

# Introduction aux services réseaux - R2.05

## Architecture des réseaux

### Cours 1

Michel Salomon

IUT Nord Franche-Comté  
Département d'informatique

basé sur un cours de Frédéric Suter

#### Ouvrage de référence

*Computer Networking : A Top-Down Approach*

Jim Kurose and Keith Ross. Addison-Wesley.

# Plan des savoirs réseaux étudiés en R2.05

## Étude des couches hautes de la pile protocolaire

- **Couche Application**

- Principe des applications
- *World Wide Web*
- Transfert de fichiers
- Courrier électronique
- Nom de domaine

- **Couche Transport**

- Rôle de la couche transport
- Description des protocoles de la couche transport

# Couche Application

- Qu'est-ce qu'une application réseau ?
- Description de protocoles de niveau application
  - *World Wide Web* - HTTP
  - Transfert de fichiers - FTP
  - Courrier électronique (courriel) - *e-mails*
    - Envoi ⇒ SMTP
    - Réception ⇒ POP3, IMAP
  - Partage de fichiers Pair-à-Pair
  - ***Domain Name System***

# Grande variété d'applications réseau

- *World Wide Web*
- Courrier électronique
- Messagerie instantanée
- Transfert de fichiers
- Partage de fichiers
- Jeux en réseau
- Vidéo à la demande
- Téléphonie sur Internet
- Vidéoconférence
- Calcul distribué (Seti@Home, Climateprediction.net, Rosetta@home, etc.)
- etc.

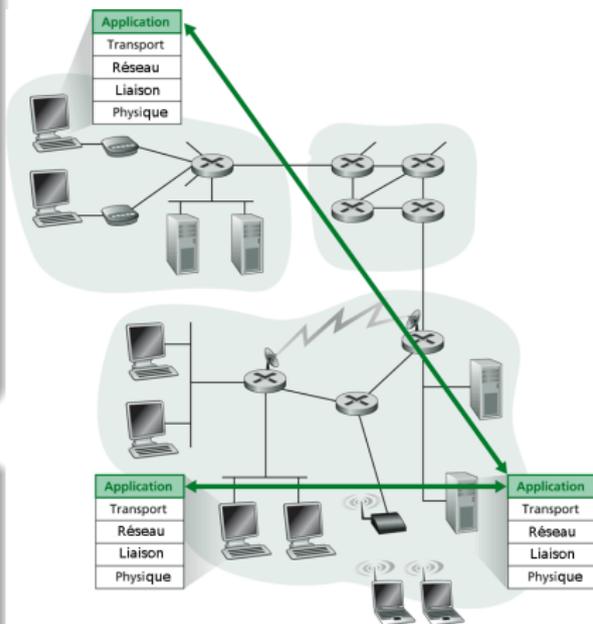
# Qu'est-ce qu'une application réseau

## Extrémités du réseau

- **Programmes** fonctionnant
  - sur **différents types d'hôtes** ;
    - PCs, PDAs, Macs, etc.
  - dans **différents systèmes d'exploitation** ;
    - Linux, MacOS, Windows, etc.
- communiquant via un réseau
- Exemple : client Web ↔ serveur Web

## Cœur du réseau

- **Pas d'applications utilisateur**
- Applications aux extrémités
  - Développement rapide
  - Déploiement rapide



# Architecture des applications - 1. Client / Serveur

## ● Client

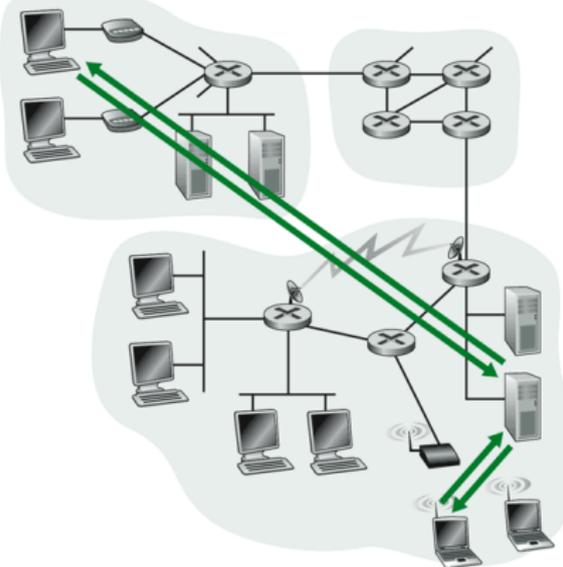
- Hôte parfois volatile
- Adresse IP dynamique (souvent)
- Caractéristiques
  - Actif (initie la communication)
  - Envoie des requêtes au serveur
  - Attend / reçoit les réponses

## ● Pas de com. entre clients

## ● Serveur

- Hôte toujours actif
- Adresse IP statique
- Caractéristiques
  - Passif
  - Attend les requêtes de clients
  - Traite une requête, renvoie une réponse
- Ferme de serveurs

## Métaphore du garçon de café et du consommateur



## Architecture des applications - 2. 3-tiers

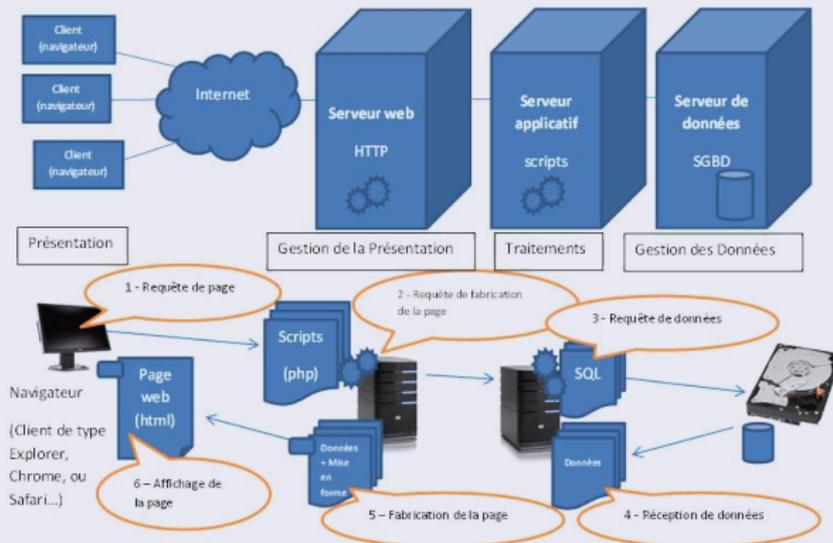
Applications “métiers” construites au-dessus  
des protocoles d'application d'Internet

- Client
  - Actif (initie la communication)
  - Envoie des requêtes à un serveur unique
  - Attend / reçoit les réponses
- Serveur **unique vu par les clients**, dit **serveur frontal**
  - Passif
  - Attend les requêtes de clients
  - **Recourt à d'autres serveurs pour traiter les requêtes**
  - Renvoie la réponse après l'avoir mis en forme
- Architecture assez largement utilisée aujourd'hui en tirant parti des langages de programmation dits web
  - PHP, JSP, Flask Python, etc.

# Architecture des applications - 2. 3-tiers

Les 3-tiers  $\Rightarrow$  Client / Métier / Données

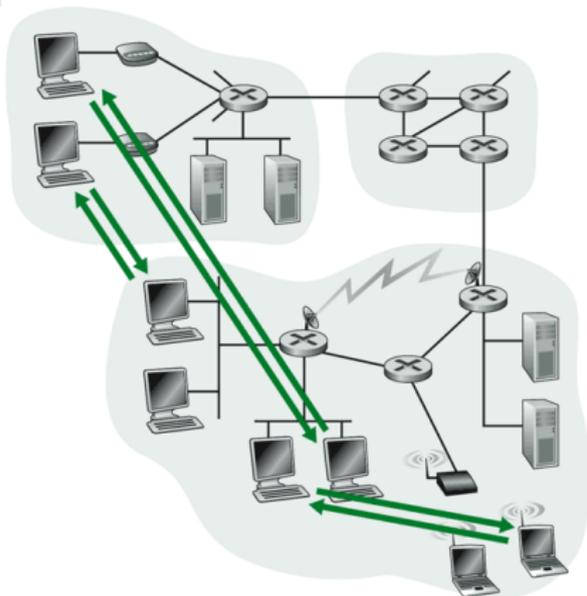
- **Tiers 1**  $\rightarrow$  présentation (affichage)
- **Tiers 2**  $\rightarrow$  traitement (application implémentant la logique métier)
- **Tiers 3**  $\rightarrow$  accès aux données (stockage dans des fichiers, SGBD)



# Architecture des applications - 3. Pair-à-Pair (vrai)

Pair-à-Pair = *Peer-to-Peer* = P2P

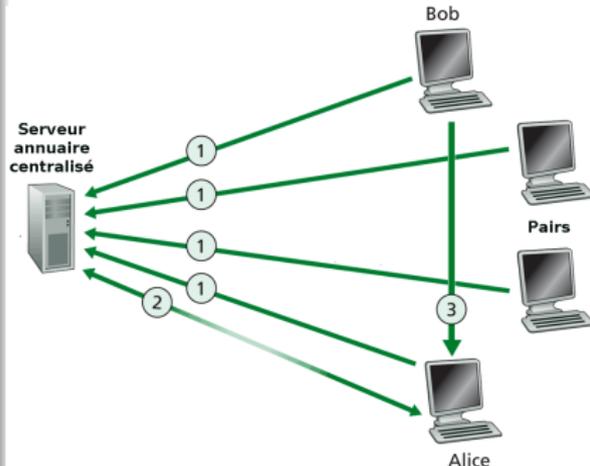
- Pair (*Peer*)
  - Hôte souhaitant partager une ressource (à disposition / l'utiliser)
    - Fichiers
    - Flux multimédias
    - etc.
  - Volatile
    - Changement d'adresse IP
    - Connexion intermittente
- **Pair "à la fois" client et serveur**
- Communication entre pairs
- Intérêt → **extension aisée**
- Inconvénient → **difficile à gérer**
- Exemple → [Gnutella](#)



