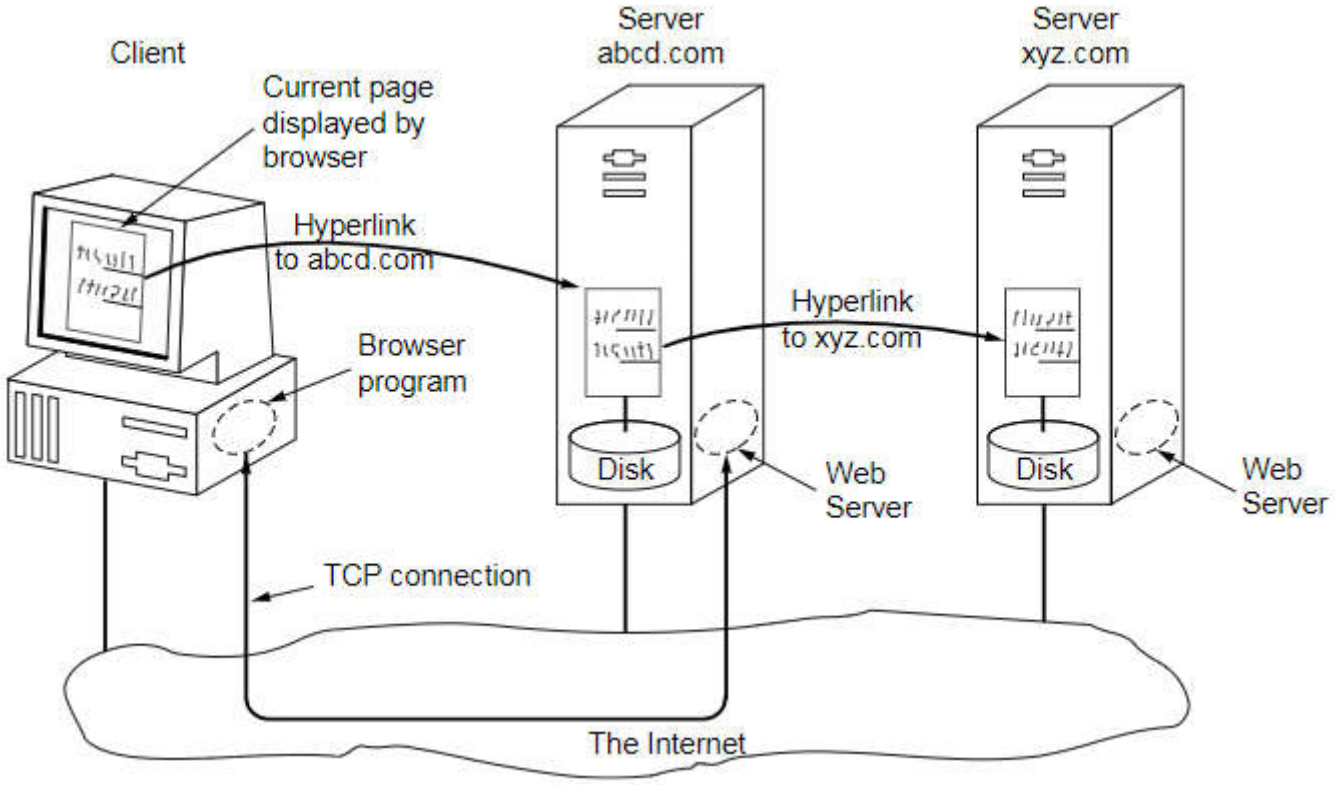
**Attention** : on confond souvent Web et Internet ! Et on appelle souvent internet la toile !

# Services WWW et HTTP


Plaque commémorative dans les locaux du CERN

Premier logo du WWW créé par Robert Cailliau

# WWW : la toile


Client
Server
abcd.com
Server
xyz.com
Current page
displayed by
browser
Hyperlink
to abcd.com
Hyperlink
to xyz.com
Browser
program
Disk
Web
Server
Disk
Web
Server
TCP connection
The Internet

# Services WWW et HTTP

Le **serveur Web** s'exécute en tant que service et propose différents types de fichiers (ressources).

