

## Architecture des Réseaux (ARes) 2/5 : Application

Olivier Fourmaux  
(olivier.fourmaux@sorbonne-universite.fr)

Version 8.0



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

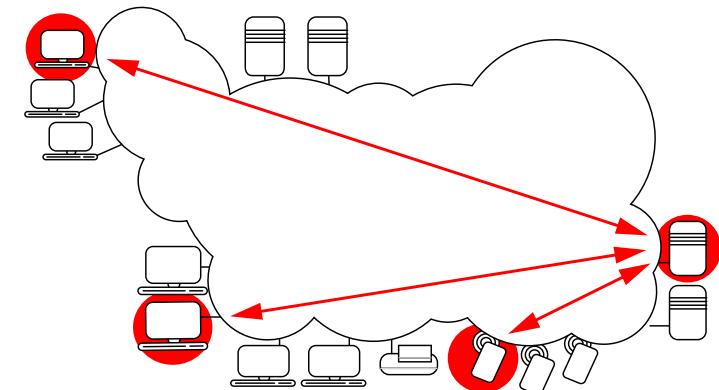
- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## Applications



## Couche Application

### Definition

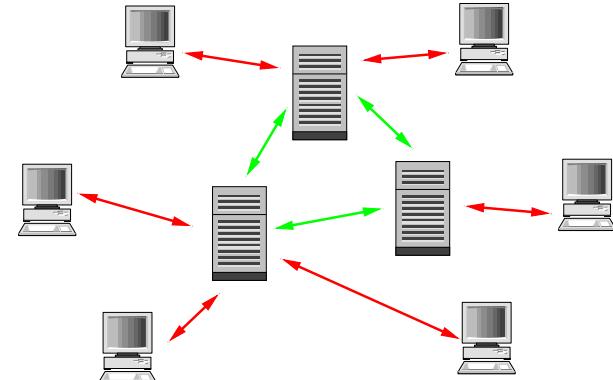
**La couche application** Ensemble des programmes et protocoles de haut niveau qui permettent aux utilisateurs de communiquer

### Remarques :

- standardise les échanges entre les applications les plus courantes
  - accès au web (HTTP), envoi d'e-mail (SMTP, POP, IMAP) ...  
 applications ≠ protocoles de la couche application
- définit l'interface réseau avec les utilisateurs
  - s'appuie sur les services de bout-en-bout définis dans les couches inférieures
- supporte les environnements hétérogènes

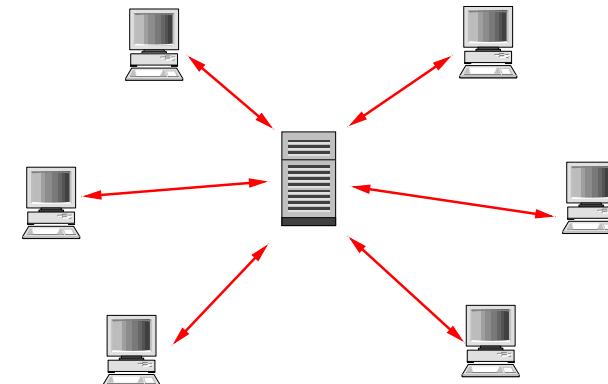
## Architectures (2)

### Client/serveur avec réPLICATION des serveurs



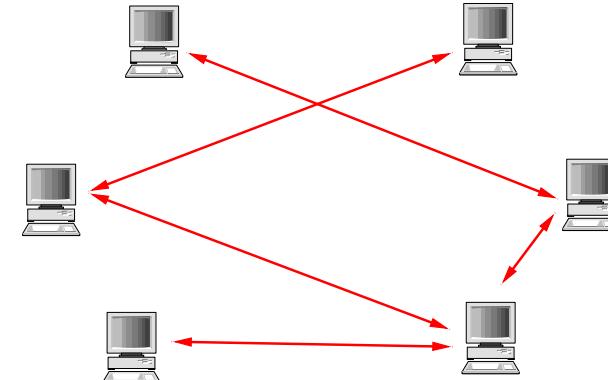
## Architectures (1)

### Client/serveur centralisé classique

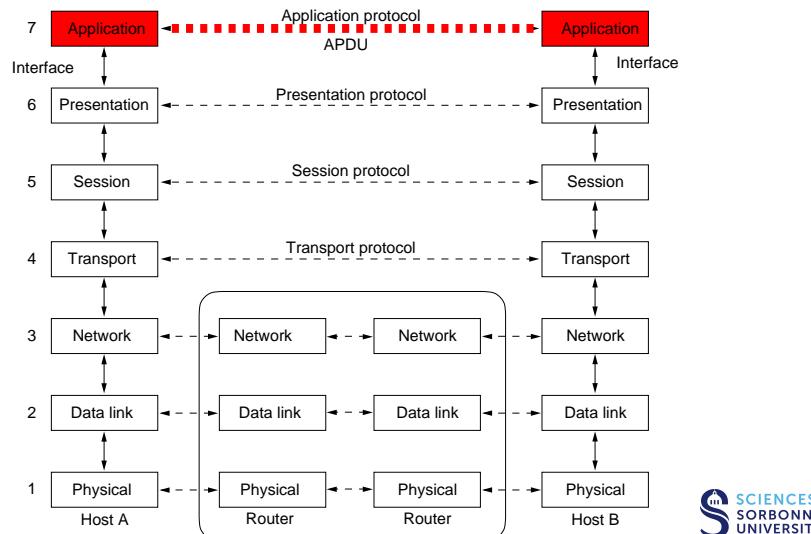


## Architectures (3)

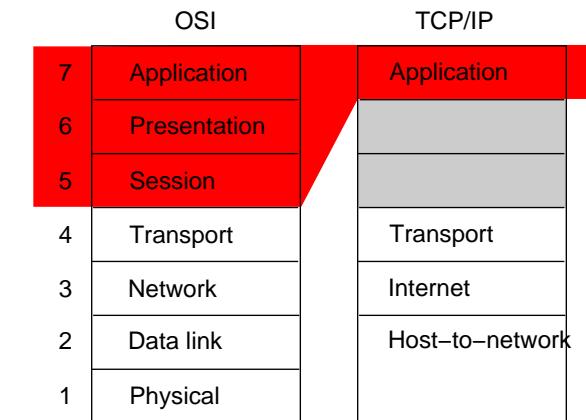
### Peer-to-peer classique



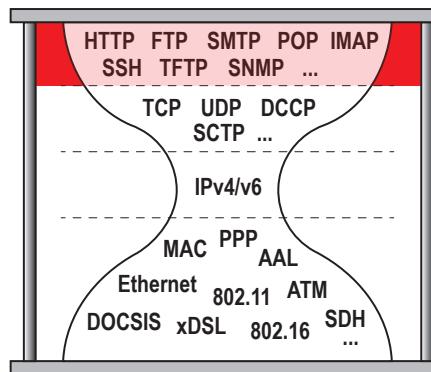
## Couche Application : modèle OSI



## Couche Application : modèle TCP/IP (1)



## Couche Application : modèle TCP/IP (2)



Dans l'Internet, des centaines de protocoles applicatifs existent !

HTTP pour surfer sur la toile

FTP pour transférer des données

SMTP pour échanger du courrier électronique...

## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau

## Applications de connexion à distance

A partir d'un terminal ouvert sur une machine locale, connexion sur une machine distante

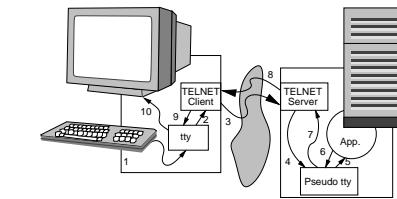
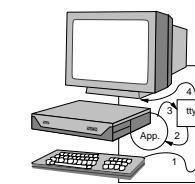
- plusieurs protocoles :
  - TELNET
  - RLOGIN
  - SSH...
- application de type client/serveur
  - client : interagit avec l'utilisateur et les protocoles réseaux
  - serveur : idem au niveau de l'application distante
- besoin d'interactivité
  - tout ce qui tapé sur le clavier local est envoyé **rapidement** sur la connexion
  - tout ce qui est reçu de la connexion est affiché **rapidement** sur l'écran local



## TELNET (*TELecommunication NETwork protocol*)

Application développée dès 1969 (RFC 15) et standardisé à l'IETF en 1983 (RFC 854 et Internet Standard STD 8)

- repose sur une connexion **TCP** (port serveur = 23)
- mécanisme de négociation d'options
- service de terminal virtuel
- pas de confidentialité (mot de passe en clair...)



## TELNET : options

Plusieurs échanges initiaux pour les options (RFC 855) :

- le client émet des **requêtes** (WILL WON'T DO DON'T)
  - Command: Do Suppress Go Ahead
  - Command: Will Terminal Type
  - Command: Will Negotiate About Window Size
  - Command: Will Terminal Speed...
- le serveur renvoie des **réponses** (DO DON'T WILL WON'T)
  - Command: Do Terminal Type
  - Command: Will Suppress Go Ahead
  - Command: Dont Negotiate About Window Size
  - Command: Do Terminal Speed...
- chaque extrémité implémente une version minimale du NVT
  - négociation d'options pour les machines plus évoluées



## TELNET : NVT

Définition d'un terminal virtuel (*Network Virtual Terminal*)

- pas de format de message, mais un en codage des données
- codage vers un système de représentation commun : **NVT**
  - chaque système peut transcoder
  - terminal local réel ⇔ terminal réseau virtuel**
- Exemple :
  - local* : cc maa<bs>x.c
  - NVT* : **c . x IAC EC a a m** c cIAC = *Interpret As Command* (octet valeur 255)
  - il n'est pas nécessaire de connaître la conversion vers chaque type de machine
  - communication dans les **environnement hétérogènes**
  - contrôle **in-band**



## TELNET : Accès à d'autres serveurs

Exemple d'accès à un serveur web avec TELNET :

```
Unix> telnet hobbes.lip6.fr 80
      Trying 137.86.111.77...
      Connected to hobbes.lip6.fr.
      Escape character is ']'.
GET /index.html HTTP/1.0
HTTP/1.1 200 OK
Date: Tue, 24 Sep 2002 15:33:07 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
...
Connection closed by foreign host.
```

➡ connexion TCP brute (limitation NVT)



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## SSH (Secure SHell)

- **communications cryptées**, assurant :
  - authentification
  - confidentialité
  - intégrité
- repose sur une connexion **TCP** (port serveur = 22)
  - rajoute une couche transport intermédiaire
  - authentification cryptée
  - négociation des algorithmes
  - (mux. de sessions, tunnels : X11, relayage de port, SOCKS...)
- standardisation tardive (janvier 2006) : RFCs 4251 à 4254
- nombreuses implémentations
  - OpenSSH (natif sur BSDs, GNU/Linux, MacOSX, Cygwin...)
  - PuTTY (Windows et Unixes)...



## Applications de transfert de fichiers usuelles

Copie d'un fichier d'un système vers un autre en environnement hétérogène

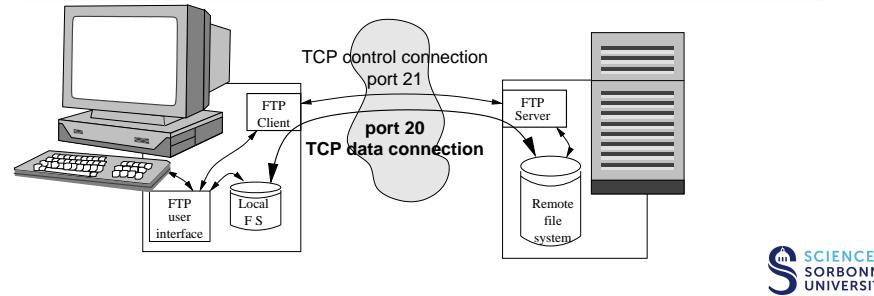
- plusieurs protocoles :
  - FTP
  - TFTP
  - RCP, SCP, SFTP...
- application de type client/serveur
  - **client** : interagit avec l'utilisateur, le système de fichier local et les protocoles réseaux
  - **serveur** : interagit avec les protocoles réseaux et le système de fichier distant
- ne pas confondre avec les systèmes de fichiers distants
  - NFS (Sun, TCP/IP), SMB (Microsoft)...