# Architecture des applications - 4. Hybride

Améliorer la gestion de l'architecture P2P en ajoutant des serveurs

- **Serveur(s)** servant d'**annuaire(s)**
- Fonctionnement
  - ① L'annuaire enregistre les pairs
  - ② À chaque demande d'un pair, l'annuaire trouve un / des pair(s)
  - ③ Communication(s) entre pair demandeur et pair(s) indiqué(s)
- Exemple → [Napster](#)
  - L'annuaire enregistre les fichiers mis à disposition par chaque pair
- Exemple → Messagerie instantanée
  - L'annuaire répertorie les contacts



# Processus communicants - Qu'est-ce qu'un processus ?

Processus = programme s'exécutant sur un hôte

- Architecture client / serveur
  - **Processus client** → initie une communication
    - Exemples → navigateur Web, lecteur de courrier élec., etc.
  - **Processus serveur** → attend d'être contacté
    - Exemples → serveur Web, serveur courrier élec., etc.
- Architecture pair-à-pair
  - **Processus** à la fois **client et serveur**

Communication entre hôtes = communication entre processus

# Processus communicants - Comment les identifier ?

## Un hôte peut exécuter

- des processus clients
- des processus serveurs
- des processus à la fois clients et serveurs

Un processus communicant (client ou serveur) doit être identifié

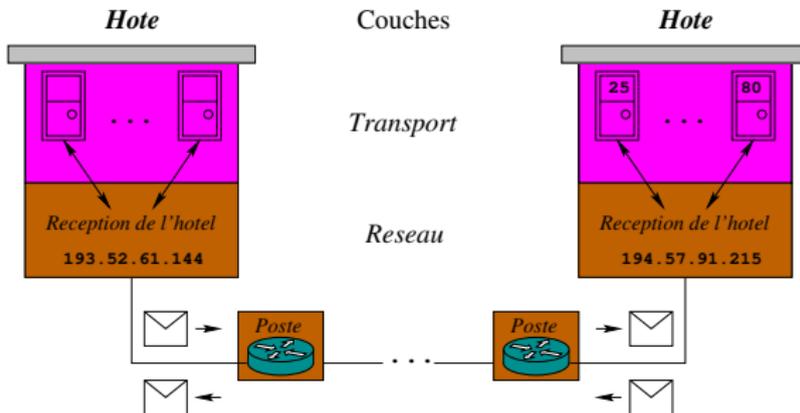
## Comment différencier les processus ?

- Par l'adresse IP identifiant sur Internet chaque hôte ?
- Non ! Plusieurs processus s'exécutent sur un même hôte

- **Identification par une adresse IP et un numéro de port**
- Exemples
  - Serveur Web (protocole HTTP) → 80
  - Serveur d'envoi de courrier électronique → 25
  - etc.

# Processus communicants - Illustration par analogies

- Processus → occupant d'une chambre d'hôtel
  - Adresse IP de l'hôte = adresse de l'hôtel
  - Numéro de port du processus = numéro de chambre
- Communication (commutation de paquets) → courrier postal
  - Expéditeur = adresse IP source et numéro de port
  - Destinataire = adresse IP destination et numéro de port



# Protocoles de la couche application et contraintes

## Protocoles

- Domaine public → Standards
  - **H**yper**T**ext **T**ransfer **P**rotocol → Web
  - **S**imple **M**ail **T**ransfer **P**rotocol → envoi d'e-mail
  - **P**ost **O**ffice **P**rotocol → relève d'e-mail
  - etc.
- Propriétaires → S'imposent par l'usage
  - Gnutella
  - FastTrack (utilisé par Kazaa, etc.)
  - etc.

## Contraintes imposées par une application

- Perte de données → accepter ou non un peu de perte
- Temps de réponse → requérir ou non des délais courts
- Débit → requérir ou non un débit minimal

# Protocoles et contraintes de certaines applications

Application	Protocole niveau appli.	Perte de paquets	Délais courts	Débit minimal
Web	HTTP	non	non	non
Envoi <i>e-mail</i>	SMTP	non	non	non
Relève <i>e-mail</i>	POP	non	non	non
Transfert fichiers	FTP	non	non	non
Audio /vidéo	Propriétaires	oui	oui	oui
Jeux	Propriétaires	oui	oui	oui
...	...	...	...	...

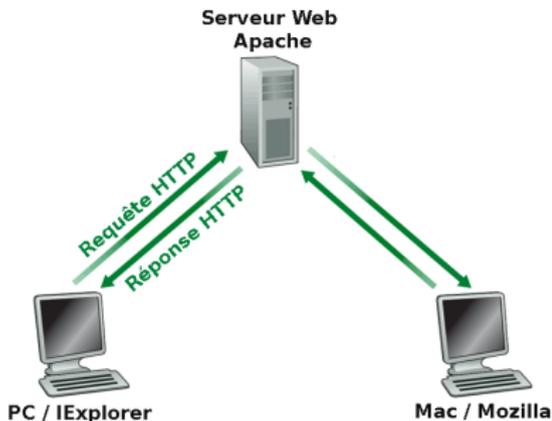
## Protocoles de niveau transport

- *Transmission Control Protocol* → pas de perte de paquets
- *User Datagram Protocol* → perte de paquets

# HyperText Transfer Protocol

Protocole de communication entre navigateur Web et site Web

- Modèle Client / Serveur
  - Navigateur Web
    - Envoie des **requêtes**
    - Affiche les **réponses**
  - Serveur Web (port 80)
    - Reçoit des requêtes
    - Envoie des pages Web, des objets multimédias en réponses
- Transfert de pages Web
  - Hypertexte
- Utilise TCP (poignée de mains)



HTTP n'est pas sécurisé, HTTPS est la variante sécurisée (port 443)

# Qu'est-ce qu'une page Web ?

- Fichier écrit avec un **langage de balisage**
    - *HyperText Markup Language*
    - *eXtensible HyperText Markup Language*
  - Rôles des balises
    - Mettre en forme le contenu de la page
    - Inclure d'autres ressources (ou objets) multimédias
  - Ressources (ou objets) multimédias
    - De différentes natures
      - Audios → fichiers sons ; visuelles → images JPEG, etc.
      - Audiovisuelles → fichiers vidéos
      - Logicielles → applets Java, etc.
      - Textuelles → autres fichiers HTML, etc.
    - Si c'est un fichier → référencé par une adresse
      - Nom de l'hôte + chemin permettant d'atteindre le fichier
- [http://www.univ-fcomte.fr/download/site-principal/image/graphisme/entete\\_1\\_ufc.jpg](http://www.univ-fcomte.fr/download/site-principal/image/graphisme/entete_1_ufc.jpg)

# Qu'est-ce qu'une page Web ?

- Pages web liées par des hyperliens
  - Dynamiques → le serveur génère une page à la volée
  - Statiques → le serveur stocke les différentes pages