Les **navigateurs Web** sont les applications clientes utilisées pour se connecter et accéder aux ressources stockées sur un serveur Web. A la réception de ces ressources, le navigateur interprète les données et les présente à l'utilisateur.

Le service web s'appuie sur le protocole HTTP

# Protocole HTTP

Le protocole HTTP (Hypertext Transfer Protocol) fait partie des protocoles de la suite TCP/IP. Il constitue l'un des protocoles d'applications les plus utilisés.

- Protocole basé sur le modèle requête/réponse : requête simple (sans état).
- Protocole très flexible
- Protocole non sécurisé (les données voyagent en clair)

Port standard utilisé : port 80

# URIs et URLs

Le développement de WWW a été possible grâce à la mise en place et à l'utilisation d'un système permettant d'identifier et de localiser les ressources.

**URI** : Uniform Ressource Identifier.

Chaîne de caractères respectant une norme syntaxique et permettant d'identifier une ressource.

mailto:dhalluin@cril.fr

ftp://eleloup@mimosa.univ-mrs.fr/

**URL** : Uniform Ressource Locator.

Utilisé par WWW pour localiser un élément sur le réseau.

**protocole://serveur.domaine[:port]/chemin**

**Remarque :** une URL est une URI mais l'inverse n'est pas toujours vrai

# Protocole HTTP

HTTP a connu de nombreuses évolutions.

Versions :

- HTTP 1.0 : très simple
- HTTP 1.1 : amélioré (introduction des connexions persistantes)
- HTTP-NG : refonte totale
- SPDY, WebSocket….

# Protocole HTTP 1.0

Protocole très simple en ligne de caractères.

Types de requêtes principales

- **GET** : récupère les informations et données de l'URI
- **HEAD** : récupère les informations d'entête
- **POST** : envoi de données de formulaire
- **PUT** : enregistrement (upload) de données
- **DELETE** : suppression / effacement
- **OPTIONS** : demande des options de communication

# Protocole HTTP 1.0

En réponse, le serveur retourne un ensemble de données (page) contenant (1ere ligne) un statut de retour :

100-199 : statut d'information

200-299 : succès

- 200 : OK
- 201 : créé

300-399 : redirections

- 301 : redirection
- 304 : non modifié

400-499 : requêtes incorrectes

- 400 : mauvaise requête
- 403 : interdit
- 404 : non trouvé

500-599 : erreurs côté serveur

- 500 : erreur interne du serveur
- 503 : service indisponible

# Protocole HTTP 1.0

## La requête GET

Requête cliente permettant d'obtenir des données. Un navigateur Web envoie le message GET pour demander des pages au serveur Web.

En réponse, le serveur retourne une ligne d'état (exemple HTTP/1.0 200 OK) ainsi qu'un message dont le corps peut contenir le fichier demandé, un message d'erreur ou d'autres informations.

# Protocole HTTP 1.0

## La requête POST

Requête utilisée pour envoyer vers le serveur des messages contenant de la donnée.

**Exemple** : envoie les données entrées dans le formulaire d'une page Web

# Protocole HTTP 1.0

## La requête PUT

Requête utilisée pour télécharger des ressources ou du contenu vers le serveur Web.

**Exemple** : envoie les données entrées dans le formulaire d'une page Web

# Protocole HTTP 1.0

## Gestion de sessions

Le protocole ne proposant pas d'états (mémoire), comment associer une session à plusieurs requêtes consécutives ?

Mise en place de cookies

Côté serveur : set-cookie:nom=valeurUnique

Réutilisé dans les entêtes des requêtes suivantes.

Pratique pour les utilisateurs (achats en ligne, les forums,…)

Et surtout pour le marketing !

# Protocole HTTP 1.1

Basé sur la RFC 2616 (1999)

Permet de connaître les propriétés exactes du correspondant.

Permet d'héberger plusieurs sites sur le même serveur

Permet de définir la notion de connexion persistante

# Protocole HTTP/2

Basé sur la RFC 7540 (2015)

S'appuie sur SPDY (Google)

Permet de réduire la latence en réutilisant la même socket TCP

Permet de multiplexer le transfert de plusieurs fichiers (composant une même page web) -> utilisation d'une seule connexion

Utilisation de TLS 1.2 pour le chiffrement.