## FTP (File Transfer Protocol)

Standard TCP/IP pour le transfert de fichiers (RFC 959)

- signalisation **out-of-band**, deux connexions **TCP**
- **accès interactif**
- contrôle d'accès (mais mot de passe en clair)



## FTP : Données

Nombreuses représentations des données (hôtes hétérogènes) :

- type de fichiers :
  - **non structurés**
  - enregistrements
  - pages
- encodeur des données :
  - **ASCII** (*American Standard Code for Information Interchange*)
  - **EBCDIC** (*Extended Binary-Coded Decimal Interchange Code*)
  - **binaire**
- type de transmission :
  - **flux**
  - **bloc**
  - **compressé**

⇒ vérifier le type de données transférées

## FTP : Connexions

Deux connexions **TCP** sont utilisées en parallèle :

- connexion de **contrôle**
  - **permanente** (créeée à l'ouverture de la session FTP)
  - full duplex initiée par le client (port serveur = **21**)
  - utilisée uniquement pour échanger les **requêtes** et **réponses**
  - besoin d'**interactivité** (et de fiabilité)
- connexion de transfert de **données**
  - **temporaire** (créeée à chaque transfert de fichier)
  - full duplex (initiée par le **serveur**)
    - transmission préalable du **port client** à utiliser
  - envoi de fichiers et de liste de fichiers/répertoires
  - besoin de **débit** (et de fiabilité)
  - libérée à la fin de chaque transfert de fichier

## FTP : Requêtes

Codage ASCII NVT ⇒ Mode interactif possible (si lisible)

```
Unix> telnet galion.ufr-info-p6.jussieu.fr 21
Trying 197.18.176.12...
Connected to localhost.
Escape character is '^'.
220 ProFTPD 1.2.0pre10 Server (Debian) [galion.ufr-info-p6.jussieu.fr]
help
214-The following commands are recognized (* =>'s unimplemented).
214-USER    PASS    ACCT*    CWD    XCWD    CDUP    XCUP    SMNT*
214-QUIT    REIN*   PORT    PASV    TYPE    STRU*   MODE*   RETR
214-STOR    STOU*   APPE    ALLO*   REST    RNFR    RNTO   ABOR
214-DELE    MDTM   RMD     XRMDF   MKD    XMKD    PWD    XPWD
214-SIZE    LIST    NLST    SITE    SYST    STAT    HELP    NOOP
214 Direct comments to root@galion.ufr-info-p6.jussieu.fr.
```



Ne pas confondre avec les commandes de l'interface utilisateur de **ftp**

## Commandes utilisateur du programme ftp

```
Unix> ftp
ftp> help
Commands may be abbreviated. Commands are:
!      debug      mdir      sendport    site
$      dir        mget      put         size
account disconnect mkdir      pwd        status
append  exit       mls       quit       struct
ascii   form       mode      quote      system
bell    get        modtime   recv      sunique
binary  glob       mput      reget     tenex
bye    hash       newer     rstatus   tick
case   help       nmap      rhelp     trace
cd     idle       nlist     rename   type
cdup   image     ntrans    reset    user
chmod  lcd        open      restart  umask
close  ls         prompt   rmdir    verbose
cr    macdef    passive  runique ? SCIES
delete mdelete   proxy    send    SORBOUN
                                         NE UNIVERSITE
```

Olivier Fourmaux (olivier.fourmaux@ sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## FTP : Exemple

### Programme ftp (interface utilisateur)

```
[toto@hobbes]$ ftp calvin.lip6.fr
Connected to calvin.lip6.fr.
220 FTPD 1.2pre8 Server (Debian)
Name (calvin.lip6.fr):toto
331 Password required for toto.
Password:
230 User toto logged in.
ftp> get toinst.txt remote: toinst.txt
local: toinst.txt
200 PORT command successful.
150 Opening BINARY mode data connection
for toinst.txt (1 bytes).
226 Transfer complete.
1 bytes received in 0.377s (0.0026 KB/s)
ftp> quit
221 Goodbye.
[toto@hobbes]$
```

### Protocole FTP (connexion de contrôle)

```
220 FTPD 1.2pre8 Server (Debian)
USER toto
331 Password required for toto.
PASS AB]Ga!9F
230 User toto logged in.

PORT 192,33,82,12,4,15
200 PORT command successful.
RETR toinst.txt
150 Opening BINARY mode data connection
for toinst.txt (1 bytes).
226 Transfer complete.

QUIT
221 Goodbye.
```



Olivier Fourmaux (olivier.fourmaux@ sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## FTP : Réponses

Codage usuel : *status + description textuelle*

status	description	status	description
x0z	syntaxe		
1yz	réponse positive préliminaire	x1z	information
2yz	réponse positive complète	x2z	connexions
3yz	réponse positive intermédiaire	x3z	authentification
4yz	réponse négative transitoire		
5yz	réponse négative définitive	x5z	système de fichier

- 150 Opening BINARY mode data connection
- 200 Command successful
- 220 ProFTPD 1.2.0pre10 Server (Debian)
- 226 Transfer complete
- 230 User toto logged in
- 331 Username OK, Password required
- 425 Can't open data connection
- 500 Syntax error (Unknown command)...



Olivier Fourmaux (olivier.fourmaux@ sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## FTP : Divers

**Anonymous** : compte invité sur certains serveurs FTP :

- username: anonymous
- password: adresse@electronique.org

**Mode passif** : ouverture de la connexion donnée en sens inverse

- si l'ouverture usuelle de la **connexion donnée** impossible
  - filtrage des adresses (*firewall*)
  - translation d'adresses (NAT)
- commande PASV (RFC 1579)
  - le client envoie la commande PASV au serveur a.b.c.d
  - le serveur alloue le port 256x+y, fait une ouverture passive sur ce port et en informe le client avec une réponse
    - ⇒ 227 Entering passive mode (a,b,c,d,x,y)
  - le client fait une ouverture active vers le port 256x+y



Olivier Fourmaux (olivier.fourmaux@ sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## TFTP (*Trivial File Transfer Protocol*)

- protocole léger pour le transfert de fichiers (version 2 : RFC 1350)
- datagrammes UDP sur le port 69

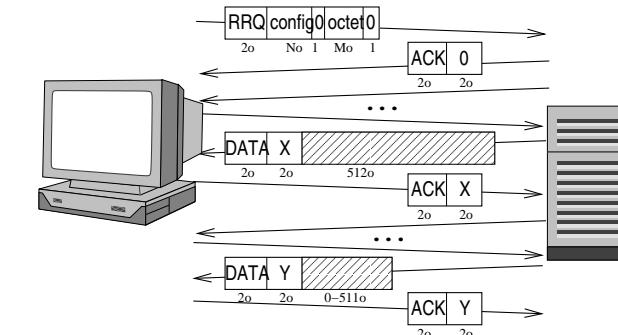
5 messages :

opcode	nom	description
1	RRQ	requête de lecture
2	WRQ	requête d'écriture
3	DATA	données numérotées
4	ACK	acquittement
5	ERREUR	message d'erreur

- messages DATA avec 512 octets (sauf le dernier de taille inférieure ou éventuellement nulle)
- protocole *stop-and-wait*
  - numérotation des messages DATA
  - acquittement immédiat avec ACK
- pas de contrôle d'accès (sous Unix, souvent limité à /tftpboot)

## TFTP : Exemple

```
[toto@hobbes]$ tftp calvin.lip6.fr
tftp> get config
Received 5220 bytes in 0.377 secs
tftp> quit
[toto@hobbes]$
```



## RCP, SCP, SFTP

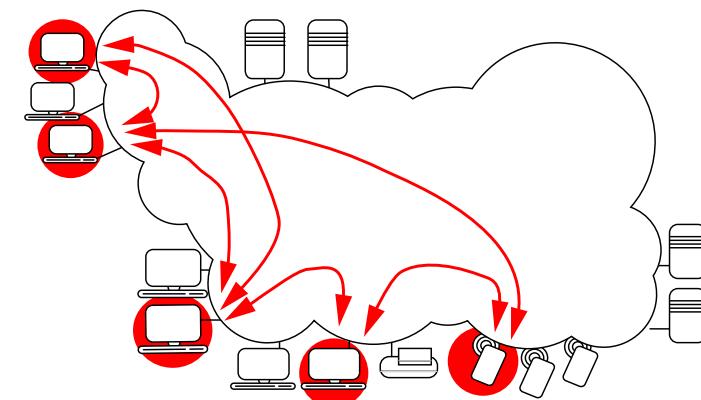
### Protocole R\* : RCP

- spécifique à Unix et associé aux r\* commandes (dont rcp)
  - le client rcp fonctionne avec un serveur rshd
  - idem rlogin : obsolète, problèmes de sécurité...

### Protocoles sécurisés : SCP, SFTP

- scp : copie simple similaire à rcp encapsulé dans SSH
- sftp : idem FTP mais facilement encapsulable
  - SFTP est un nouveau protocole (groupe IPSEC de l'IETF)
  - SFTP peut être utilisé avec SSH (par défaut avec de nombreux clients sftp)
  - SFTP est différent de FTPS qui introduit la sécurisation au niveau des connexions avec SSL/TLS (*Secure Socket Layer/Transport Layer Security*)

## Applications de partage de fichier Peer-to-peer



## De nombreuses applications Peer-to-peer...

... mais peu de standards

- **partage de fichiers**

- Napster, eDonkey, BitTorrent...
- FastTrack (KaZaA, Grokster et Imesh), Gnutella2...
- Gnutella...
- BitTorrent

- **distribution de flux audio/vidéo**

- VoD : Peercast, Joost...
- P2PTV : Coolstreaming, TVants, PPVLive...

- **stockage anonyme**

- Freenet, Entropy...

- **calcul distribué**

- ...



## Napster



Programme de partage de fichiers MP3

- pas le premier, mais le plus connu.
  - très informatif sur l'intérêt du *peer-to-peer*...  
... sur les problèmes techniques, légaux, politiques, économiques
- une technologie de rupture?
  - historique
    - fin 98 : Shawn Fanning (19 ans) débute le développement
    - 05/99 : création de *Napster Online Music Service*
    - 06/99 : premiers tests du logiciel
    - 12/99 : premières poursuites juridiques (Metallica, RIAA...)
    - mi 00 : plus de **60M** d'utilisateurs  
importante part du trafic universitaire (30% à 50%)
    - 02/01 : jugement par la Cour d'Appel des US
    - mi 01 : 160K utilisateurs...

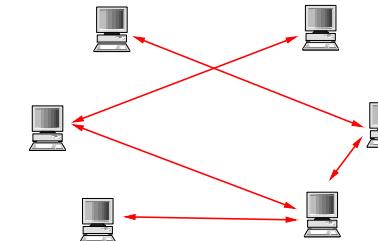


## Fonctionnalités P2P

Caractéristiques des systèmes *peer-to-peer*

- pas de rôle distinct

- évite les points de congestion ou les nœuds défaillants
- besoin d'algorithmes distribués
  - découverte de service (nommage, adressage, calcul de métriques)
  - maintien d'un état du voisinage
  - routage au niveau applicatif (lié au contenu)
  - robustesse, gestion de perte de liens ou de nœuds ...