## Source de la page Web d'accueil de [www.univ-fcomte.fr](http://www.univ-fcomte.fr)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML+RDFa 1.0//EN"
"http://www.w3.org/MarkUp/DTD/xhtml-rdfa-1.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr" version="XHTML+RDFa 1.0" dir="ltr"
xmlns:og="http://ogp.me/ns#"
xmlns:content="http://purl.org/rss/1.0/modules/content/"
xmlns:dc="http://purl.org/dc/terms/"
xmlns:foaf="http://xmlns.com/foaf/0.1/"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xmlns:sioc="http://rdfs.org/sioc/ns#"
xmlns:sioct="http://rdfs.org/sioc/types#"
xmlns:skos="http://www.w3.org/2004/02/skos/core#"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#">
<head profile="http://www.w3.org/1999/xhtml/vocab">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link rel="shortcut icon" href="https://www.univ-fcomte.fr/sites/all/themes/ufc_central/favicon/favicon.ico" type="image/vnd.microsoft.icon" />
<meta name="viewport" content="initial-scale=1.0, width=device-width" />
<meta name="generator" content="Drupal 7 (https://www.drupal.org)" />
<link rel="canonical" href="https://www.univ-fcomte.fr/" />
<link rel="shortlink" href="https://www.univ-fcomte.fr/" />
<meta property="og:site_name" content="Université de Franche-Comté" />
<meta property="og:type" content="website" />
<meta property="og:url" content="https://www.univ-fcomte.fr/" />
<meta property="og:title" content="Université de Franche-Comté" />
<title>Université de Franche-Comté |</title>
<link rel="apple-touch-icon" sizes="57x57" href="/sites/all/themes/ufc_central/favicon/apple-icon-57x57.png">
<link rel="apple-touch-icon" sizes="60x60" href="/sites/all/themes/ufc_central/favicon/apple-icon-60x60.png">
<link rel="apple-touch-icon" sizes="72x72" href="/sites/all/themes/ufc_central/favicon/apple-icon-72x72.png">
<link rel="apple-touch-icon" sizes="76x76" href="/sites/all/themes/ufc_central/favicon/apple-icon-76x76.png">
<link rel="apple-touch-icon" sizes="114x114" href="/sites/all/themes/ufc_central/favicon/apple-icon-114x114.png">
```

# Quel est le rôle des cookies ?

## Utilisés pour mémoriser des informations / états

- Autorisation
- Panier d'achats
- Profil de consommation
- État d'une session utilisateur (exemple : Webmail)
- etc.

## Cookies et vie privée

- Les cookies permettent à un site d'en apprendre beaucoup
  - Habitudes de consommation
  - Suivre votre exploration d'un site
  - etc.
- Certains cookies peuvent stocker des données "sensibles"
  - Nom et e-mail
  - Identifiant et mot de passe de connexion

# Conserver un état via un cookie

- Composants

- ① En-tête de requête HTTP (client → serveur) Cookie
- ② En-tête de réponse HTTP (client ← serveur) Set-cookie
- ③ Fichier sur la machine de l'utilisateur, géré par son navigateur
- ④ Base de données sur le site Web

- Exemple

- En surfant sur Internet, Alice visite un site de vente en ligne pour la première fois
- À sa première requête HTTP arrivant sur le site, sont créés :
  - un identifiant unique ;
  - une entrée pour cet identifiant dans une base de données
- L'association site-identifiant est stockée localement sur la machine d'Alice via un cookie géré par le navigateur

Un cookie peut être **temporaire** ou **persistant** (en jours, mois, ...)

# Un cookie est-ce un mouchard et peut-on s'en passer ?

- Un cookie n'est **pas vraiment un mouchard**
  - Il permet effectivement de suivre la visite d'un site
    - Via un cookie attaché à une bannière (publicitaire)
  - Mais pas de façon nominative (infos attachées à un hôte)
  - Difficile d'obtenir l'identité (croisement de bases de données)
- Parfois, classement abusif d'un cookie comme mouchard
  - Outils de sécurité le voyant comme un *spyware* (espionnage)
  - Un espionnage est un programme → pas le cas d'un cookie
- Certains services sur le Web nécessitent souvent un cookie
  - Webmail ; Forums ; etc.

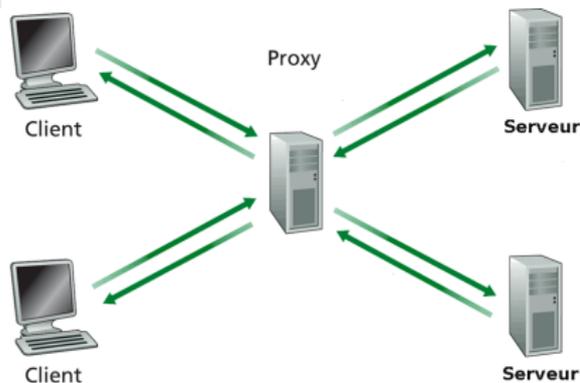
## Politique de Cookies - Bases légales

- Cookies nécessaires au fonctionnement, à la sécurité du site
  - Intérêt légitime
- Pour les autres types de cookies (tiers et publicitaires)
  - Consentement de l'utilisateur nécessaire

# Cache Web (ou serveur Proxy) - Objectif

Satisfaire la requête d'un client sans impliquer le serveur d'origine

- Utiliser un serveur intermédiaire
- Configuration du navigateur
  - Passage par le serveur Proxy faisant office de cache
- Requêtes HTTP du navigateur envoyées au cache
  - Ressource présente dans le cache → envoi de l'objet caché
  - Ressource absente du cache
    - ① requête envoyée par le Proxy au serveur d'origine
    - ② réponse reçue mise en cache par le Proxy
    - ③ envoi de l'objet caché



# Cache Web (ou serveur Proxy) - Mise en œuvre et intérêt

## Mise en œuvre

- Joue à la fois le rôle de client et de serveur
- Habituellement déployé par un FAI ou une institution (Université, Entreprise, etc.)
- Doit contrôler régulièrement la validité des ressources stockées

## Pourquoi mettre en cache ?

- Réduire le temps de réponse à une requête
- Réduire le trafic réseau sur un lien d'accès (vers le routeur)
- Permet aux fournisseurs de contenu lents de livrer efficacement leur contenu

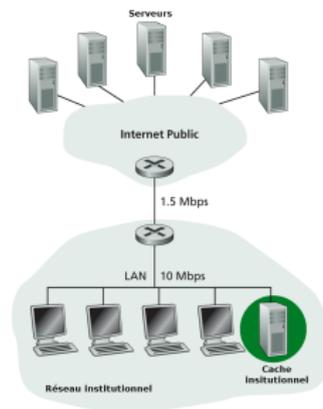
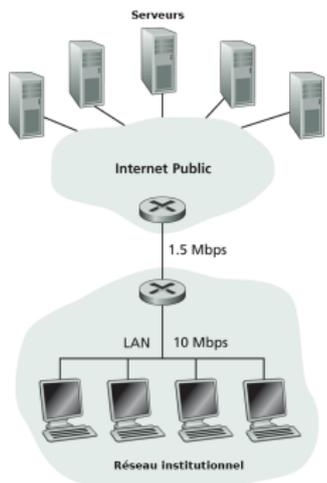
## Sécurité accrue

- Clients masqués par le serveur Proxy sur Internet
- Filtrage des requêtes

# Cache Web (ou serveur Proxy) - Intérêt

Que faire en cas de saturation d'un lien accès ?

Un serveur Proxy donne de meilleurs résultats et à moindre coût, si on compare avec l'augmentation du débit du lien



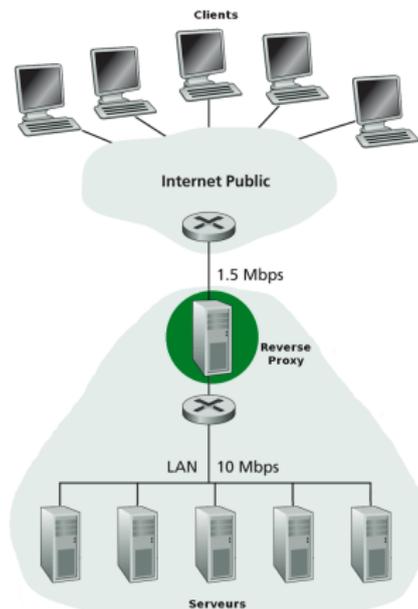
Taux de réussite d'un cache en général entre 20 et 70%

# Reverse Proxy - Utilisation dans l'autre sens

Intermédiaire entre des clients Internet et des serveurs internes

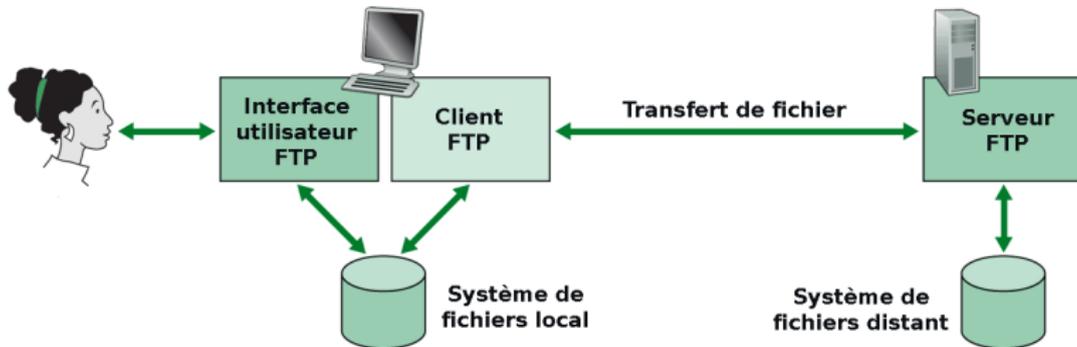