# Protocole HTTPS

**HTTP Secure :** version « sécurisée » du protocole HTTP

HTTPS peut procéder à l'**authentification** et au **chiffrement** pour sécuriser des données lors de leur transfert entre client et serveur.

HTTPS spécifie des règles supplémentaires de transmission entre la couche application et la couche transport.

Basé sur SSL.

# L'homme du jour

(Sir) **Tim Berners-Lee**, né le 8 juin 1955 à Londres.

Informaticien et physicien au CERN, il est le principal inventeur du WWW (World Wide Web). Avec le belge Robert Cailliau, ils développent les trois technologies clés du web : HTML, HTTP et les adresses URL.

En 2004, il est fait Chevalier de l'Empire Britannique.

En avril 2017, il reçoit le prix Turing.

Aujourd'hui considéré comme l'un des plus importants scientifiques du XX$^{ème}$ siècle, il préside le World Wide Web Consortium (W3C) qu'il a fondé.

# Service de partage de fichiers et protocole SMB

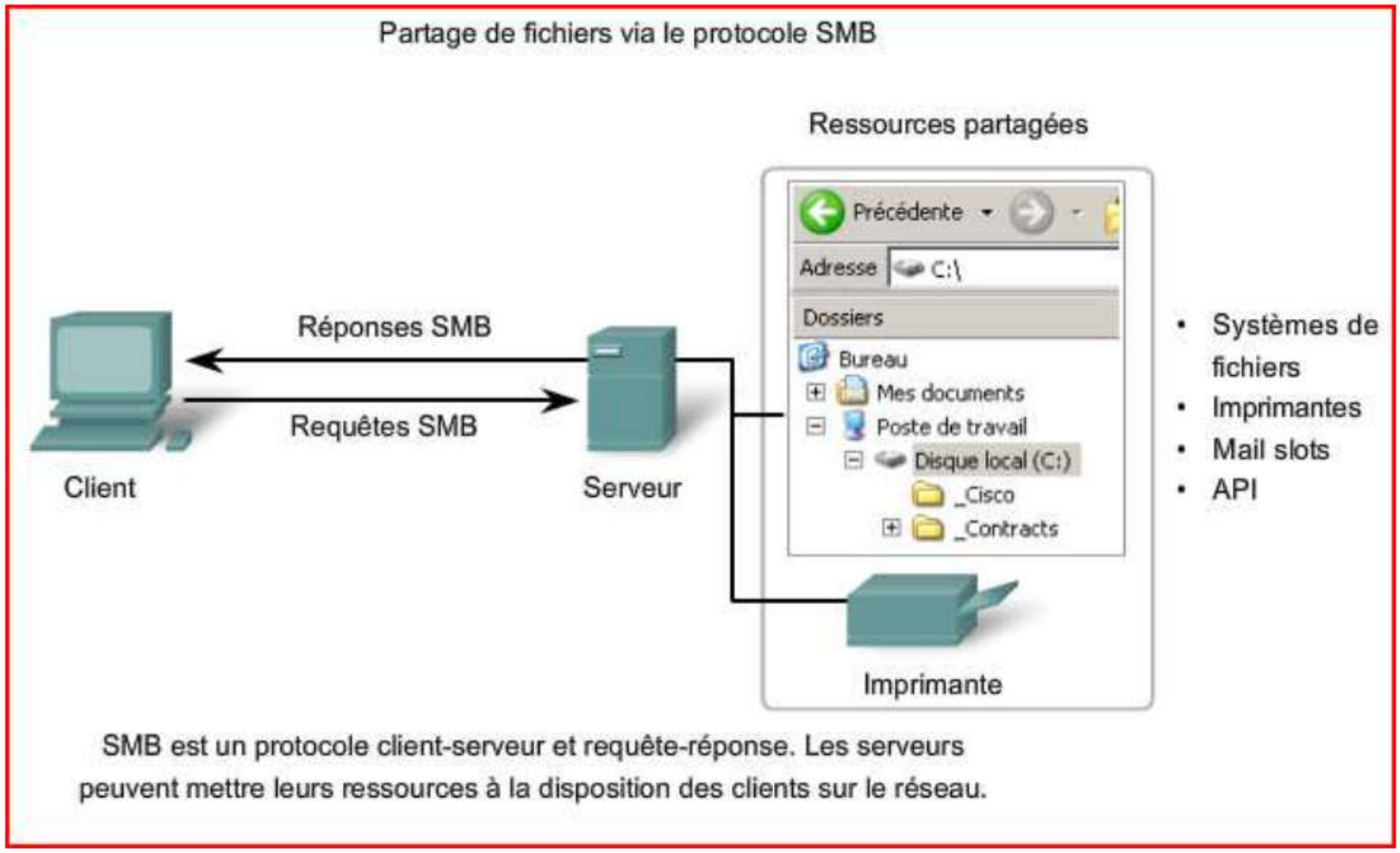
**SMB** : Server Message Block

Protocole de partage de fichiers client/serveur développé par IBM à la fin des années 80.

Contrairement à FTP, les clients établissent une connexion à long terme vers les serveurs.

Une fois connecté, le client peut accéder aux ressources résidant sur le serveur comme si elles étaient situées en local.

# Service de partage de fichiers et protocole SMB



# Service de partage de fichiers et protocole SMB

Le partage de fichiers et services d'impression SMB est devenu la base des réseaux Microsoft.