## Napster : Principe

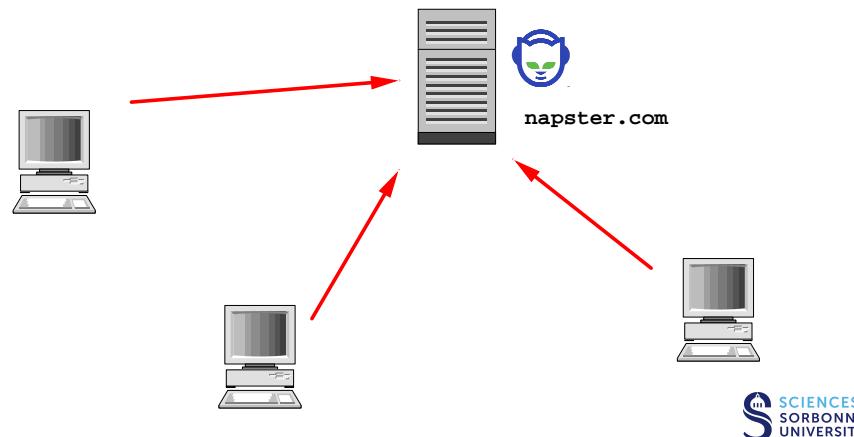
Approche mixte :

- recherche client/serveur avec liste centralisée
- échange direct des données recherchées entre pairs
- connexions point à point TCP (port 7777 ou 8888)
- 4 étapes :
  - Connexion au serveur Napster
  - Envoi de sa liste de fichiers au serveur (*push*)
  - Envoi des mots recherchés et récupération d'une liste de pairs
  - Sélection du meilleur pair (*pings*)



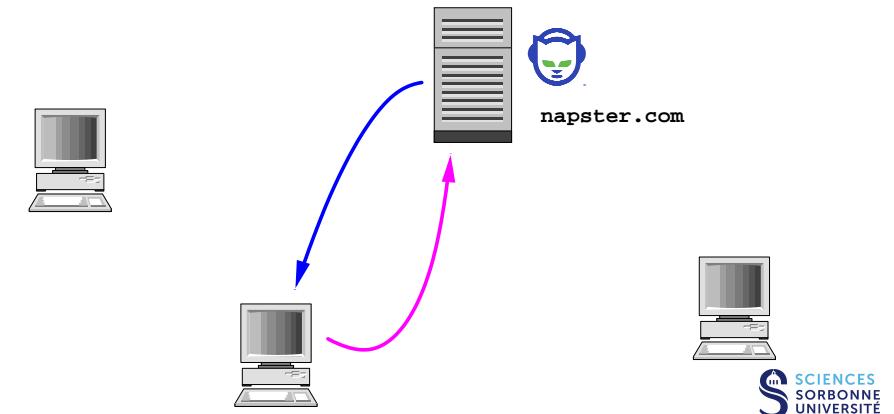
## Napster : Upload

Les utilisateurs chargent la liste des fichiers à partager



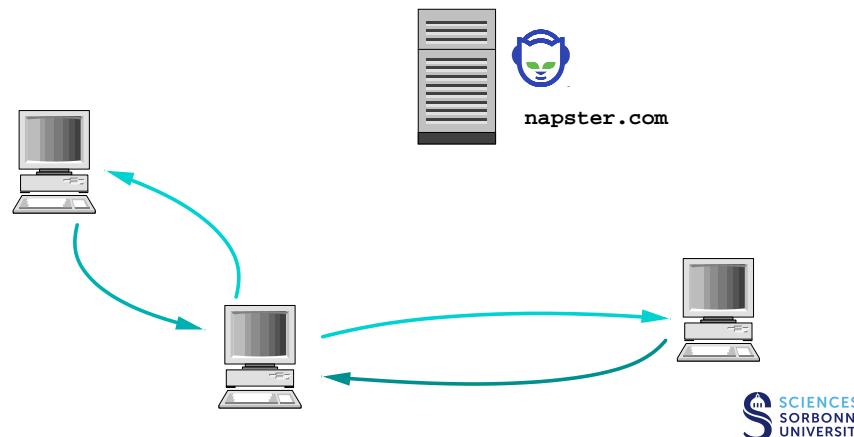
## Napster : Search

Un utilisateur émet une requête de recherche  
Le serveur indique les localisations potentielles



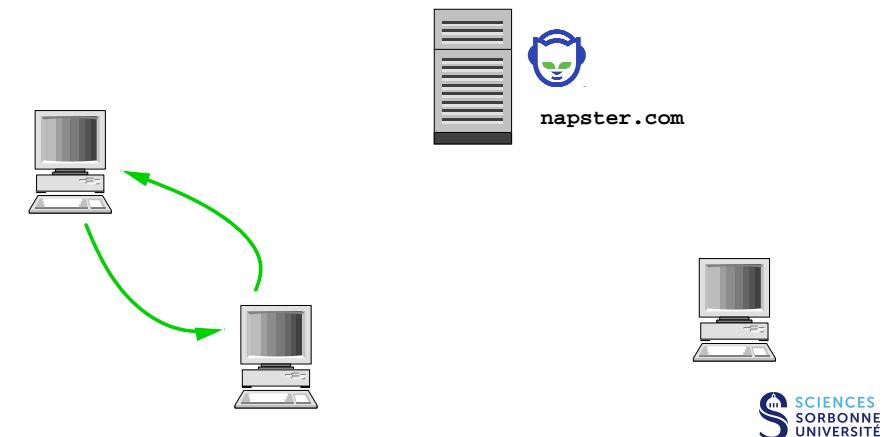
## Napster : Pings

Ping des pairs potentiels (recherche du meilleur débit de transfert)



## Napster : Download

L'utilisateur récupère directement le fichier chez le pair choisi



## Napster : remarques

- serveur centralisé
  - un point de défaillance
  - risque de congestion
  - partage de charge en utilisant la rotation DNS
  - contrôlé par une entreprise
- absence de sécurité
  - mot de passe en clair
  - pas d'authentification
  - pas d'anonymat
  - code propriétaire
  - téléchargement de mises-à-jour



## Gnutella : motivations (2)

### Peer-to-peer Networking

- connexion exclusive entre les applications des pairs
- problème :
  - recherche de fichiers **décentralisée**
- chaque application :
  - stocke une sélection de fichiers
  - oriente les requêtes de recherche (*route query*) de et vers ses pairs
  - répond aux demandes de transfert de fichiers
- historique
  - 03/00 abandon du projet *freelance* au bout de qqs jours après son lancement par AOL (Nullsoft)
  - trop tard : déjà plus de 20K utilisateurs...
  - nombreux correctifs facilité par l'approche *open source*



## Gnutella : motivations



Partage de fichiers complètement distribué

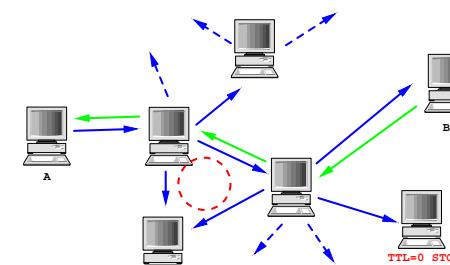
- corrige les défauts majeurs de Napster :
  - *Open source*
  - pas de serveurs - pas d'index global
  - connaissance locale seulement
- mais toujours les mêmes problèmes juridiques, économiques...
  - pas de responsable direct du service
  - absence d'anonymat
- le RIAA attaque directement des utilisateurs !



## Gnutella : principe

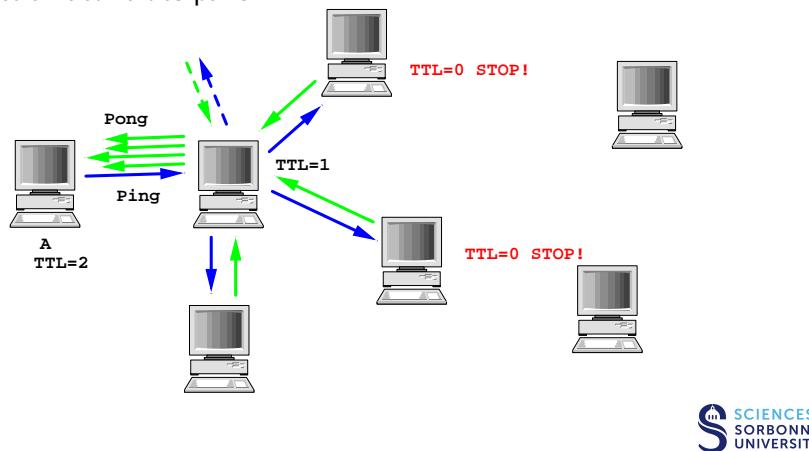
### Recherche par inondation (*flooding*)

- si je n'ai pas le fichier recherché :
  - je demande à 7 pairs s'ils ont ce fichier
- s'ils ne l'ont pas, ils contactent à leur tour 7 pairs voisins
  - recherche récursive limitée à N sauts
- détection de boucle par mémorisation temporaire des requêtes
  - les messages peuvent être reçus deux fois



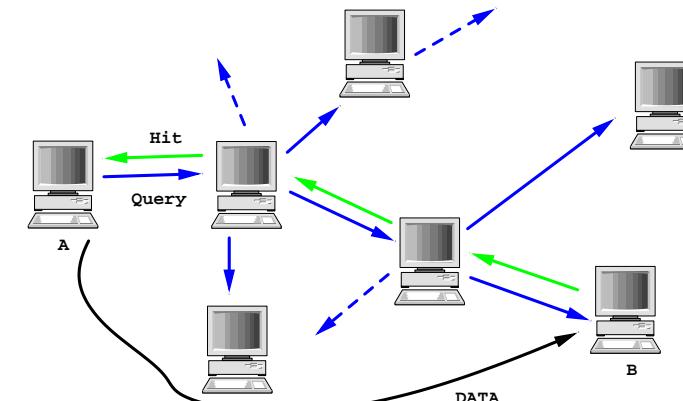
## Gnutella : identification des pairs

Détection active des pairs



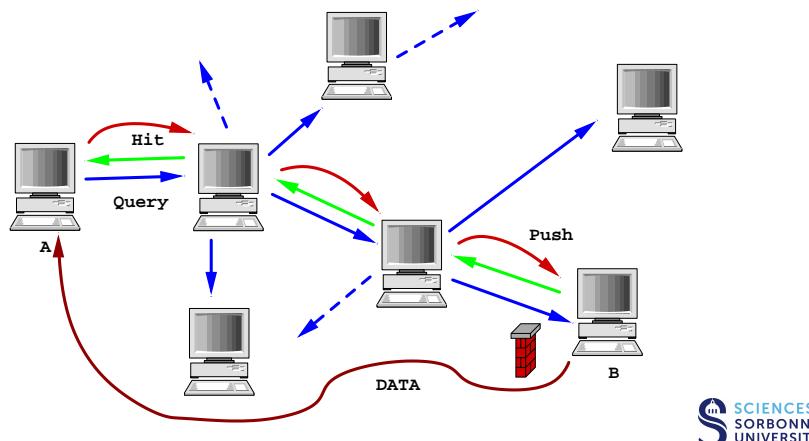
## Gnutella : recherche de fichier

Requête pour trouver une information



## Gnutella : Firewall (2)

Retourner la connexion des données



## Gnutella : Remarques

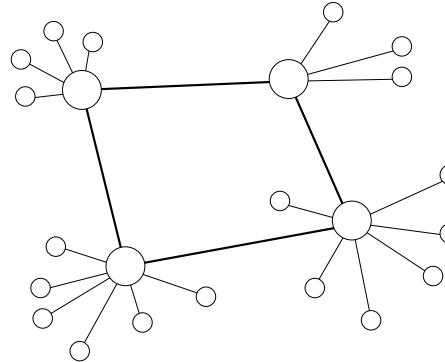
Leçons retenues :

- saturation des petits pairs (modems)
- possibilité d'indiquer que l'on a un fichier mais que l'on est saturé
- taille du réseau atteignable limitée (rupture de connectivité liée aux modems)
- création d'une hiérarchie de pairs
- anonymat ?
  - le pair où l'on récupère le fichier nous connaît
  - ∃ protocoles permettant de ne pas connaître le destinataire

## Evolutions P2P

Gnutella2, KaZaA (réseau FastTrack)...

- hôtes hétérogènes
- topologie hiérarchique
  - *Super-Nodes*



## BitTorrent (1)

### Partage d'un fichier :

- découpage en bloc de 64Ko à 1Mo (*Chunk*)
- création d'un .torrent
  - métadonnées
  - signature pour chacun des *chunks*
- mise en place d'un *tracker*
  - machine qui supervise la distribution
- échange de données entre tous les demandeurs (*leechers*)
  - la source (*seed*) n'est sollicitée que pour amorcer

### Spécificité :

- pas de système de recherche
- pas de téléchargement direct (type HTTP)
- avantages :
  - économique
  - redondant
  - résistance aux *flash-crowd*



## BitTorrent (2)

Stratégies :

- sélection de pair
  - *tit-for-tat* + *choking*
    - encourage la coopération et diminue les *free-riders*
    - sélectionne les meilleurs contributeurs, étouffe les autres
    - mécanisme périodique (10 s)
  - *optimistic unchoke*
    - découverte de nouveaux pairs
    - alimente un nouveau pair aléatoirement
    - mécanisme périodique (30 s)
- sélection de *chunk* :
  - *rarest first*
    - donne le *chunk* le plus rare en premier
    - maximise l'entropie de chaque *chunk*
  - *random first*
    - pour accélérer le démarrage des nouveaux



## BitTorrent (3)

### Evolutions :

- Indexage/recherche
  - initialement moteurs de recherche spécialisés (web)
  - *tracker* distribué (table de hachage distribuée)
    - basée sur Kademlia
- *multitracker*
  - redondance
  - surcouche en signalisation
- cryptage des échanges
  - *Protocol header encrypt (PHE)*
  - *Message stream encryption/Protocol encryption (MSE/PE)*
- distribution de contenus (*streaming A/V*)
  - nombreux projets...