- Mise en cache des requêtes les plus fréquentes (pages / objets statiques)
- Sécurité accrue du site interne
- Répartition de la charge via une redirection vers différents serveurs

Un *Reverse Proxy* soulage un serveur et sécurise l'accès

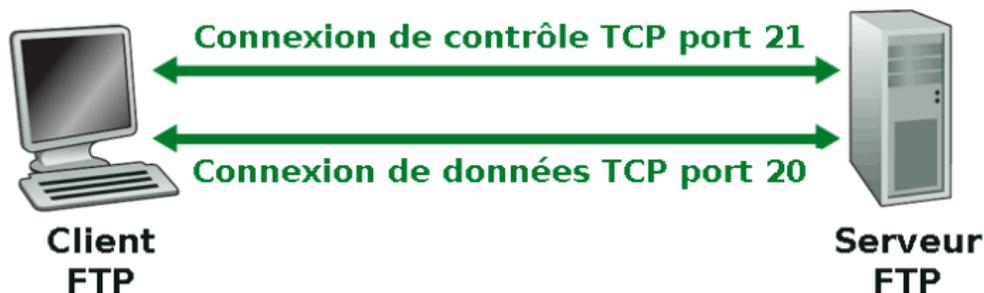


# File Transfer Protocol

- Transférer un / des fichier(s) depuis / vers un hôte distant
- Modèle Client / Serveur
  - Client FTP → initie le transfert
  - Serveur FTP → en attente de connexions sur le port 21



# File Transfer Protocol



- Utilisation de la connexion de contrôle
  - Pour l'authentification du client
    - l'autoriser ou non à se connecter
  - Pour envoyer les demandes de transfert
    - envoyer ou recevoir un ou des fichier(s)
- Utilisation de la connexion de données
  - Pour la transmission des données

# Courrier électronique

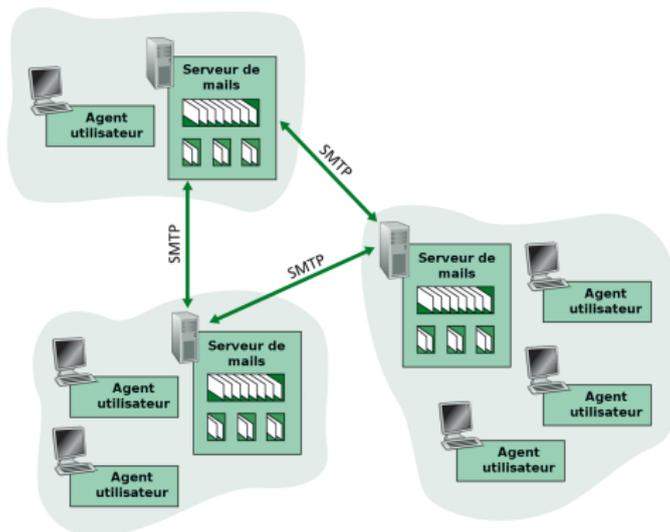
- La messagerie électronique est devenue un outil indispensable
- Normalisée il y a plusieurs dizaines d'années
- Pouvant être l'objet d'utilisations malveillantes ou abusives
  - Diffusion de SPAM, c.-à-d. du courrier élec. non sollicité
  - Diffusion de vers
  - etc.

## Composants majeurs

- Agents utilisateurs
- Serveurs de messagerie (ou *mail*)
- Différents protocoles
  - Envoi de courriel → ***Simple Mail Transfer Protocol*** (port 25)
  - Relève d'un courriel dans une **Boîte Aux Lettres**
    - ***Post Office Protocol*** (port 110)
    - ***Internet Message Access Protocol*** (port 143)

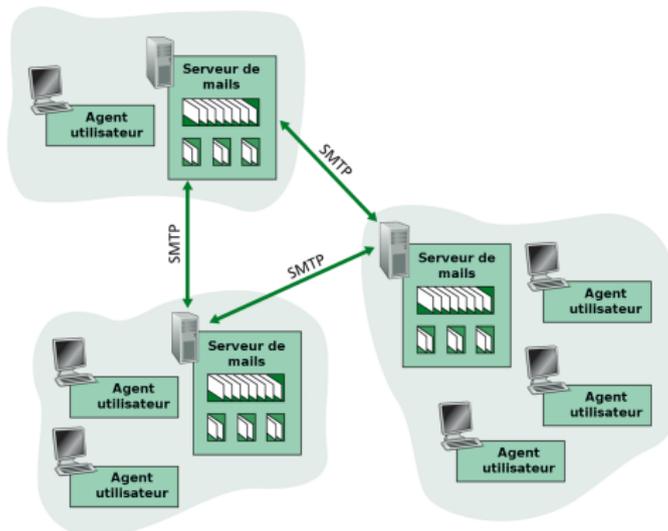
# Composants du courrier électronique - Agent utilisateur

- Client de messagerie
  - Eudora, Outlook, Thunderbird, Pine, etc.
- Composition, édition et lecture d'e-mails
- Messages sortants et entrants stockés sur le serveur



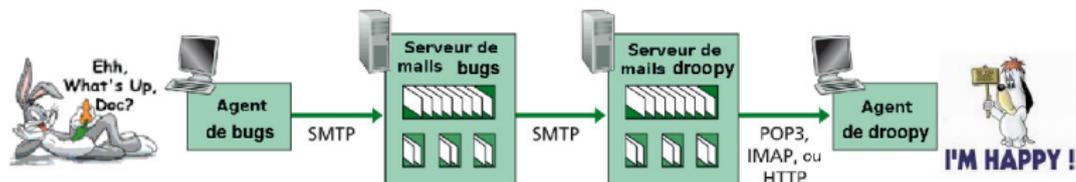
# Composants du courrier électronique - Serveurs

- Une **file de messages**
  - Messages élec. sortants → à envoyer
- Des **Boîtes Aux Lettres**
  - Une BAL contient les messages entrants → à relever
- Transfert des messages entre serveurs via le protocole SMTP

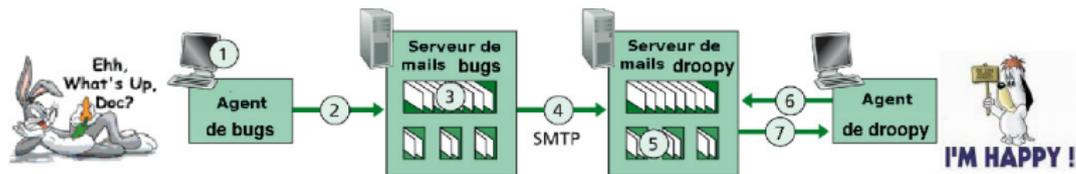


# Comment est transmis un message ?

- Considérons deux utilisateurs :
  - *bugs bunny* ayant `bugsbunny@acme.com` pour adresse élec.
  - *droopy* ayant `droopy@texavery.com` pour adresse élec.
- Serveurs à disposition des deux utilisateurs
  - Habituellement deux serveurs distincts
    - envoi de messages → `smtp.acme.com` et `smtp.texavery.com`
    - relève des messages
      - `pop.acme.com` ou `imap.acme.com` ;
      - `pop.texavery.com` ou `imap.texavery.com`
- *Bugs bunny* envoie un message à *droopy*

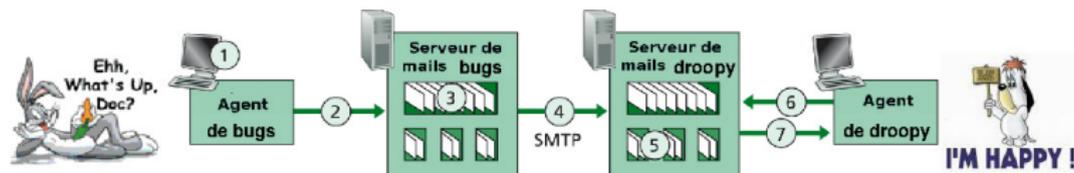


# Comment est transmis un message ?



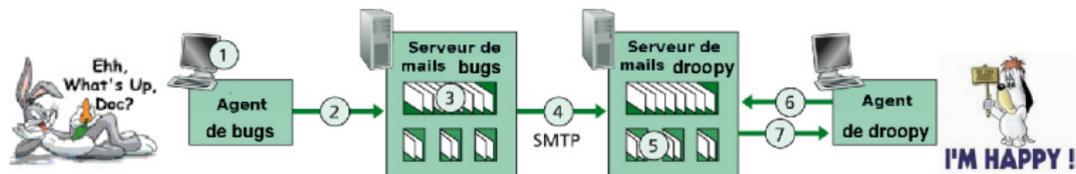
1 *Bugs rédige un e-mail pour droopy@texavery.com*

# Comment est transmis un message ?



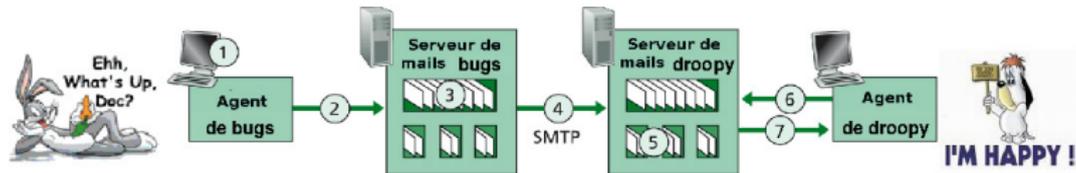
- 1 *Bugs* rédige un e-mail pour `droopy@texavery.com`
- 2 L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente

# Comment est transmis un message ?



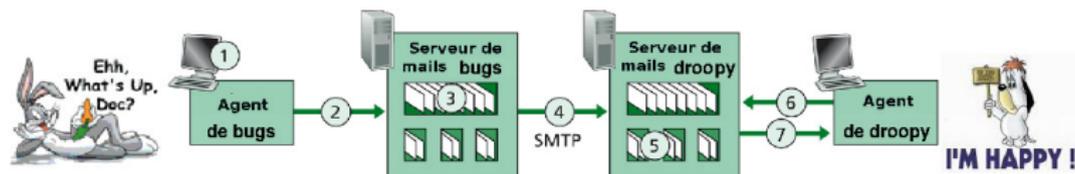
- 1 *Bugs* rédige un e-mail pour `droopy@texavery.com`
- 2 L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente
- 3 Le serveur SMTP de *Bugs* ouvre une connexion TCP avec le serveur SMTP de *Droopy*

# Comment est transmis un message ?



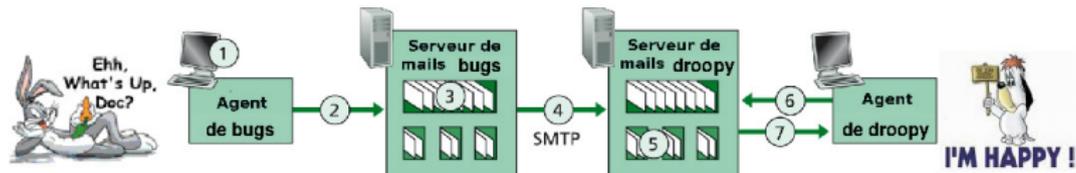
- 1 *Bugs* rédige un e-mail pour `droopy@texavery.com`
- 2 L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente
- 3 Le serveur SMTP de *Bugs* ouvre une connexion TCP avec le serveur SMTP de *Droopy*
- 4 Le serveur SMTP de *bugs* envoie le message sur la connexion

# Comment est transmis un message ?



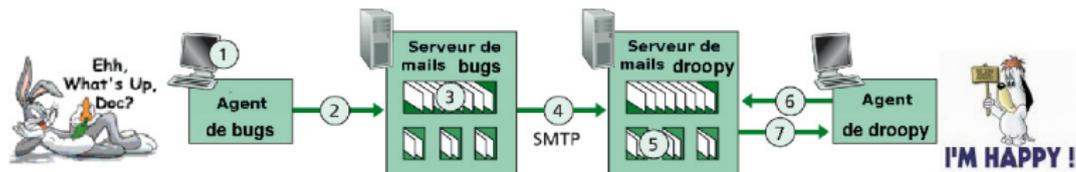
- 1 *Bugs* rédige un e-mail pour `droopy@texavery.com`
- 2 L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente
- 3 Le serveur SMTP de *Bugs* ouvre une connexion TCP avec le serveur SMTP de *Droopy*
- 4 Le serveur SMTP de *bugs* envoie le message sur la connexion
- 5 Le serveur SMTP de *Droopy* place le message dans sa BAL

# Comment est transmis un message ?



- ① *Bugs* rédige un e-mail pour `droopy@texavery.com`
- ② L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente
- ③ Le serveur SMTP de *Bugs* ouvre une connexion TCP avec le serveur SMTP de *Droopy*
- ④ Le serveur SMTP de *bugs* envoie le message sur la connexion
- ⑤ Le serveur SMTP de *Droopy* place le message dans sa BAL
- ⑥ *Droopy* demande à son agent de relever sa BAL

# Comment est transmis un message ?



- ❶ *Bugs* rédige un e-mail pour `droopy@texavery.com`
- ❷ L'agent utilisateur de *bugs* envoie le message à son serveur `smtp.acme.com` → mis dans la file d'attente
- ❸ Le serveur SMTP de *Bugs* ouvre une connexion TCP avec le serveur SMTP de *Droopy*
- ❹ Le serveur SMTP de *bugs* envoie le message sur la connexion
- ❺ Le serveur SMTP de *Droopy* place le message dans sa BAL
- ❻ *Droopy* demande à son agent de relever sa BAL
- ❼ Le serveur POP de *Droopy* envoie le(s) nouveau(x) message(s)

# Simple Mail Transfer Protocol (port 25)

- Protocole de transfert d'emails dans Internet
  - d'un client à un serveur ;
  - d'un serveur à un autre serveur (modèle client / serveur)
- Transfert direct de serveur à serveur
- Assure l'acheminement jusqu'à une BAL
  - 1 Analyse de la partie droite de l'adresse élec. du destinataire
  - 2 Si
    - le domaine le concerne  
→ recherche la BAL associée à la partie gauche de l'adresse ;
    - le domaine ne le concerne pas  
→ demande l'adresse IP du serveur SMTP le gérant
  - 3 Le message est
    - soit rangé dans la BAL ;
    - soit envoyé au serveur SMTP identifié par l'adresse IP

# Simple Mail Transfer Protocol (port 25)

- Interaction Client / Serveur sous forme Commande / Réponse
  - Commande → texte ASCII
  - Réponse → code et phrase de statut