Linux et Unix fournissent une méthode de partage avec les réseaux Microsoft basée sur SMB et nommée SAMBA.

Les systèmes Apple prennent également en charge le partage de ressources via SMB.

# Service de partage de fichiers et protocole SMB

## Protocole SMB

Décrit l'accès au système de fichiers et la manière dont les clients peuvent demander des fichiers.

Décrit le protocole de communication interprocessus de manière à :

- Démarrer et authentifier des sessions,
- Contrôler les accès aux ressources (fichiers, imprimantes),
- Permettre à une application d'envoyer ou recevoir des messages vers ou depuis un autre périphérique,
- Mettre fin à une session

# Protocole Gnutella

Protocole sur lequel sont basées les applications Peer to Peer.

Permet de mettre des fichiers à disposition des autres utilisateurs à travers internet.

Les logiciels clients permettent de se connecter aux services via internet et de localiser des ressources partagées par d'autres homologues Gnutella.

Nombreuses applications clientes : Bearshare, Gnucleus, Limewire, WinMX, Xolox….

# Protocole Gnutella

Pas d'utilisation de BdD centrale pour enregistrer les fichiers disponibles.

Chaque périphérique (nœud Gnutella) indique aux autres quels fichiers sont disponibles chez eux.

Les clients recherchent les autres nœuds Gnutella auxquels se connecter.

Les nœuds traitent les demandes d'obtention d'emplacement de ressources, les réponses, les messages de contrôle permettant de découvrir d'autres nœuds.