## P2P : autres

Systèmes *peer-to-peer* structurés de recherche par le contenu :

- **Chord**
  - identificaton par clé (SHA-1 sur une chaîne)
  - localisation par clé (SHA-1 sur l'adresse du noeud)
    - positionnement sur le noeud successeur le plus proche
- **Tapestry**
  - routage des identificateurs (hash) selon le suffixe des noeuds
- **CAN (Content Addressable Network)**
  - système de coordonnées cartésiennes virtuelles ...



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

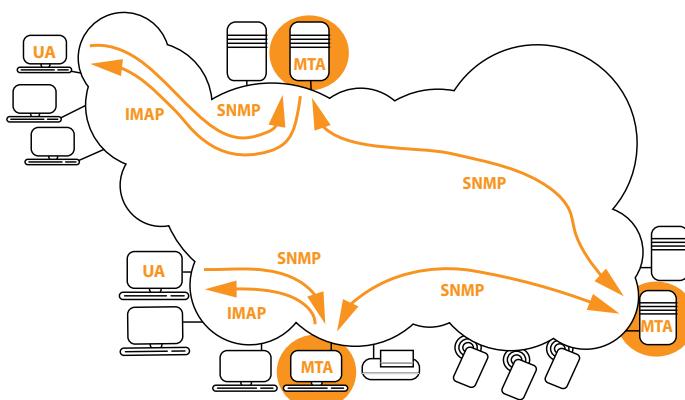
- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## Application de messagerie électronique



## SMTP : introduction

Echange de messages asynchrones à travers l'Internet

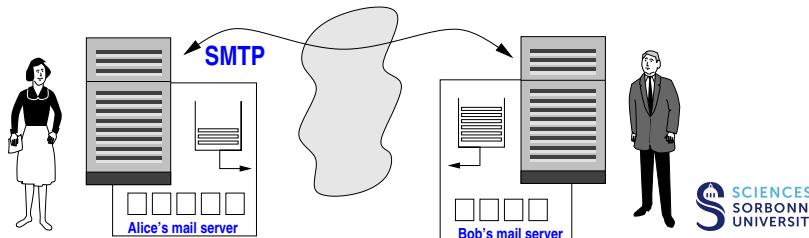
- la première "killer app."
- trois éléments de base :
  - UA (*User Agent*)
    - mail, elm, pine, mutt...
    - Eudora, Outlook et MS Mail, Mail.app, Mozilla Thunderbird...
  - serveurs de mail ou MTA (*Mail Transfer Agent*)
    - sendmail...
    - compose l'**infrastructure** du système de distribution
    - **boîtes aux lettres** des utilisateurs locaux
    - file d'attente des messages au départ ou en transit
    - temporisation et **reprise** si destinataire inaccessible
  - un protocole : **SMTP**



## SMTP : principes

Simple Mail Transfer Protocol (RFC 821 - STD 10, mise-à-jour RFC 5321)

- application client/serveur
- repose sur le service fiable des connexions **TCP**
- ancien
  - ✓ largement répandu
  - ✗ messages encodées en ASCII NVT
- connexion aux serveurs mail sur le port **25**



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application



## SMTP : commandes (1)

Serveur SMTP en mode interactif :

```
Unix> telnet galion.ufr-info-p6.jussieu.fr 25
Trying 192.133.82.123...
Connected to galion.ufr-info-p6.jussieu.fr
Escape character is '^]'.
220 galion.ufr-info-p6.jussieu.fr SMTP Sendmail 8.9.3; Wed, 25 Sep 2002 00:54:15 -help
214-This is Sendmail version 8.9.3
214-Topics:
214- HELO MAIL RCPT DATA
214- QUIT VRFY NOOP RSET
214- HELP
214-For more info use "HELP <topic>".
214-To report bugs in the implementation send email to
214-   sendmail-bugs@sendmail.org.
214-For local information send email to Postmaster at your site
214 End of HELP info
```



## SMTP : exemple

```
220 hobbes.lip6.fr SMTP Sendmail 8.9.3; Wed, 22 Sep 2008 00:59:49 +0200
HELO calvin.lip6.fr
250 hobbes.lip6.fr Hello calvin.lip6.fr, pleased to meet you
MAIL FROM: pere-noel@hobbes.lip6.fr
250 pere-noel@hobbes.lip6.fr... Sender ok
RCPT TO: totu@hobbes.lip6.fr
550 totu@hobbes.lip6.fr... User unknown
RCPT TO: toto@hobbes.lip6.fr
250 toto@hobbes.lip6.fr... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Cher Toto,
N'oubliez pas de m'envoyer votre liste de cadeaux
Le Pere Noel.

.
250 BAA01090 Message accepted for delivery
QUIT
221 hobbes.lip6.fr closing connection
```



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## SMTP : commandes (2)

Commandes SMTP de base (RFC 821), ensemble minimal :

HELO	Présentation du nom de domaine du client
MAIL	Identification de l'expéditeur du message
RCPT	Identification du destinataire du message
DATA	Envoi du contenu jusqu'à une ligne avec seulement un ". "
QUIT	Termine l'échange de courrier
VRFY	Vérification de l'adresse du destinataire
NOOP	Pas d'opération, force le serveur à répondre
RSET	Annule la transaction



## SMTP : réponses

Codage lisible usuel :

- *status + description :*
  - 220 SMTP Sendmail 8.9.3
  - 221 Closing connection
  - 250 Command successful
  - 354 Enter mail, end with "." on a line by itself
  - 550 User Unknown



## Evolution de l'enveloppe : ESMTP

Quelques commandes ESMTP (RFC 1425, mise-à-jour RFC 5321) :

EHLO	Utilisation de ESMTP et présentation du client
SIZE	Taille maximum de message acceptée par le serveur
8BITMIME	Possibilité d'envoyer le corps encodé sur 8 bits
X???	Extension SMTP locale

Négociation des extensions ESMTP :

```
EHLO hobbes.lip6.fr.  
250-hobbes.lip6.fr Hello [62.62.169.227], pleased to meet you  
250-PIPELINING  
250-SIZE 35000000  
250-VRFY  
250-STARTTLS  
250-ENHANCEDSTATUSCODES  
250-8BITMIME  
250 DSN
```



## SMTP : format des messages initiaux

Messages codés en ASCII NVT (RFC 822, mis-à-jour RFC 2822)

- **l'enveloppe**
  - modifiée par entités SMTP successives
    - commandes MAIL FROM: et RCPT TO:
- **le message**
  - principalement inséré par l'agent utilisateur
    - commande DATA
- **entête**
  - chaque champ sur une ligne => nom: valeur
    - From: Toto at Paris 13 <toto@galere.univ-paris13.fr>
    - Date: Mon, 22 Sep 2003 01:13:20 +0200
    - To: Titi at Paris 6 <titi@hypnos.lip6.fr>
    - Subject: rapport TER
    - X-Scanned-By: isis.lip6.fr
- une ligne vide
- **corps**
  - terminaison par une ligne avec seulement ".."



## Evolution du format des entêtes

Caractères non ASCII dans les entêtes :

=?charset?encode?encoded-text?=

- **charset** : us-ascii, iso-8859-x, ...
- **encode** : le texte encodé doit rester en ASCII NVT
  - **Qoted-printable** (Q) pour les jeux de caractères latins :
    - caractères > 128 => encodé sur 3 caractères (= et code hexa.)
    - caractère espace => toujours =20
  - **Base64** (B) :
    - trois octets de texte (24 bits) => encodée sur 4 car. ASCII
    - valeur sur 6 bits (0, 1, 2... 63) => ABC...YZab...yz01...9+/-
    - bourrage avec "==" si non aligné sur 4 caractères.
- **encoded-text** :  
=?iso-8859-2?Q?Igen,=20k=F6sz=F6n=F6m?=
- =?iso-8859-1?B?QnJhdm8sIHZvdXMgYXZleiBy6XVzc2kgIQo=?



## MIME (*Multipurpose Internet Mail Extensions*)

Nouveaux entêtes MIME (RFC 2045 et RFC 2046)

- **Mime-Version: 1.0**
- **Content-Type: type/sous-type; paramètres**
  - **simples** : text/plain; charset="ISO-8859-1"
    - text/html, image/jpeg...
  - **structurés** : multipart/mixed; Boundary=hjfdskjhfdshf
    - chaque bloc du message débute par : hjfdskjhfdshf
    - imbrication possible
- **Content-Disposition: présentation du bloc (RFC 2183)**
- **Content-Transfer-Encoding: encodage du bloc**
  - 7 bits compatible avec les anciens MTA RFC 821
    - 7bit (ASCII NVT)
    - quoted-printable (recommandé pour tout texte)
    - base64 (recommandé pour les flux d'octets)
  - 8 bits si la commande 8BITMIME est acceptée
    - 8bit et Binary (lignes ou bloc de données sur 8 bits)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## ESMTP : exemple de message MIME

```
From: Olivier Fourmaux <olivier.fourmaux@lip6.fr>
Date: Wed, 20 Feb 2002 01:21:01 +0100
To: Toto <toto@free.fr>
Subject: Document no 3.02
Mime-Version: 1.0
Content-Type: multipart/mixed; boundary="/9DWx/yDrRhgMJTb"
Content-Disposition: inline
Content-Transfer-Encoding: 8bit
User-Agent: Mutt/1.2.5i

--/9DWx/yDrRhgMJTb
Content-Type: text/plain; charset=iso-8859-1
Content-Disposition: inline
Content-Transfer-Encoding: 8bit

Voici le document secret que vous m'avez demandé...

--/9DWx/yDrRhgMJTb
Content-Type: application/pdf
Content-Disposition: attachment; filename="sujet-exam-RES.pdf"
Content-Transfer-Encoding: base64
```

JVBERi0xLjIKJcfsj6IKNSAwIG9iago8PC9MZW5ndGgNiAwIFIvRmlsdGVyIC9GbGF0ZUR1
Y29kZT4+CnN0cmBhQp4n01dy7YdtRGd3684Mx6L07T63ZkBdhgXvYlJF1MHNsYm+sHhkCS...



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## MIME : types et sous-types

/etc/mime.types

audio/midi	multipart/mixed
audio/mpeg	multipart/parallel
audio/x-wav	multipart/signed
application/mac-binhex40	
application/msword	
application/octet-stream	text/html
application/postscript	text/plain
application/vnd.hp-PCL	text/richtext
application/vnd.ms-excel	text/rtf
application/x-debian-package	text/xml
application/x-doom	text/x-java
application/x-gnumeric	text/x-tex
application/x-java-applet	text/x-vcard
application/x-javascript	
application/x-msdos-program	video/mpeg
application/x-tar	video/quicktime
audio/basic	video/x-msvideo



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

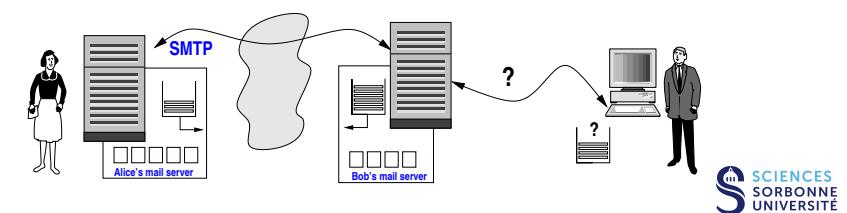
## Remise finale des messages

Machine accédant sporadiquement au réseau ?