```

Serveur : 220 texavery.com
Client  : HELO acme.com
Serveur : 250 Hello acme.com, pleased to meet you
Client  : MAIL FROM : <bugsbunny@acme.com>
Serveur : 250 bugsbunny@acme.com... Sender ok
Client  : RCPT TO : <droopy@texavery.com>
Serveur : 250 droopy@texavery.com ... Recipient ok
Client  : DATA
Serveur : 354 Enter mail, end with "." on a line by itself
Client  : Subject : Playing together?
Client  : Do you like my cartoons?
Client  : How about playing in the same one?
Client  : .
Serveur : 250 Message accepted for delivery
Client  : QUIT
Serveur : 221 texavery.com closing connection
  
```

# Enveloppe, structure et format d'un message

- L'enveloppe d'un message correspond aux données SMTP
  - MAIL FROM: → adresse de l'expéditeur
  - RCPT TO: → adresse du destinataire
- L'en-tête d'un message contient les informations suivantes :
  - To :
  - From :
  - Subject :
  - le chemin suivi par le message ;
  - etc.
- Le corps du message suit l'en-tête
- Un "." seul sur une ligne termine un message

Quasiment pas de liens entre l'enveloppe  
et le message (en-tête+texte)

## Détails de l'en-tête

- Return-Path : adresse pour répondre ou renvoyer le message
- Received : chaque serveur ayant reçu le message y ajoute une ligne → on peut retracer le chemin suivi par le message
- X- : extensions de différents composants (serveur, etc.)
- Message-ID : identifiant unique ajouté par le premier serveur
- From : adresse de l'expéditeur
- To : adresse du destinataire
- Subject : objet du message
- Date : date d'émission écrite par le client de l'expéditeur
- MIME-Version : version du mode de codage des données
- Content-Type : type de codage utilisé
- charset : jeu de caractères utilisé
- Content-Transfer-Encoding : codage sur 7 ou 8 bits

# Format MIME - Qu'est-ce que c'est ?

## Limitations historiques de SMTP

Caractères du jeu ASCII US pour les messages (en-tête et corps)

- Accentuations non disponibles !
- Transferts de données binaires plus complexes !

## Solution : *Multipurpose Internet Mail Extensions*

- Transfert de contenus quelconques :
  - textes écrits dans des jeux de caractères étendus / différents ;
  - images, vidéos, sons, etc.
- Comment ?
  - Des en-têtes supplémentaires permettent de décrire le contenu
  - Plusieurs formats de contenu sont disponibles

# Format MIME - En-têtes

- **MIME-Version** : numéro de version
- **Content-Type** : décrit le contenu précisément afin que le message puisse être affiché par le destinataire
  - `text/html` ;
  - `image/gif` ;
  - `application/pdf` ;
  - etc.
- **Messages composites**
  - `multipart` définit un séparateur (`boundary`)
    - `multipart/mixed` : plusieurs éléments à la suite ;
    - `multipart/parallel` : des éléments à montrer simultanément ;
    - etc.
  - entre deux séparateurs : `Content-Type`, `Content-...`, etc.
- etc.

# Format MIME - Exemple

```
From: bugsbunny@acme.com
To: droopy@texavery.com
Subject: Text and image
MIME-Version: 1.0
Content-Type: multipart/mixed;
boundary="MonDelimiteur"

-MonDelimiteur
Hello,
This the text of my email.

-MonDelimiteur
Content-Transfer-encoding: base64
Content-Type: image/jpeg
base64 encoded data .....
..... base64 encoded data
-MonDelimiteur
```

# Format MIME - Exemple

```

From: bugsbunny@acme.com
To: droopy@texavery.com
Subject: Text and image
MIME-Version: 1.0
Le délimiteur ne doit pas être contenu dans un des objets\
Content-Type: multipart/mixed;
boundary="MonDelimiteur"
√1er objet (- en début de ligne suivi du délimiteur)
-MonDelimiteur
√Données du 1er objet
Hello,
This the text of my email.
√Fin du 1er objet / début du 2nd
-MonDelimiteur
√Méthode d'encodage du 2nd objet
Content-Transfer-encoding: base64
Content-Type: image/jpeg
√Type du 2nd objet et données\
base64 encoded data .....
.....
..... base64 encoded data
√Fin du 2nd objet
-MonDelimiteur

```

# Format MIME - Exemple

## En-têtes

```

From - Mon Sep 26 12:33:42 2022
X-Account-Key: account6
X-UIDL: 95000.P8K09WfhNatQ6y0Nd5zBXcAKI0Wlu,v7G5Plc421IVY=
X-Mozilla-Status: 0001
X-Mozilla-Status2: 00000000
X-Mozilla-Keys:
Return-Path: <michel.salomon@univ-fcomte.fr>
Received: from ufcpriv204.univ-fcomte.fr (LHLO ufcpriv204.univ-fcomte.fr
(172.20.194.204) by zstore01.univ-fcomte.fr with LMTP; Mon, 26 Sep 2022
12:33:40 +0200 (CEST)
Received: from ufcpriv204.univ-fcomte.fr (localhost [127.0.0.1])
  by postfix.imss91 (Postfix) with ESMTP id 2DE9E4045CE3
  for <michel.salomon@univ-fcomte.fr>; Mon, 26 Sep 2022 12:33:40 +0200 (CEST)
Received: from smtps.univ-fcomte.fr (ufc218.univ-fcomte.fr [194.57.91.218])
  by ufcpriv204.univ-fcomte.fr (Postfix) with ESMTP id 1F73B4045CCD
  for <michel.salomon@univ-fcomte.fr>; Mon, 26 Sep 2022 12:33:40 +0200 (CEST)
Content-Type: multipart/mixed; boundary="-----PrqmsJ03V8BwziUhijaKi6DR"
Message-ID: <14024034-9ea0-3c76-249c-e06eae110ccd@univ-fcomte.fr>
Date: Mon, 26 Sep 2022 12:33:39 +0200
MIME-Version: 1.0
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
Thunderbird/91.13.0
From: Michel Salomon <michel.salomon@univ-fcomte.fr>
Subject: Message pour illustrer MIME
To: michel.salomon@univ-fcomte.fr
Content-Language: fr
X-TM-AS-GCONF: 00

This is a multi-part message in MIME format.
-----PrqmsJ03V8BwziUhijaKi6DR
Content-Type: text/plain; charset=UTF-8; format=flowed
Content-Transfer-Encoding: 8bit

```

# Format MIME - Exemple

## En-têtes

Voici un message contenant un pdf en pièce jointe

-----

\* Michel Salomon - MCF / Assistant professor

-

\* UFC / IUT de Belfort-Montbéliard

\* 19 avenue du Maréchal Juin

\* 90000 Belfort

-----

-----Prqmsj03V8BwziUhjaKi6DR

Content-Type: x-unknown/pdf; name="coursP1.pdf"

Content-Disposition: attachment; filename="coursP1.pdf"

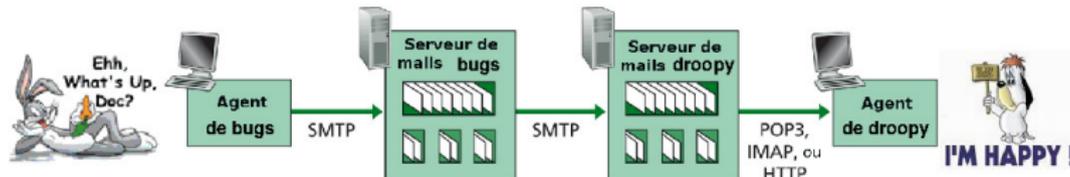
Content-Transfer-Encoding: base64