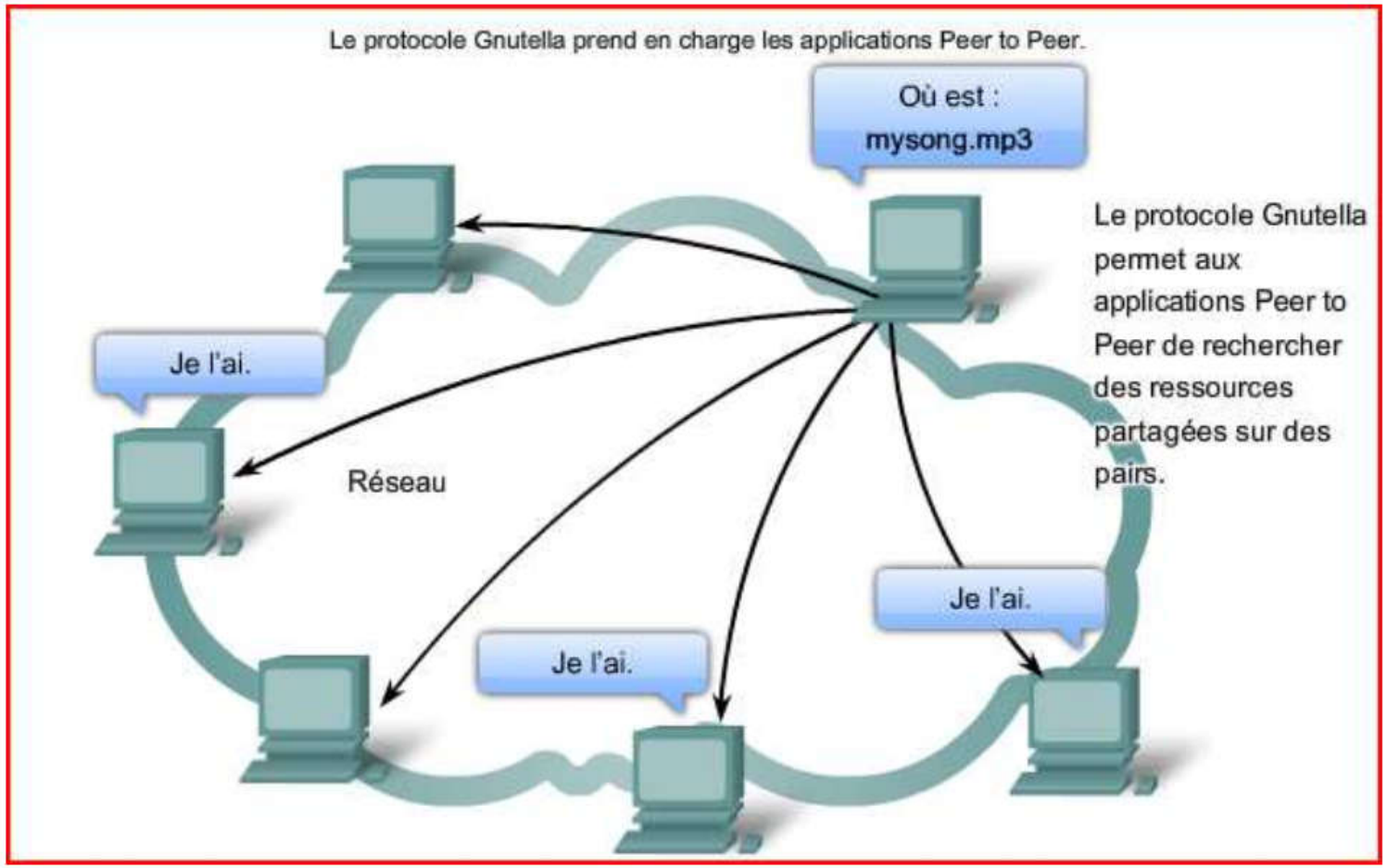
# Protocole Gnutella

Le protocole définit 5 types de paquets différents :

- Ping : découverte des périphériques
- Pong : réponse au ping
- Query : emplacement de fichiers
- Query hit : réponse à une demande
- Push : requête de téléchargement

Le transfert de fichiers se fait en HTTP généralement.

# Protocole Gnutella



# Sécuriser les échanges

La plupart des protocoles présentés (http, telnet, pop3…) n'assure aucun niveau de sécurisation.

Il devient très facile de 'sniffer' des échanges sur le réseau et ainsi de récupérer login, mot de passe ou autres.

Il existe des versions 'sécurisées' de ces protocoles (https, pops) basés sur SSL ou SSH.

# SSL

SSL : Secure Socket Layer

Système permettant l'échange de données entre deux périphériques de manière sûre en assurant :

- **Confidentialité** : les données échangées ne peuvent pas être lues par d'autres périphériques
- **Intégrité** : les données échangées ne peuvent pas être modifiées
- **Authentification** : on s'assure de l'identité des parties (utilisateur, entreprise ou programme)

# SSL

SSL est un complément de TCP/IP permettant de sécuriser n'importe quel protocole s'appuyant sur TCP/IP.

Développé par Netscape et RSA Security.

Développements repris par l'IETF sous le nom TLS (Transport Layer Security)

-> SSL/TLS

SSL est standardisé. Il est considéré comme sûr car analysé par de nombreux spécialistes. Cela explique qu'il soit très utilisé.

OpenSSL est une version open source.

# SSL : fonctionnement

SSL se base sur deux protocoles :

**SSL Handshake protocol** : protocole démarrant l'échange et négociant les clés d'échanges ainsi que les protocoles de chiffrement et de signature à utiliser.