⇒ Messages stockés sur le dernier MTA (celui de l'ISP par exemple)

- plusieurs alternatives combinables :

- accès direct au serveur (montage NFS ou SMB)
- POP
- IMAP
- HTTP



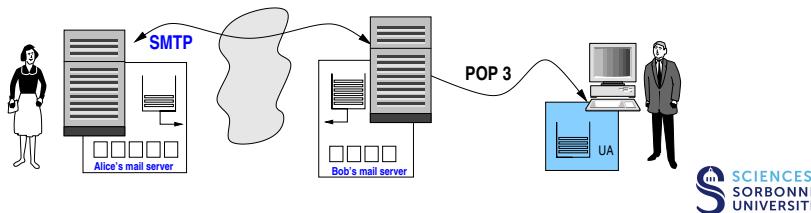
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## POP3

*Post Office Protocol – Version 3 (RFC 1939)*

- simple
- connexion TCP sur le port 110
- trois phases :
  - autorisation (identification)
  - transaction (réception et marquage)
  - mise à jour (suppression effective du serveur)



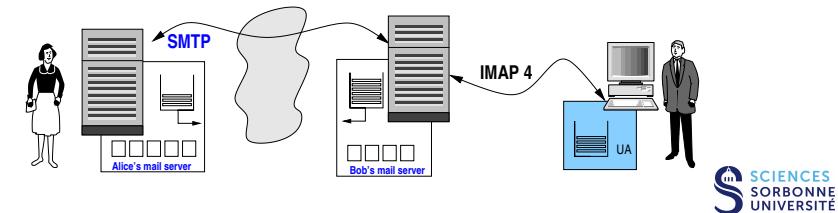
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## IMAP4

*Internet Mail Access Protocol – version 4rev1 (RFC 3501)*

- complexe
- connexion TCP sur le port 143
- même fonctionnalité que POP avec :
  - accès par attribut (12<sup>eme</sup> e-mail d'Alice)
  - récupération de partie de message (3<sup>eme</sup> pièce jointe)
  - **synchronisation** de boîtes aux lettres



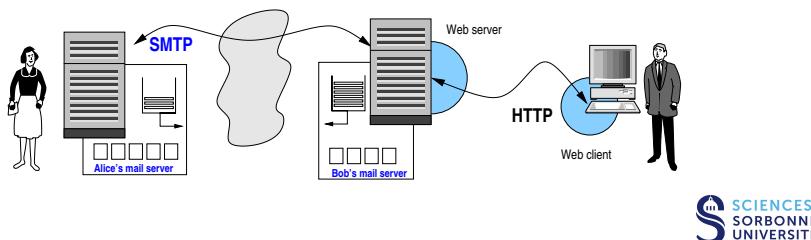
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Web-mail

UA sur le serveur SMTP et interface Web

- comptes web spécifiques :
  - Hotmail, Yahoo!, Gmail...
- autre moyen d'accès au serveur d'entreprise ou de l'ISP :
  - horde/IMP, Squirrelmail, Zimbra...



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Messagerie et sécurité

Les protocoles de base ne sont pas sécurisés

- échange textuel non confidentiel (contrôle et données)
- aucune authentification avec SMTP
- identifiant et mot de passe en clair avec POP et IMAP

Quelques solutions :

- PGP (*Pretty good privacy*) en environnement hostile :
  - authentification, intégrité et confidentialité (données signées et/ou cryptées)
- OpenPGP (RFC 4880) : GPG (*Gnu Privacy Guard*)
- si confiance dans le site distant, sécurisation des connexions :
  - si le site distant est accessible via SSH
    - accès à distance sur le serveur via SSH (textuels), **tunnels SSH**
- si clients et serveurs avec TLS
  - SMTP+STARTTLS (RFC 3207), POP3S and IMAPS (RFC 2595)
  - HTTPS pour sécuriser le Web-Mail...

Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

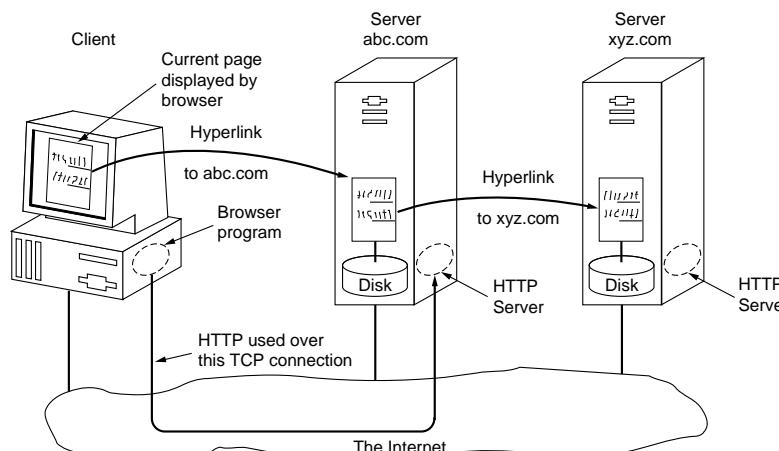
- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## HTTP : Principe



## World Wide Web

→ 90' : Internet = réseau seulement académique

90' → : World Wide Web

- système d'accès aux données convivial et intuitif (graphique)
  - développé au CERN par Tim Berners-Lee à partir de 1990
  - client (*browser*) :
    - NCSA Mosaic en 1993 (U. of Illinois Urbana-Champaign)
      - le WWW ne compte que 200 sites
      - première intégration d'images
      - gain de popularité exponentiel !
    - Netscape Navigator en 1994 (⇒ Mozilla Firefox en 1998)
    - Microsoft Internet Explorer en 1995 (début de la *browser wars*)
    - et beaucoup d'autres (voir le site du W3C)
  - serveur (*web server*) :
    - NCSA httpd Web Server (⇒ Apache en 1998)
    - Microsoft IIS (*Internet Information Service*) en 1995
- ⇒ un protocole : **HTTP**



## HTTP : Terminologie

- une **page web** ou un **document** est composé d'objets
  - fichiers texte au format HTML
  - images GIF, JPEG...
  - applets JAVA
  - ...
- un document consiste généralement en un **fichier HTML de base** avec des références vers d'autres objets désignés par des URL
  - **HTML (HyperText Markup Language)** est un langage à balises pour la description de documents contenant des hyper-liens identifiés par des URL
  - une **URL (Uniform Resource Locator)** indique un protocole pour récupérer sur une machine un objets à travers le réseau
    - `http://www.lip6.fr/info/linux.html`
    - `ftp://ftp.lip6.fr/pub/linux/distrib/debian/ls-1R.txt`
    - `file:/public/image/penguin.jpeg`
    - `mailto:olivier.fourmaux@upmc.fr`



## HTTP : Protocole

### HyperText Transfer Protocol

- connexion TCP sur le port 80
- échanges définis :
  - les **requêtes** de demande d'objets (client ⇒ serveur)
  - les **réponses** contenant les objets demandés (serveur ⇒ client)
  - pas d'état dans le serveur (**stateless protocol**)
- versions HTTP :
  - → 1997 **HTTP/1.0** (RFC 1945)
    - connexions **non persistantes**, une connexion créée par objet, charge et latence importantes (TCP 3-way handshake et *slowstart*)
  - 1998 → **HTTP/1.1 (RFC 7230 à 7237, ex-RFC 2616)**
    - compatibilité ascendante, connexions **persistantes**, possibilité de requêtes multiples (**pipelining**), transfer par blocs...
  - 2015 → **HTTP/2 (RFC 7540)**



## HTTP : Format requête

### Exemple :

```
GET /index.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4 [en] (X11;...)
Host: calvin.lip6.fr
Accept: image/gif, image/jpeg, */
Accept-Encoding: gzip
Accept-Language: fr-FR, fr, en
Accept-Charset: iso-8859-1,*,utf-8
```

#### Format général d'un message :

##### - Request line -

Method	sp	URL	sp	Version	cr	lf
Header field name	:	Value	cr	If		
Header field name	:	Value	cr	If		

...

Header lines

...

Header field name	:	Value	cr	lf
cr	If			

Entity body

### Method

- GET
- POST (formulaires)
- HEAD (test de pages)...

### Connection

- Close
- Keep-Alive



## HTTP : Exemple

### Browser :

```
GET /index.html HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/4 [en] (X11; I; Linux 0.99 i486)
Host: calvin.lip6.fr
Accept: image/gif, image/jpeg, image/png, */
Accept-Encoding: gzip
Accept-Language: fr-FR, fr, en
Accept-Charset: iso-8859-1,*,utf-8
```

### Web server :

```
HTTP/1.1 200 OK
Date: Tue, 24 Sep 2002 12:59:28 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Last-Modified: Sat, 29 Apr 2000 07:07:45 GMT
ETag: "1382c-ffe-390a8a41"
Accept-Ranges: bytes
Content-Length: 4094
Keep-Alive: timeout=15, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text...
<META NAME="GENERATOR" CONTENT="Mozilla/4.05...
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)
```

```
<META NAME="Author" CONTENT="johnie@debian.o...
<META NAME="Description" CONTENT="The initia...
<TITLE>Welcome to Your New Home Page!</TITLE>
</HEAD>
<BODY TEXT="#000000" BGCOLOR="#FFFFFF" LINK="#0...
<BR>
<H1>Welcome to Your New Home in Cyberspace!</H1>
<BR>
<IMG SRC="icons/openlogo-25.jpg" ALT="Debian">
<IMG SRC="icons/apache_pb.gif" ALT="Apache"></P>
```

```
<P>This is a placeholder page installed by the
<A HREF="http://www.debian.org/">Debian</A>
release of the
<A HREF="http://www.apache.org/">Apache</A> Web
server package, because no home page was installed
on this host. You may want to replace this as soon
as possible with your own web pages, of course....
```

```
<BLOCKQUOTE>
This computer has installed the Debian GNU/Linux
operating system but has nothing to do with the
Debian GNU/Linux project. If you want to report
something about this hosts behaviour or domain,
please contact the ISPs involved directly,
<strong>not</strong> the Debian Project.
SCIENCE
SORBONNE
UNIVERSITE
</BLOCKQUOTE>
.....
</HTML>
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)
```

## HTTP : Format réponse

### Exemple :

```
HTTP/1.1 200 OK
Date: Tue, 24 Sep 2002 12:59:28 GMT
Server: Apache/1.3.9 (Unix) Debian/GNU
Last-Modified: Sat, 29 Apr 2000 07:07:45 GMT
Content-Length: 4094
...
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="tex...
<META NAME="GENERATOR" CONTENT="Mozilla/4.05...
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)
```

#### Format général d'un message :

##### - Status line -

Version	sp	Status code	sp	Phrase	cr	lf
Header field name	:	Value	cr	If		
Header field name	:	Value	cr	If		

...

Header lines

Header field name	:	Value	cr	lf
cr	If			

Entity body

### status + description :

- 200 OK
- 301 Move permanently
- 400 Bad Request
- 404 Not Found
- 505 HTTP Version



Supported

## HTTP : authentification les utilisateurs

### Authentification (RFC 7235, ex-RFC 2617)

- 5 méthodes : simple (*Basic*), par signature MD5 (*Digest*)...
- exemple : requête du client sur une page avec procédure d'authentification basique
  - réponse du serveur avec page vide et entête :
    - 401 Authorization Required
    - WWW-Authenticate: *détails\_méthode\_d'autorisation*
  - requête du client sur la même page avec entête :
    - Authorization: *nom\_utilisateur mot\_de\_passe*
  - réponse du serveur :
    - si Ok ➡ la page demandée
    - sinon 401 Authorization Required...



toutes les méthodes du RFC sont limitées ➡ utilisation de mécanismes non HTTP

## HTTP : GET conditionnel

### 1<sup>re</sup> requête HTTP :

```
GET /carte/france.jpg HTTP/1.1
Host: www.atlas.org
```

### 2<sup>me</sup> requête HTTP :

```
GET /carte/france.jpg HTTP/1.1
Host: www.atlas.org
If-modified-since: Sat, 29 Apr 2005 ...
```

### 1<sup>re</sup> réponse HTTP :

```
HTTP/1.1 200 OK
Date: Mon, 2 Oct 2005 23:56:18
Server: Apache/1.3.9 (Unix)
Last-Modified: Sat, 29 Apr 2005 ...
Content-Type: image/jpeg
```

Données.....  
.....  
.....