```
JVBERi0xLjUKJdDUxdgKMzMgMGBvYmoKPDwKL1R5cGUgL1hPYmplY3QKL1N1YnR5cGUgL0Zv
cm0KL0JCb3ggWzAgMCAzNjluODM1IDMuOTg1XQovRm9ybVR5cGUgMQovTWf0cmI4IFsxIDAq
MCAxIDAgMF0KL1Jlc291cmNlcyAzNCAlFikL0xlbmd0aCAxNSAgICAgICAgCi9GaWx0ZXIq
L0ZsYXRIRGVjb2RlCj4+CnN0cmVhbQp42tMPzIAozuACAaf9AFAKZW5kc3RyZWFTcmVuZG9i
ago0MSAwIG9iago8PAovVHlwZSAvWE9iamVjdAovU3VidHlwZSAvRm9ybQovQkpvCBbMCAw
IDU2NjkuMjkwIDhdCi9Gb3JtVHlwZSAxYXN0cmVudC9NYXRYaXggWzEgMCAwIDVgMCAwXQovUmVz
b3Vy
```

# Post Office Protocol (port 110)

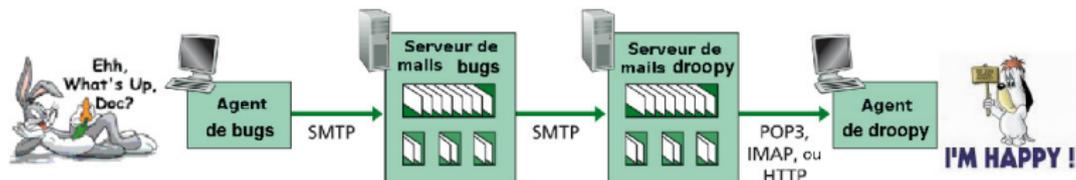
- Protocole d'accès à une BAL → récupération sur le serveur
- Permet de relever périodiquement les messages dans la BAL
  - Relever et supprimer ; Relever et garder
- Récupération complète des messages demandés
- Intéressant si on utilise un seul hôte pour lire son courrier
- Principales commandes

Commande	Description
USER <login>	Login de l'utilisateur
PASS <mot de passe>	Mot de passe associé au login
RETR <num.>	Récupération du message numéro num.
DELE <num.>	Suppression du message numéro num.
LIST <num.>	Affiche le message num.
QUIT	Quitter la connexion en cours



# Internet Message Access Protocol (port 143)

- Protocole d'accès à une BAL → récupération sur le serveur
- Permet une gestion directe à distance de la BAL
- Récupération partielle des messages possible
- Intéressant si on utilise différents hôtes pour lire son courrier
- Avantage par rapport à POP
  - Fonctionnalités avancées (accès concurrents, plusieurs BAL, classement des messages, etc.)
  - Changement aisé de l'agent utilisateur
- Inconvénient
  - Prévu pour une utilisation en ligne



# Vulnérabilité et attaques

## Vulnérabilités

- Absence d'authentification, de confidentialité et d'intégrité
- Contenus exécutables
- Failles de logiciels
- etc.

## Attaques

- Atteinte à la vie privée
- Usurpation d'identité
- Diffusion de SPAM
- Diffusion de programmes malveillants (vers, virus, rançongiciel, etc.)
- Déni de service
- etc.

# Absence d'authentification, confidentialité et d'intégrité

- Aucun de ces services n'est garanti par le protocole SMTP
- Transfert des messages sans aucune vérification
- Contrôles limités sur l'enveloppe
  - Seul le relayage est contrôlé
  - Le message peut contenir n'importe quoi
- Tous les échanges se font en clair

**Ne pas faire une confiance aveugle  
à un message électronique**

# Contenus exécutables et déni de service

- Le format MIME permet de transférer des fichiers binaires
  - programmes exécutables ;
  - scripts ;
  - etc.
- propagation de vers, etc.
  - Exemple : ILOVEYOU (*Love Bug*) → milliards de \$ de dégats
    - Lettre d'amour ; 3,1 millions de machines infectées en 4 jours
    - Écrit en VBScript, il détruisait aléatoirement des fichiers
    - Utilisait Windows Address Book et Outlook pour se propager
- Utilisation déraisonnable de la messagerie ?
  - Grand nombre de messages
  - Messages de taille importante
- le serveur de messagerie se met hors-ligne

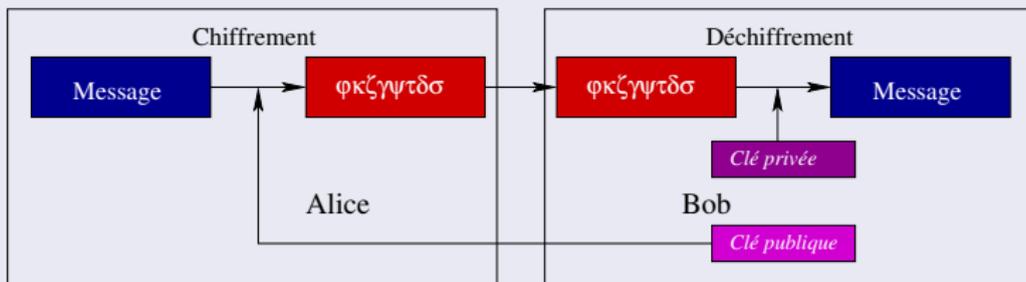
- **Ne pas ouvrir automatiquement les pièces jointes**
- **Une BAL peut être l'objet d'un Mail-bombing**

# Solutions pour sécuriser la messagerie

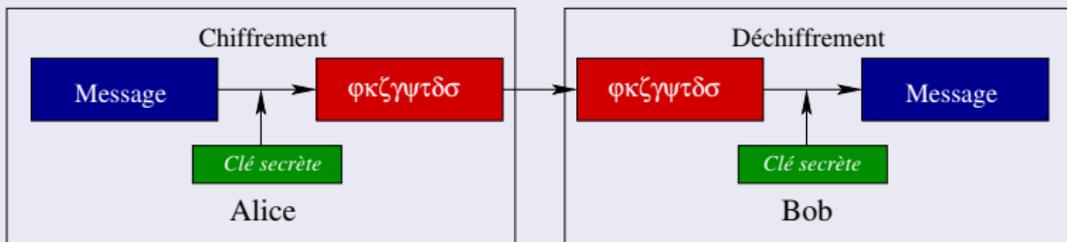
- Sécurisation des flux
  - Sécurisation du contenu  $\Rightarrow$  couche Application
    - Chiffrement et signature électronique des messages
    - Outils : S/MIME et PGP
  - Sécurisation des échanges  $\Rightarrow$  couche Transport
    - Authentification, confidentialité et intégrité des données échangées entre serveurs, entre serveur et agent utilisateur
    - Outils : protocole **Transport Layer Security/SSL** pour sécuriser les protocoles de la couche Application
      - SMTP avec TLS/SSL  $\rightarrow$  port 465 ou 587
      - POP avec TLS/SSL  $\rightarrow$  port 995
      - IMAP avec TLS/SSL  $\rightarrow$  port 993
- Filtrage et analyse du contenu des messages
  - Protection contre les programmes malveillants, le SPAM
- Bonnes pratiques de l'utilisateur
  - Charte d'utilisation

# Sécurisation du contenu - Confidentialité

## Chiffrement asymétrique

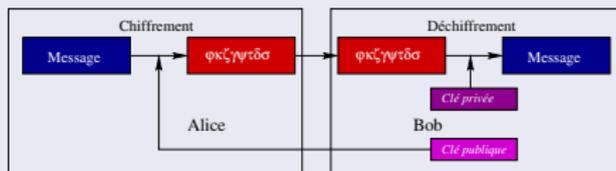


## Chiffrement symétrique

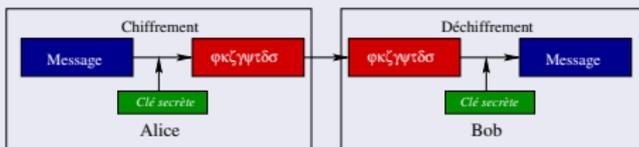


# Sécurisation du contenu - Confidentialité

## Chiffrement asymétrique



## Chiffrement symétrique



- Chiffrement en pratique
  - ① Chiffrement asymétrique pour échanger une clé secrète
  - ② Chiffrement symétrique avec la clé secrète
- Diffusion des clés publiques
  - Serveur de diffusion des clés ; Certificats

# Sécurisation du contenu - Intégrité

## L'expéditeur signe son message

- 1 Calcule une empreinte du contenu → fonction de hachage
  - Contenus différents → empreintes différentes
  - Suite de bits de longueur finie appelée aussi condensat ou haché (*digest* ou *hash*)
  - Pas de prédiction de l'empreinte à partir du contenu
- 2 Chiffre l'empreinte avec sa clé privée, puis l'ajoute au contenu

## Le destinataire vérifie la signature

- 1 Déchiffre l'empreinte dans le message avec la clé publique de l'expéditeur