**SSL Record protocol** : protocole contrôlant les échanges et assurant le chiffrement des données échangées.

# SSL : négociation (handshake)

Le but est de toujours utiliser les méthodes de chiffrement et de signature les plus puissantes.

Chacune des parties envoie à l'autre les informations suivantes :

- Version SSL
- Liste des méthodes de chiffrement, de signature, de compression connues
- Des nombres aléatoires
- Des certificats (authentification des parties) -> **PKI** (Public Key Infrastructure)

# SSL : négociation (handshake)

La négociation consiste à utiliser les algorithmes de chiffrement, signatures et compression les plus puissants et communs. C'est également à ce stade que la longueur des clés est déterminée.

Les algorithmes de chiffrement choisis peuvent être symétriques (DES, 3DES, RC4,…) ou asymétriques (RSA, Diffie-Hellman). Ils permettent d'assurer la **confidentialité**.

Les algorithmes de signatures (HMAC, MD5, SHA,…) permettent d'assurer l'**intégrité**.

# SSL : communication (record)

**Emetteur**

les données sont :

- Découpées en paquets,
- Compressées,
- Signées,
- Chiffrées
- Envoyées

**Récepteur**

les données sont :

- Réceptionnées
- Déchiffrées,
- Vérifiées (signature),
- Décompressée,
- Réassemblées

# Utilisation de SSL

SSL est utilisé dans de nombreux protocoles pour les rendre sécurisés :

- HTTPS = HTTP + SSL (cadenas sur les navigateurs)
- FTPS = extension de FTP avec SSL

# Utilisation de SSL

Pour sécuriser les autres protocoles (ou échanges) il est possible de créer un **tunnel SSL** qui peut être vue comme se trouvant juste avant la couche transport.


Le canal de connexion entre les deux parties est alors sécurisées (les deux parties sont authentifiées, les données sont signées et chiffrées)**.**

**C'est bien la couche transport qui est sécurisée**

**Rien n'est modifié au niveau de l'application.**

# SSH

SSH : Secure shell peut être vu comme la version sécurisée de telnet, ou rlogin.

Mis au point par le finlandais Tatu Ylönen en 1995.

Basé sur des techniques assez proches de SSL avec une identification spécifique du client.

Des extensions de SSH ont permis de sécuriser d'autres protocoles (ex. **SFTP** : SSH FTP).

Ici encore on peut considérer que SSH sécurise la couche transport.

# C’est quoi un VPN ?

VPN : Virtual Private Network

Permet de mettre un place un système équivalent à un réseau privé mais en utilisant un réseau public (internet) : création directe entre deux machines

Sécurise les échanges au sein du réseau virtuel.

Permet également de cacher son adresse IP -> anonymat

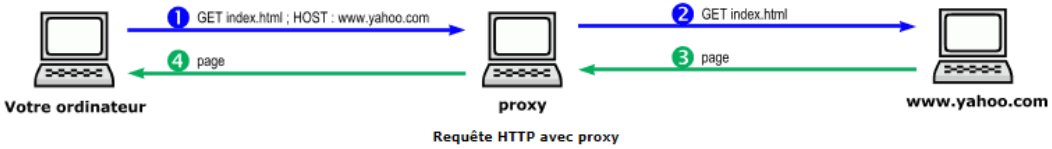
Souvent basé sur SSL ou IPSec

# C’est quoi un VPN ?

# C’est quoi un Proxy ?

Il s’agit d’un **intermédiaire**.

Machine ou application qui se situe entre une machine et le réseau global et qui « filtre » tout ou partie des requêtes.



Requête HTTP avec proxy

# C’est quoi un Proxy ?

Le proxy permet :

- De sécuriser le réseau interne en interdisant, par exemple, d’accéder à tout ou partie d’internet
- De sécuriser les accès depuis l’extérieur (firewall).
- De masquer des données concernant la machine appelante
- De servir de cache pour les pages souvent accédées (proxy-cache)

Mais attention :

possibilité de censure, perte de confidentialité, niveau de sécurité moindre.

# Conclusion

La couche application est la couche la plus proche de l'utilisateur.

Elle propose l'ensemble des logiciels et services permettant à celui-ci d'utiliser le réseau de manière « transparente » en proposant un grand nombre de protocoles utilisés partout et par tous.