### 2<sup>me</sup> réponse HTTP :

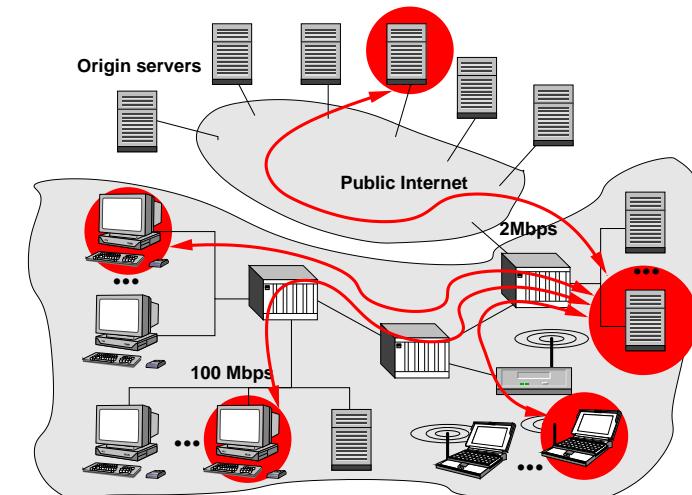
```
HTTP/1.1 304 Not Modified
Date: Mon, 3 Oct 2005 00:06:43
Server: Apache/1.3.9 (Unix) Debian/GNU
```

## HTTP : mécanisme de gestion d'état

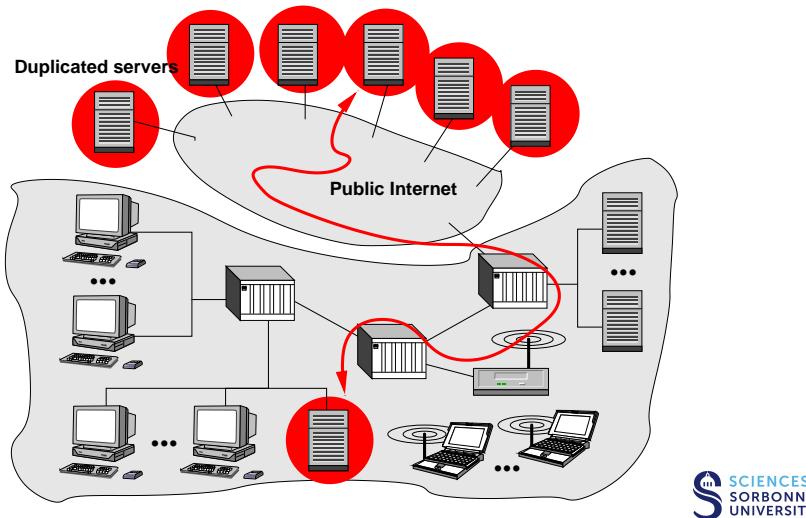
### Cookies (RFC 6265, ex-RFC 2109/2965)

- identifiant associé à un utilisateur sur sa machine :
- le serveur indique un *cookie* avec l'entête :
  - Set-cookie: *nombre\_identifiant*
- le cookie est stocké chez le client qui, lorsqu'il demandera la même page sur le même serveur, l'intégrera grâce à l'entête :
  - Cookie: *nombre\_identifiant*
- de nombreux attributs peuvent être associés au *cookie* pour en limiter la portée

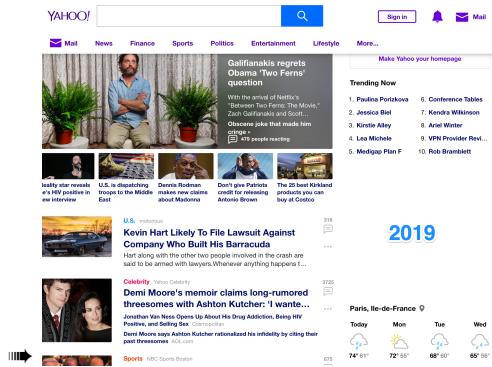
## HTTP : Cache et proxy



## HTTP : CDN



## HTTP : évolution du contenu



## HTTP : évolution

### HTTP/2 (RFC 7540)

- problèmes avec HTTP/1.1 :
  - standard trop complexe
  - pipelining non utilisé
  - trop de connexions TCP en parallèle
  - transfert initiés par le client
  - encodage des entêtes inefficace et non compressé
- évolution avec HTTP/2 :
  - multiplexage de plusieurs transferts dans une connexion TCP
  - évitement du blocage en tête de file (HOL)
  - envois potentiellement initiés par le serveur (PUSH)
  - protocole binaire

efficacité réelle ?

## HTTP/2 : Exemple

```
User-Agent: curl/7.31.2
Host: test-http2.lip6.fr
Accept: */*
Connection: Upgrade, HTTP2-Settings
Upgrade: h2c
HTTP2-Settings: AAMAAABkAAQAAP__
```

```
HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: h2c
.....d.....
```

```
PRI * HTTP/2.0
```

```
SM
```

```
.....z.%P.....f.i.....a.Y>..a.j..m@...jb..I|...M..
.q.;l.z....a.j..m@q..%6!h....[h.Yy^I..m.....r.(a....!
.....j
.!>.*')SR.J.v..i....T.5.....|.
.R..Jk.....
<!DOCTYPE html>
<head>
<meta charset="utf-8">
...
```

## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## RTP

*Real-Time Transport Protocol (RFC 3550 pour la v2)*

- distribution de données multimédia de bout-en-bout
- gère de nombreux paramètres nécessaires pour les applications
  - identification (type de données, participants)
  - numérotation séquentielle et temporelle des paquets
  - synchronisation de flux
  - surveillance de qualité
  - communication vers un ou plusieurs destinataires



les algorithmes de codage, de synchronisation, de lecture sont implantés au niveau de l'application



## Applications multimédia

De nombreux mécanismes sont associés à ces applications (ex : pour une vidéo-conférence) :

- établissement d'une session
  - préparation (médias, codecs, protocoles...)
  - annonce et invitation (news, web, e-mail...)
  - initiation (signalisation ou connexion)
- participation
  - **émission** : codage/compression, "paquetisation", transmission...
  - **réception** : décodage/décompression, lecture audio/vidéo...

► Besoin de protocoles spécifiques pour :

- encapsuler les données multimédia
- gérer les participants
- ...



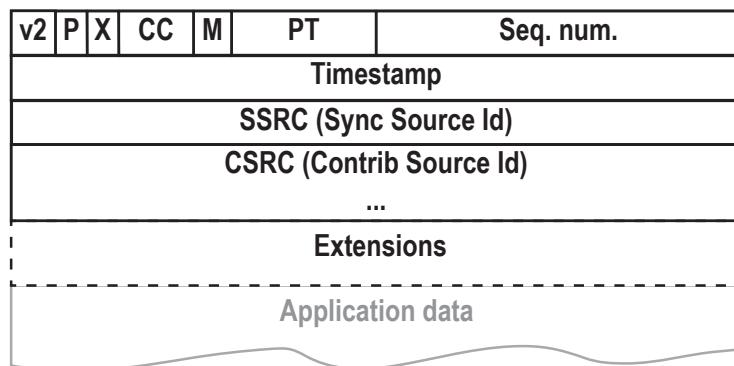
## Transport et RTP

*Couche transport associée à RTP/RTCP*

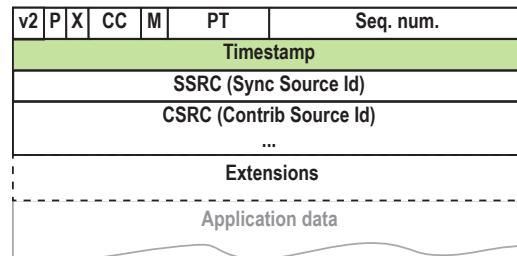
- RTP est-il un protocole de transport ?
  - pas de mux/démux
- doit être associé à un protocole de transport classique
  - généralement UDP
  - TCP, DCCP, SCTP...
- configuration classique avec UDP :
  - numéro de port pair pour RTP
  - numéro de port impair suivant pour RTCP
  - les numéros de port RTP et RTCP doivent être différents



## Format d'un message RTP



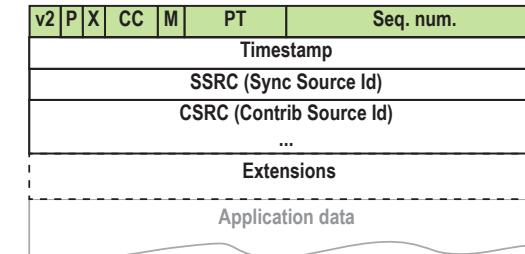
## En-tête RTP : estampille temporelle



- instant d'échantillonnage du premier octet du paquet RTP
  - lié à une horloge qui s'incrémente de façon monotone et linéaire
  - fréquence de l'horloge dépendant de l'application (spécifiée par le profil)
- instants de présentation
  - calculés à partir d'une origine temporelle, dépend du profil
- valeur initiale aléatoire



## En-tête RTP



- P=1 si bourrage (padding), X=1 si extension présente
- CC : nombre d'identifiants CSRC qui suivent l'entête
- M : marqueur (spécifique à un profil)
- PT : type de paquet RTP (spécifique à un profil)
- Seq. num. : incrémenté de 1 pour chaque paquet RTP (valeur initiale aléatoire)



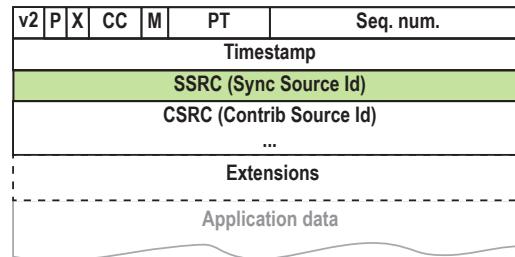
## En-tête RTP : estampille temporelle (suite)

L'estampille temporelle permet la lecture au bon rythme des données multimédia :

- résolution suffisante** pour permettre le calcul de la gigue
  - (unité = une trame vidéo) est insuffisant
  - si les messages sont générés régulièrement (ex : 160 échantillons audio), alors l'horloge peut être celle d'échantillonnage (et s'incrémenter de 160 à chaque message)
- horloge identique** pour tous les messages générés au même moment
  - ex : tous les messages associés à une même trame vidéo auront la même estampille temporelle (mais pas le même numéro de séquence)
- selon le codage la croissance des estampille peut être non monotone
  - ex : messages transportants des trames MPEG interpolées (IPBBBPPBB...)

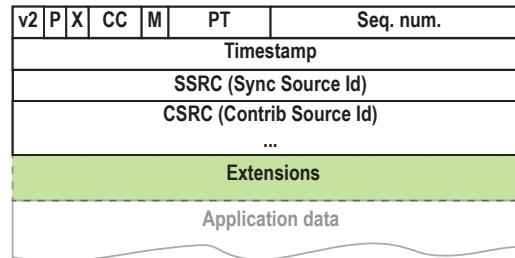


## En-tête RTP : SSRC



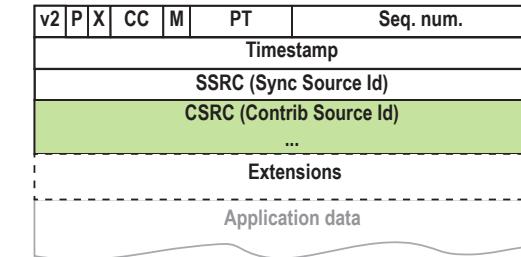
- identifie la source sur laquelle les données du paquet sont synchronisées
- à chaque SSRC correspond un interval de numéro de séquence
- choisi de manière aléatoire pour ne pas avoir 2 SSRC identiques dans la même session RTP (les implémentations de RTP doivent pouvoir gérer les collisions)
- chaque application peut avoir plusieurs SSRC

## En-tête RTP : extensions



- permet de réaliser une implementation propriétaire
  - dans un cadre expérimental par exemple (pour de tester de nouveaux formats de données)
  - paramètres d'extension définis par l'implementation

## En-tête RTP : CSRC



- liste de 0 à 15 items de 32 bits
- identifie les sources qui ont des données dans le paquet RTP.
- les CSRC sont insérés par les "mixers"
- plusieurs utilisations possibles, exemple :
  - identifier les interlocuteurs dans une session de télé-conférence (le mixer indique son SSRC et les SSRC des sources initiales qui deviendront des CSRC)

## Profiles RTP audio et vidéo de base

Le RFC 3551 définit un profil de base sans négociation de paramètres

PT	encodage	A/V	Clock (Hz)
0	PCMU	A	8000
2	G721	A	8000
3	GSM	A	8000
5	DVI4	A	8000
7	LPC	A	8000
8	PCMA	A	8000
9	G722	A	8000
15	G728	A	8000
26	JPEG	V	90000
31	H261	V	90000...

## Règles d'encodage RTP

#### Règles génériques d'encodage RTP :

- diffusion contrôlée au niveau de la source
    - choix d'émission avec différent PT
    - segmentation des données au niveau du codeur
    - utilisation du bit M si besoin
  - **multiplexage** de plusieurs types de données
    - interdit dans une session RTP
    - utiliser plusieurs sessions en parallèle
  - mécanismes audio et vidéo très différents...