- 2 Calcule l'empreinte du contenu (sans l'empreinte précédente) et la compare avec l'empreinte obtenue précédemment

# Sécurisation du contenu - Authentification

- Authentification du propriétaire d'une clé publique
- Certificat = carte d'identité numérique
  - Principe
    - Clé publique signée par une Autorité de Certification (CA)
  - La confiance dans une clé résulte de la confiance dans la CA
    - Infrastructure de gestion des clés
    - Autorité d'Enregistrement
      - valide l'identité du propriétaire, signe la demande (Mairie)
    - Autorité de Certification
      - reçoit la demande, signe la clé publique et publie le tout (à l'image d'une Préfecture)
  - Format de certificat standard
    - X.509 v3 (1996)

# Sécurisation du contenu - S/MIME et PGP, HTTPS

## *Secure/Multipurpose Internet Mail Extension*

- Extension du format MIME
- Développé par RSA Data Security Inc.

## *Pretty Good Privacy*

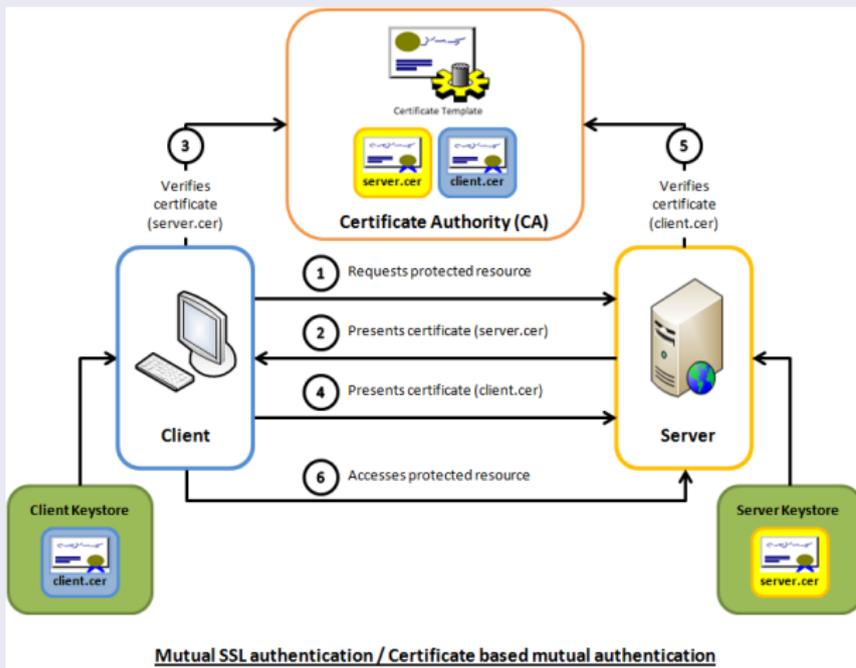
- Même fonctionnalités que S/MIME
- Développé par F. Zimmermann en 1991
- Populaire et bien développé
  - OpenPGP → [www.openpgp.org](http://www.openpgp.org)
  - GNU Privacy Guard → [www.gnupg.org](http://www.gnupg.org)
  - Plugin Enigmail pour Thunderbird, icedove

Tout deux fournissent un ensemble complet de cryptosystèmes : chiffrements asymétriques et symétriques, signature, empreintes

# Sécurisation du contenu - S/MIME et PGP, HTTPS

## Poignée de main (*handshake*) SSL / TLS et passage en HTTPS

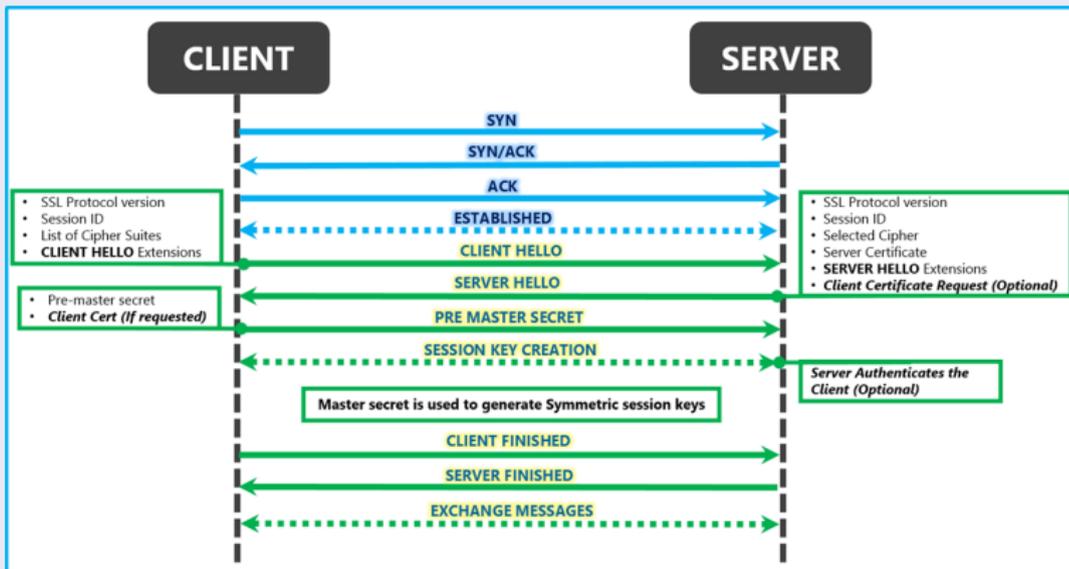
- Authentification mutuelle via des certificats



# Sécurisation du contenu - S/MIME et PGP, HTTPS

## Poignée de main (*handshake*) SSL / TLS et passage en HTTPS

- Construction des clés de chiffrement symétrique



# Partage de fichiers Pair-à-Pair

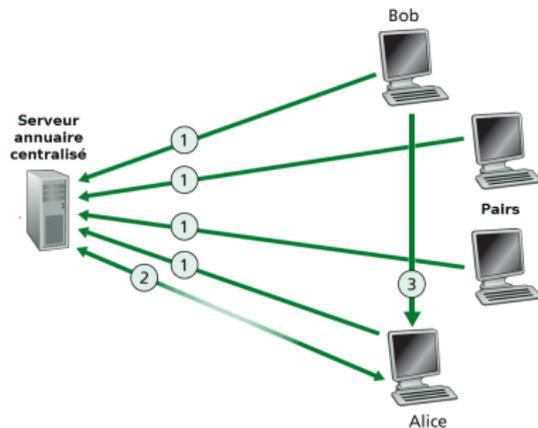
## Exemple

- Alice a une connexion intermittente → adresse IP dynamique
- Alice exécute son client P2P sur son portable
- Cherche *Hey Jude*
- L'application indique les pairs connectés ayant une copie
- Alice en choisit un → Bob
- Copie du fichier de l'hôte de Bob vers le portable d'Alice
- Si Bob se déconnecte, le client P2P d'Alice peut essayer de récupérer le reste du fichier depuis d'autres pairs
- D'autres utilisateurs peuvent charger les fichiers d'Alice
- Le pair d'Alice est à la fois client et un serveur temporaire

Tous les pairs sont serveurs → hautement extensible

# P2P avec un annuaire centralisé

- Conception originale de Napster
  - ➊ À la connexion un pair envoie au serveur central, deux infos :
    - Son adresse IP
    - Son contenu
  - ➋ Alice demande *Hey Jude*
  - ➌ Alice demande le fichier à Bob



## Problème de l'annuaire centralisé

- Point de défaillance et goulot d'étranglement
- Violation des droits d'auteur → fermeture possible
- Fermeture plus difficile si complètement décentralisé

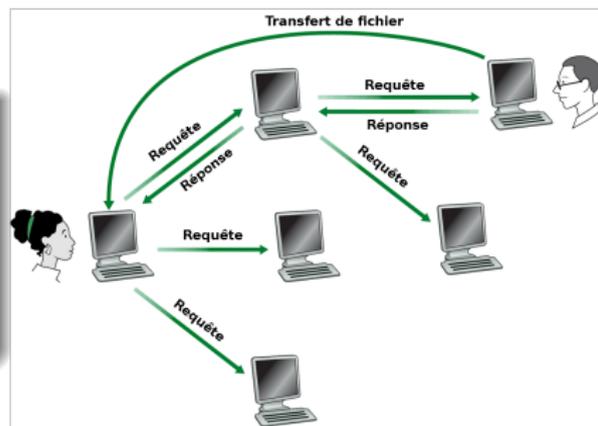
Transfert décentralisé, mais recherche fortement centralisée

# P2P (vrai) sans serveur central - Principe

Gnutella → inondation de requêtes → faiblement extensible

- Totalement décentralisé → pas de serveur
- Protocole du domaine public
- Nombreux clients Gnutella implémentant le protocole
- Création d'un réseau de couverture reliant les pairs

- Requête envoyée sur des connexions existantes
- Propagation par les pairs
- Réponse renvoyée via le chemin inverse



# P2P (vrai) sans serveur central - Connexion des pairs ?

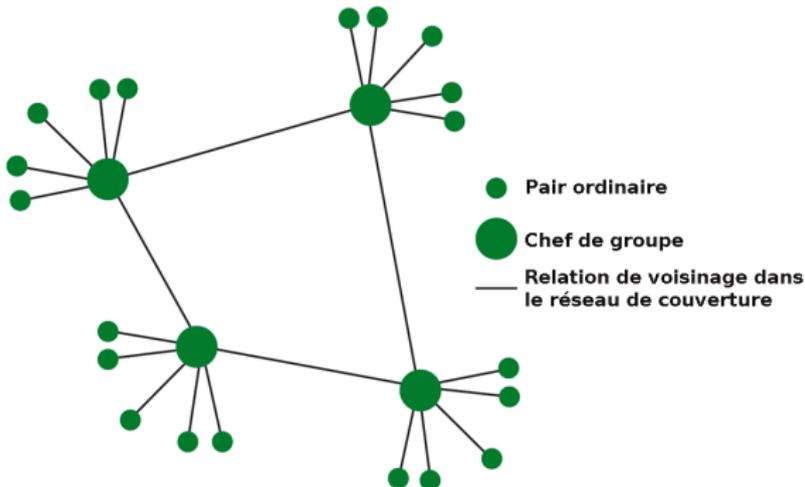
- 1 Un nouveau pair X doit trouver un autre pair Gnutella  
Inondation (ou broadcast) impossible sur Internet  
→ liste de pairs candidats, fixes et actifs
- 2 X essaie chacun des pairs de la liste, jusqu'à établir une connexion TCP avec un pair Y
- 3 X envoie un message ping à Y qui le propage
- 4 Tous les pairs recevant ping répondent avec un pong
- 5 X reçoit les messages pong → établissement de nouvelles connexions TCP

Un pair a généralement moins de 10 voisins

# P2P hybride - Principe

## Kazaa

- Un pair est soit chef de groupe, soit rattaché à un chef de groupe
  - Connexion TCP entre un pair et son chef de groupe
  - Connexion TCP entre quelques paires de chefs
- Un chef de groupe suit le contenu de tous les pairs enfants



# P2P hybride - Requêtes

## Kazaa

- Un fichier ↔ une clé et un descripteur
- Le client envoie une requête de mots clés à son chef de groupe
- Le chef retourne des réponses contenant pour chacune
  - une clé, des méta-données, une adresse IP
- Les requêtes / réponses peuvent être propagées entre chefs
- Le client choisit les fichiers à télécharger

## Spécificités de Kazaa

- File d'attente des requêtes
- Priorités incitatives → plus on partage, plus on est prioritaire
- Téléchargements en parallèle

# Domain Name System

- Identification des hôtes et routeurs connectés à Internet
  - Adresse **Internet Protocol v4** (32 bits) = 193.52.61.135
  - Adresse IPv6 (128 bits) = fe80::213:77ff:fe26:4bcd
- Identification des hôtes par un être humain
  - Nom symbolique = slayer.iut-bm.univ-fcomte.fr