## Règles d'encodage audio RTP (2)

**Sessions multi-canaux** : les échantillons d'un même instant doivent être dans le même message RTP en suivant cet ordre

number	channel							
	1	2	3	4	5	6		
stereo	l	r					r	right
3	l	r	c				c	center
4	l	c	r	S			S	surround
5	F1	Fr	Fc	S1	Sr		F	front
6	l	lc	c	r	rc	S		



## Règles d'encodage audio RTP (1)

Règles de base pour l'encodage audio avec RTP :

- suppression des silence (grâce aux estampilles temporelles)
    - mais ajout possible d'un descripteur de silence (*Confort Noise*)
    - positionnement du marqueur au premier message audio après un silence (pour distinguer le silence d'un problème réseau)
  - horloge RTP indépendante du nombre de canaux utilisés et de l'encodage
    - si N canaux alors N échantillons pendant une période
  - fréquences d'échantillonage initiales
    - 8000, 11025, 16000, 22050, 24000, 32000, 44100 et 48000 Hz



## Recommandation de fonctionnement pour l'audio RTP

Taille des données audio transportées :

- données audio dans un message : **20 ms ou une trame** encodée
    - l'interval de temps dans un message ➡ délai de bout-en-bout mini.
    - interval plus long ➡ augmentation du délai et du taux de perte
    - valeurs acceptables : 0 à 200 ms
  - encodages basés sur des **échantillons**
    - chaque échantillon codée sur S bits fixes
    - les bits des échantillons sont écrits à la suite
      - si multicanal, les échantillons du même instants sont écrits à la suite dans l'ordre définis
    - durée d'un messages dépend du nombre N d'échantillons
    - les données ( $S \times N$  bits) doivent être un multiple de l'octet
  - encodages basés sur des **trames**
    - encode un intervalle audio de taille fixe prédéfinie
    - codage résultant de taille variable ou fixe
      - plusieurs trames peuvent être intégrées dans un message (nécéssité de pouvoir les séparer)
    - en multicanal, canaux écrit à la suite dans l'ordre défini



## Exemple audio échantillonnée : L16

### Codage PCM 16 bits

- données = suite de mots de 16 bits
  - entiers signés (en complément à 2)
  - transmis selon l'ordre des octets du réseau (*MSB first*)
- information complémentaires transmises hors-bande (SDP...)
  - fréquence d'échantillonnage
  - nombre de canaux (et ordre si besoin)
  - traitement analogique amont (amplification etc.)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Autres formats d'encapsulation RTP audio

RFC 2658 RTP PureVoice(tm) Audio

RFC 3389 RTP Comfort Noise (CN)

RFC 3558 RTP for Enhanced Variable Rate Codecs (EVRC)

RFC 4184 RTP for AC-3 Audio

RFC 4348 RTP for VMR-WB Audio Codec

RFC 4352 RTP for AMR-WB+ Audio Codec

RFC 4867 RTP for AMR (Adaptive Multi-Rate) and AMR-WB Audio Codecs

RFC 5188 RTP for the Enhanced Variable Rate Wideband Codec (EVRC-WB)

RFC 5215 RTP for Vorbis Encoded Audio

RFC 5219 A More Loss-Tolerant RTP Payload Format for MP3 Audio

RFC 5391 RTP for ITU-T Recommendation G.711.1

RFC 5404 RTP for G.719

RFC 5574 RTP for the Speex Codec

RFC 5577 RTP for ITU-T Recommendation G.722.1 (wideband)

RFC 5584 RTP for the Adaptive TRansform Acoustic Coding (ATRAC) Family

RFC 5993 RTP for GSM Communications Half Rate (GSM-HR)

RFC 6884 RTP for the Enh. Var. Rate Narrowband-Wideband Codec (EVRC-NW)

RFC 7587 RTP for the Opus Speech and Audio Codec



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Exemple audio avec trame : GSM

Codage GSM : blocs de 160 échantillons (20 ms) codées en 33 octets

Octet	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
0	1	1	0	1	LArC0.0	LArC0.1	LArC0.2	LArC0.3
1	LArC0.4	LArC0.5	LArC1.0	LArC1.1	LArC1.2	LArC1.3	LArC1.4	LArC1.5
2	LArC2.0	LArC2.1	LArC2.2	LArC2.3	LArC2.4	LArC3.0	LArC3.1	LArC3.2
3	LArC3.3	LArC3.4	LArC4.0	LArC4.1	LArC4.2	LArC4.3	LArC5.0	LArC5.1
4	LArC5.2	LArC5.3	LArC6.0	LArC6.1	LArC6.2	LArC7.0	LArC7.1	LArC7.2
5	Nc0.0	Nc0.1	Nc0.2	Nc0.3	Nc0.4	Nc0.5	Nc0.6	bc0.0
6	bc0.1	Mc0.0	Mc0.1	xmaxc00	xmaxc01	xmaxc02	xmaxc03	xmaxc04
7	xmaxc05	xmc0.0	xmc0.1	xmc0.2	xmc1.0	xmc1.1	xmc1.2	xmc2.0
8	xmc2.1	xmc2.2	xmc3.0	xmc3.1	xmc3.2	xmc4.0	xmc4.1	xmc4.2
9	xmc5.0	xmc5.1	xmc5.2	xmc6.0	xmc6.1	xmc6.2	xmc7.0	xmc7.1
10	xmc7.2	xmc8.0	xmc8.1	xmc8.2	xmc9.0	xmc9.1	xmc9.2	xmc10.0
11	xmc10.1	xmc10.2	xmc11.0	xmc11.1	xmc11.2	xmc12.0	xmc12.1	xmc12.2
12	Nc1.2	Nc1.1	Nc1.2	Nc1.3	Nc1.4	Nc1.5	Nc1.6	bc1.0
13	bc1.1	Mc1.0	Mc1.1	xmaxc10	xmaxc11	xmaxc12	xmaxc13	xmaxc14
14	xmaxc15	xmc13.0	xmc13.1	xmc13.2	xmc14.0	xmc14.1	xmc14.2	xmc15.0
15	xmc15.1	xmc15.2	xmc16.0	xmc16.1	xmc16.2	xmc17.0	xmc17.1	xmc17.2
16	xmc18.0	xmc18.1	xmc18.2	xmc19.0	xmc19.1	xmc19.2	xmc20.0	xmc20.1
17	xmc20.2	xmc21.0	xmc21.1	xmc21.2	xmc22.0	xmc22.1	xmc22.2	xmc23.0
18	xmc23.1	xmc23.2	xmc24.0	xmc24.1	xmc24.2	xmc25.0	xmc25.1	xmc25.2
19	Nc2.0	Nc2.1	Nc2.2	Nc2.3	Nc2.4	Nc2.5	Nc2.6	bc2.0
20	bc2.1	Mc2.0	Mc2.1	xmaxc20	xmaxc21	xmaxc22	xmaxc23	xmaxc24
21	xmaxc25	xmc26.0	xmc26.1	xmc26.2	xmc27.0	xmc27.1	xmc27.2	xmc28.0
22	xmc28.1	xmc28.2	xmc29.0	xmc29.1	xmc29.2	xmc30.0	xmc30.1	xmc30.2
23	xmc31.0	xmc31.1	xmc31.2	xmc32.0	xmc32.1	xmc32.2	xmc32.3	xmc32.4
24	xmc33.0	xmc33.1	xmc33.2	xmc34.0	xmc34.1	xmc34.2	xmc34.3	xmc34.4
25	xmc36.1	xmc36.2	xmc37.0	xmc37.1	xmc37.2	xmc38.0	xmc38.1	xmc38.2
26	Nc3.0	Nc3.1	Nc3.2	Nc3.3	Nc3.4	Nc3.5	Nc3.6	bc3.0
27	bc3.1	Mc3.0	Mc3.1	xmaxc30	xmaxc31	xmaxc32	xmaxc33	xmaxc34
28	xmaxc35	xmc39.0	xmc39.1	xmc39.2	xmc40.0	xmc40.1	xmc40.2	xmc41.0



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Règles d'encodage vidéo RTP

Règles de base pour l'encodage vidéo avec RTP :

- la plupart des encodages vidéo utilisent une horloge à 90000 Hz
  - identique à celle de MPEG
  - multiple entier de 24 (HDTV), 25 (PAL/SECAM), 29.97 (NTSC) et 30 Hz (HDTV)
  - suffisamment élevée pour calculer la gigue
- estampille temporelle ➔ instant d'échantillonnage de l'image
  - si image sur plusieurs messages : même estampille
- le bit de marquage est utilisé sur le dernier message d'une image



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Exemple vidéo : H264

H264 spécifie une couche d'abstraction du réseau  
(NAL : Network Abstraction Layer)

- paquetisation contrôlée au niveau du codeur
  - découpage des images en tranches (*slices*) autonomes (au niveau du décodage image)
  - puis partition en unités NAL (NALU)
- NALU autonomes (au niveau du décodage transport)
  - entête NALU (1 octet)
  - données NALU (flux d'octets bruts : image ou paramètres H264)
- intégration dans RTP
  - habituellement une NALU  $\sim 1.4$  Ko
  - plusieurs NALU par messages
  - grandes NALU sur plusieurs messages



## RTCP

Real-Time Transport Control Protocol (RFC 3550)

- totalement intégré à RTP
  - même RFC 3550
- transmet les informations de session
  - synchronisation
  - participants
  - ...
- fournit des statistiques sur la qualité de la session
- transmet des informations de contrôle sur la session
  - ex : identifier un participant sur les écrans des participants



## Autres formats d'encapsulation RTP vidéo

- RFC 2435 RTP for JPEG-compressed Video
- RFC 2250 RTP for MPEG1/MPEG2
- RFC 3640 RTP for Transport of MPEG-4 Elementary Streams
- RFC 4425 RTP for Video Codec 1 (VC-1)
- RFC 4587 RTP for H.261 Video Streams
- RFC 4628 RTP for H.263 Video Streams
- RFC 4629 RTP for the 1998 Version of ITU-T Rec. H.263 Video (H.263+)
- RFC 5371 RTP for JPEG 2000 Video Streams
- RFC 6184 RTP for H.264 Video
- RFC 6190 RTP for Scalable Video Coding (extension H264 AVC)
- RFC 6416 RTP for MPEG-4 Audio/Visual Streams
- RFC 6469 RTP for DV (IEC 61834) Video
- RFC 7741 RTP for VP8 Video
- RFC 7798 RTP for High Efficiency Video Coding (HEVC)



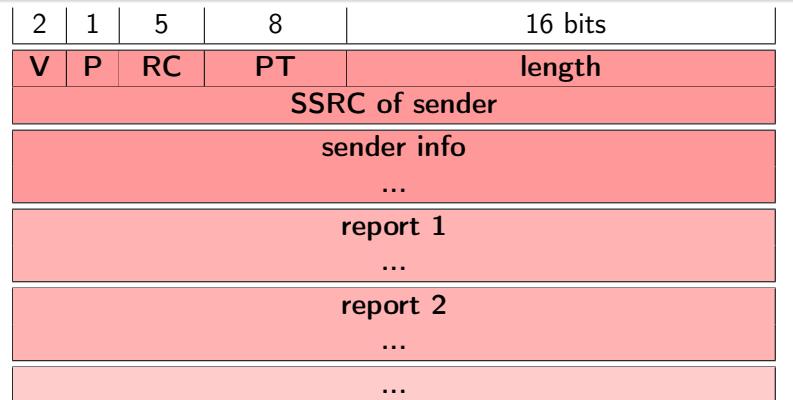
## Format des messages RTCP

2	1	5	8	16 bits
V	P	xC	PT	length

- **V** = version (2 actuellement)
- **P** = bourrage (*padding*) inclus dans la longueur
  - taille = valeur du dernier octet de bourrage (crypto par bloc)
- **xC** = nombre d'éléments dans ce message RTCP
- **PT** = type de message :
  - 200 = SR (*Sender Report*) : informations par les émetteurs
  - 201 = RR (*Receiver Report*) : statistiques par les récepteurs
  - 202 = SDES (*Source Description*) : information sur la source en UTF8 (CNAME, e-mail, numéro de téléphone, localisation...)
  - 203 = BYE : quitter une session RTP
  - 204 = APP (*APPlication*) : fonctions spécifiques de l'application
- **length** = nombre de mots de 32 bits (entête et bourrage inclus)
- délimitation s'il y a plusieurs messages RTP dans l'entité de