Comment obtenir pour un hôte  
la correspondance entre adresse IP et nom ?

Translation assurée par le **Domain Name System**  
(ou **système de noms de domaine**)

- Base de données distribuée stockant les correspondances
  - Hiérarchie de **serveurs de noms** ;
  - dénommés également **serveurs DNS**
- Protocole de niveau Application

# Domain Name System

Translation assurée par le *Domain Name System*  
(ou **système de noms de domaine**)

- Base de données distribuée stockant les correspondances
  - Hiérarchie de **serveurs de noms** ;
  - dénommés également **serveurs DNS**
- Protocole de niveau Application

## Inconvénients d'un système centralisé ?

- Intolérant aux pannes → Plus d'Internet si arrêt du DNS
- Goulot d'étranglement → Volume de trafic
- Délais → Importants si on est "loin"
- Mise à jour → Combien de fois ? Qui ? Taille des données

Un système centralisé n'est pas adapté vu le nombre d'hôtes, etc.

# Résolution et résolution inverse avec DNS

## Résolution directe ou **résolution de nom de domaine**

- Translation nom symbolique → adresse IP
  - Requête → Quelle est l'adresse IP de `www.univ-fcomte.fr` ?
  - Réponse → L'adresse IP est `194.57.91.224`
- En principe un hôte
  - peut avoir plusieurs alias (surnoms) ;
  - un seul nom dit canonique (`ufc224.univ-fcomte.fr`)

## Résolution inverse ou **résolution d'adresse**

- Translation adresse IP → nom symbolique
  - Requête → Quel est le nom symbolique de `193.52.61.135` ?
  - Réponse → Le nom symbolique est `slayer.iut-bm.univ-fcomte.fr`
- Serveur Web répliqué
  - Plusieurs adresses IP pour un même nom
  - À chaque requête une adresse différente est renvoyée

# Comment est construit un nom symbolique ?

- Un **nom de machine** → slayer
- Un **nom de domaine** → iut-bm.univ-fcomte.fr

## Nom de domaine

- Décomposé en une suite hiérarchique de domaines
  - domaine de niveau 1 → fr
  - domaine de niveau 2 → univ-fcomte.fr
  - sous-domaine → iut-bm
- Habituellement on appelle
  - domaine de niveau 1 → domaine générique
    - Classification par activité com, net, edu, etc.
    - Classification par pays → fr, de, uk, etc.
  - domaine de niveau 2 → domaine
- Un domaine = des hôtes partageant une caractéristique
  - Localisation géographique ; une institution (Université, etc.)
  - etc.

# Une base de données distribuée et hiérarchique

- ① **Serveurs de noms locaux** (primaire et secondaire)
  - Installés par les FAI
  - Un hôte doit connaître au minimum l'adresse d'un serveur local
  - Pas la correspondance
    - contacte un serveur racine
- ② **Serveurs de noms racine** (niveau global)
  - 13 serveurs dans le monde (a.root-servers.net à m.root-servers.net)
  - Pas la correspondance
    - contacte un serveur d'autorité, via un serveur TLD
- ③ **Serveurs *Top Level Domain*** (niveau 1)
  - Répertoire les serveurs d'autorité dans un domaine
  - Exemple : fr
- ④ **Serveurs d'autorité** (niveau 2)
  - Répertoire les correspondances liées à leur sous-domaine
  - Exemple : univ-fcomte.fr

# Serveurs de noms racine

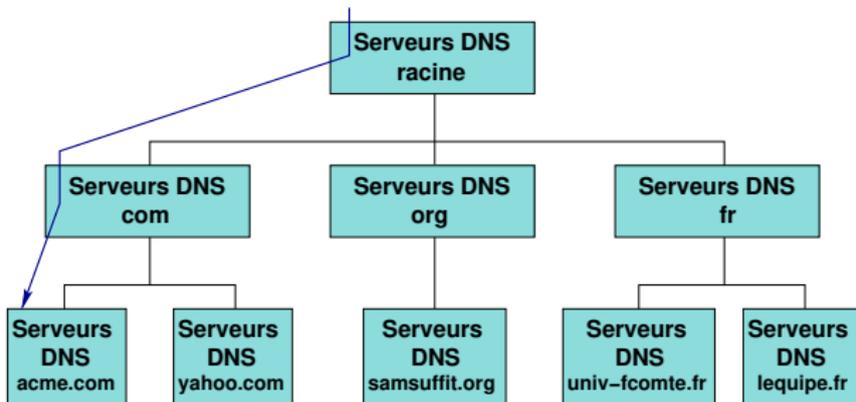
Répondent aux requêtes concernant  
les noms de domaines de niveau 1



# Parcours de la hiérarchie de serveurs

## Résolution de `cartoon.acme.com` (inconnu du serveur local)

- 1 Je demande à un serveur racine de trouver le serveur DNS `com`
- 2 Je demande au serveur DNS `com` de trouver le serveur DNS `acme`
- 3 Je demande au serveur DNS `acme.com` l'adresse IP de `cartoon`



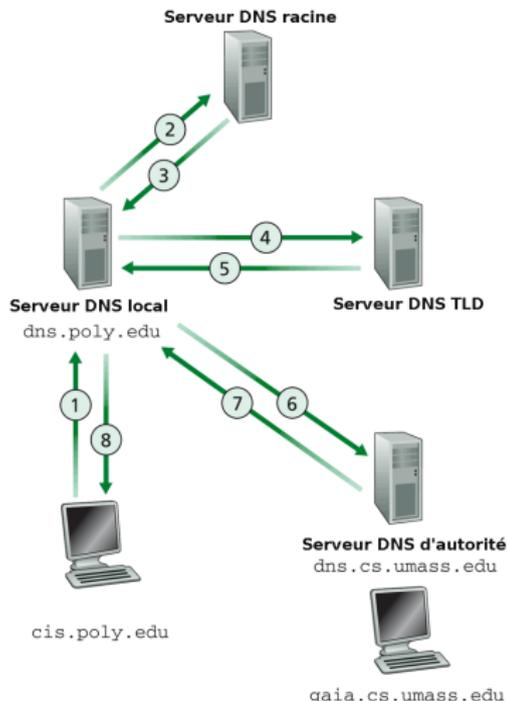
Un serveur de nom local n'appartient pas vraiment à la hiérarchie

# Résolution de nom (détails) - Requêtes itératives

Serveur local contactant les serveurs successivement

## Résolution de `gaia.cs.umass.edu`

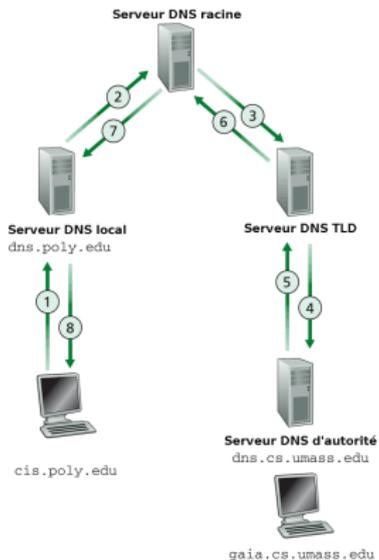
- 1 L'hôte envoie le nom au **serveur local**
- 2 Contact du **serveur racine**
- 3 Retour = liste de serveurs TLD pour le domaine générique `edu`
- 4 Contact d'**un serveur TLD**
- 5 Retour = serveur d'autorité du domaine `umass.edu`
- 6 Contact du **serveur d'autorité**
- 7 Retour = adresse IP de `gaia`
- 8 Transmission de l'adresse IP à l'hôte



# Résolution de nom (détails) - Requêtes récursives ; Mixage

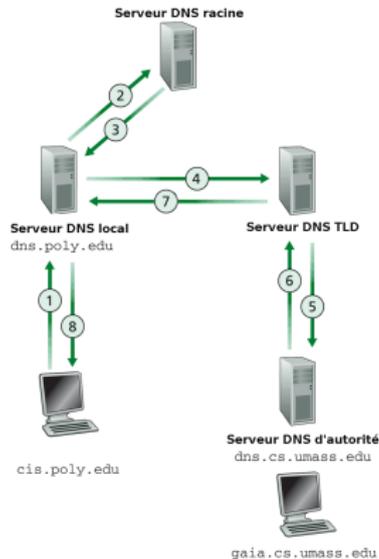
## Requête récursive

- Un serveur contact le suivant
- qui répond au demandeur



## Mixage

- Serveur racine → itératif
- Autres → récursif



# Mise en cache, mise à jour et enregistrement d'un domaine

- Mise en cache
  - Mémorisation pour une certaine durée de chaque correspondance apprise par un serveur
  - Serveurs TLD cachés sur les serveurs locaux
    - Peu de visites des serveurs racines
- Mise à jour en 24 à 48 heures
- Enregistrement d'un domaine disponible
  - Via un bureau d'enregistrement → Officier d'état civil
  - Informations à fournir
    - Noms et adresses IP des serveurs d'autorité (primaire, second.)
    - Nom et adresse IP du site Web associé
    - Nom et adresse IP du serveur de messagerie associé
  - Informations insérées dans les serveurs TLD par le bureau