## Format des messages SR



- RC = nombre de **report** inclus dans les message RTCP
- **SSRC** (*Synchronization SouRCe*) = identifiant de l'origine du msg
- 1 \* **sender info** spécifique aux messages SR
- RC \* report



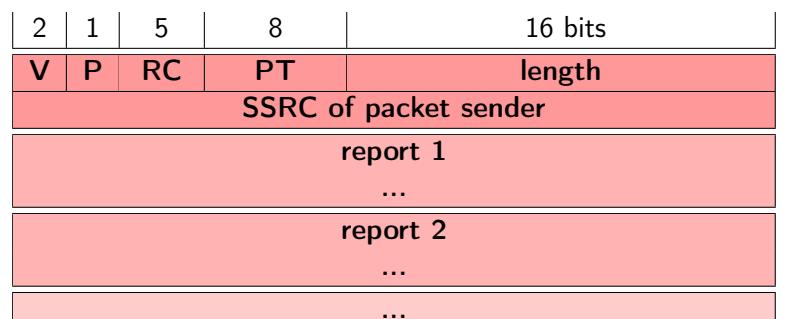
## Format des messages SR : sender info

2	1	5	8	16 bits
				NTP timestamp, most significant word
				NTP timestamp, least significant word
				RTP timestamp
				sender's packet count
				sender's octet count

- **NTP** (*Network Time Protocol*) = temps absolu (*wallclock time*)
  - estampille sur 64 bits (32 bits → secondes + 32 bits → fractions de s.)
  - relatif au 1 janvier 1900 à 0h00 (Fin en 2036 !)
- **length** = nombre de mots de 32 bits (entête et bourrage inclus)
- **RTP timestamp** = correspondance RTP du temps NTP pour cette session RTP (avec l'unité de temps spécifique)
- **sender's packet count** = nombre de messages RTP pour la session
- **sender's octet count** = idem pour qtt de données utiles



## Format des messages RR



- RC = nombre de **report** inclus dans les message RTCP
- **SSRC** (*Synchronization SouRCe*) = identifiant de l'origine du msg
- RC \* report



## Format des messages SR et RR : report

2	1	5	8	16 bits
				SSRC N (SSRC of source N)
				fraction lost   cumulative number of packets lost
				extended highest sequence number received
				interarrival jitter
				last SR (LSR)
				delay since last SR (DLSR)

- **SSRC N** = identifiant de la source auquel se rapporte ce report
- **fraction lost** = perte depuis le dernier SR/RR
- **cumul. num. of pkt lost** = perte depuis le début de la session
- **ext. hi. seq. num. received** = num. séq. RTP étendu du nombre de cycle
- **interarrival jitter** = écart moyen lissé (temps NTP sur 32 bits)



$$D(i,j) = (R_j - R_i) / (S_j - S_i)$$

LSD = longueur d'émission de données SR (32 bits)

## Calcul temporels RTCP

Manipulation du temps NTP sur 64 et 32 bits :

- **T1 = 10 novembre 1995 à 11h33 et 25,125 secondes**
  - **T1 sur 64 bits (NTP timestamp)**
    - secondes NTP : 0x b44d b705
    - fraction NTP : 0x 2000 0000
  - **T1 sur 32 bits** : 0x b705 2000
- **T2 = 10 novembre 1995 à 11h33 et 36,500 secondes**
  - **T2 sur 64 bits (NTP timestamp)**
    - secondes NTP : 0x b44d b710
    - fraction NTP : 0x 8000 0000
  - **T2 sur 32 bits** : 0x b710 8000
- si SR envoyé à T1, RR reçut à T2 avec :
  - **LSR** = 0x b705 2000
  - **DLSR** = 0x 0005 4000

➡ RTT ?

## RTSP



## RTP/RTCP

Comment partager la bande passante entre RTP et RTSP ?

- règle : limiter le trafic de contrôle à moins de 5%

Comment limiter le trafic RTCP ?

- unicast : calcul direct
- multicast : plus il y a de participants, plus il faut limiter les envois de messages RTCP

Contrôle de débit RTCP :

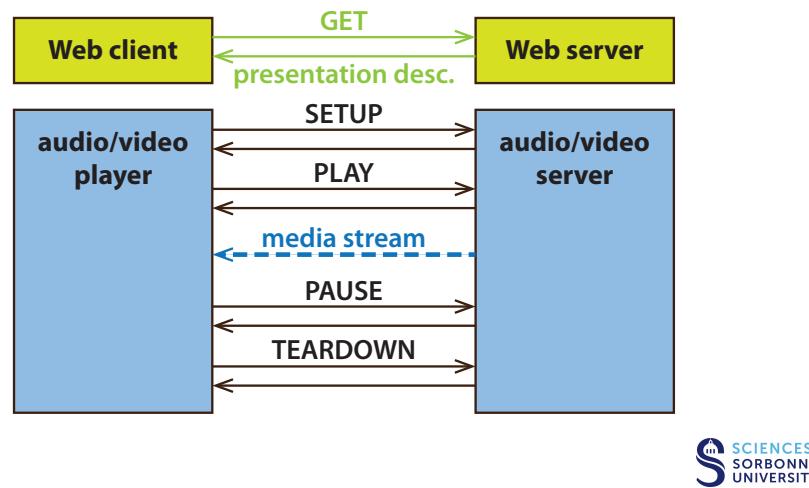
- algorithme basé sur l'estimation distribuée du nombre de participants  $N(t)$  (en comptant les SSRC des reports)
- $D_{RTCP} \leq \frac{0.05D_{RTP}}{N(t)T_{mesgRTCP}}$
- ajout d'une composante aléatoire pour éviter les synchronisations

## RTSP

*Real Time Streaming Protocol (RFC 2326)*

- contrôle hors bande de la diffusion (habituellement TCP port 554)
- identification de la ressource via URL (ex : rtsp://media.upmc.fr:554/videofile)
- fonctionnalités typiques d'un lecteur vidéo :
  - lecture/pause
  - avance rapide
  - accès à une position temporelle...
- ne définit aucun mécanismes de codage pour la vidéo/audio
- ne définit pas la méthode d'encapsulation des données
- n'impose aucun mode de mise en mémoire tampon du lecteur média

## RTSP : fonctionnement



## RTSP : réponses

La première ligne d'une réponse est un *status code* suivi d'un texte :

- 1xx (*Informational*) : requête reçue, processus en cours
  - 100 Continue
- 2xx (*Success*) : action comprise et acceptée
  - 200 OK
  - 201 Created
- 3xx (*Redirection*) : d'autres actions doivent être réalisées
  - 300 Multiple Choices
  - 302 Moved Temporarily
  - 305 Use Proxy
- 4xx (*Client Error*) : valide mais problématique à cause du client
  - 400 Bad Request
  - 415 Unsupported Media Type
  - 453 Not Enough Bandwidth
- 5xx (*Server Error*) : valide mais problématique du côté serveur
  - 500 Internal Server Error

## RTSP : commandes

La première ligne d'une requête est une commande :

- commandes principales (obligatoires)
  - **OPTIONS** : indique les commandes disponibles
  - **SETUP** : début d'une session RTSP et allocation des ressources coté serveur
  - **PLAY** : démarrage de la transmission d'un flux
  - **TEARDOWN** : arrêt définitif avec libération des ressources
- commandes additionnelles (optionnelles)
  - **DESCRIBE** : description technique d'un média (recom.)
  - **PAUSE** : arrêt temporaire de la transmission d'un flux (recom.)
  - **ANNOUNCE** : changement dans la description d'un média (SDP)
  - **RECORD** : demande d'enregistrement d'un flux
  - **REDIRECT** : redirection du client vers un nouveau serveur
  - **GET\_PARAMETER** : récupération de paramètres de transmission

## RTSP : exemple

```

C->W: GET /concert.sdp HTTP/1.1
Host: www.upmc.fr

W->C: HTTP/1.1 200 OK
Content-Type: application/x-rtsp

<session>
  <track src="rtsp://live.upmc.fr/concert/audio">
</session>

C->M: SETUP rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 2
      Transport: RTP/AVP;multicast

M->C: RTSP/1.0 200 OK
      CSeq: 2
      Transport: RTP/AVP;multicast;destination=224.2.0.1;
      port=3456-3457;ttl=16
      Session: 0456804596

C->M: DESCRIBE rtsp://live.upmc.fr/concert/audio RTSP/1.0
      CSeq: 1

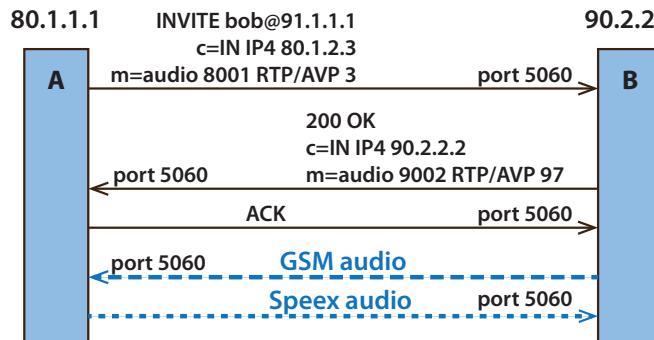
M->C: RTSP/1.0 200 OK
      CSeq: 1
      Content-Type: application/sdp
      Content-Length: 44

      v=0
      o=- 2890844526 2890842807 IN IP4 132.227.24.202
      s=RTSP Session
      m=audio 3456 RTP/AVP 0
      a=control:rtsp://live.upmc.fr/concert/audio
      c=IN IP4 224.2.0.1/16

      M->C: RTSP/1.0 200 OK
      CSeq: 3
      Session: 0456804596
  
```



## SIP : fonctionnement



- **négociation** du codec (réponse 606 not acceptable avec la liste des codec supportés)
- **rejet d'appel** (réponse busy, gone, payment, forbidden...)
- **transmission** (données multimédia envoyées avec RTP ou un autre protocole)

## Session Initiation Protocol (RFC 3261)

- mise en place d'appel
  - **notification** (intention d'établir un appel)
  - **négociation** (codages, média...)
  - **terminaison** (fin de l'appel/session)
- utilisateurs identifiés par noms et/ou e-mails (pas de numéros de téléphone)
- correspondance entre identifiants et adresses IP
- gestion des appels
  - **ajout de nouveaux média** pendant l'appel
  - **changement de codage** en cours
  - **invitation** de participants
- protocole de base de l'IMS (IP Multimedia Subsystem du 3GPP)

## SIP : exemple

```

INVITE sip:auto@localhost SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost" <sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhWWhMjYOY2M00Tc4YTI2MzgzZTN1YTRhZTMxNTE
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, Content-Type: application/sdp
Content-Type: application/sdp
Content-Length: 271
v=0
o=olivier 1208261984604 1208261984604 IN IP4 127.0.0.1
...
ACK sip:127.0.0.1:5060 SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:13764;rport
Max-Forwards: 70
Contact: <sip:matthew@127.0.0.1:13764>
To: "sip:auto@localhost" <sip:auto@localhost>;tag=tCAED3F5A
From: "Olivier"<sip:olivier@upmc.fr>;tag=5c7cdb68
Call-ID: NmNhWWhMjYOY2M00Tc4YTI2MzgzZTN1YTRhZTMxNTE
CSeq: 1 ACK
Content-Length: 0
Content-Length: 0
  
```

## SIP : conversion de nom et localisation

Comment faire correspondre l'identifiant SIP (nom/email) à l'adresse IP de l'appelé ?

- l'appelé peut être mobile
- il peut obtenir une adresse IP dynamique/privée
- il peut avoir plusieurs appareils IP (ordinateur, tablette, smartphone... etc.)

Avec quelle flexibilité ?

- en fonction du temps et du lieu
- en fonction de l'état de l'appel et/ou de l'appelé (transfert d'appel, etc.)

⇒ ce service est assuré par des serveurs SIP :

- serveur **SIP Registrar**
- serveur **SIP Proxy**



## SIP : Proxy

L'utilisateur envoie un message INVITE vers son **Proxy SIP**

- indique l'adresse SIP du destinataire (sip:olivier@upmc.fr)

Le serveur Proxy SIP est responsable de l'acheminement des messages SIP

- éventuellement à travers plusieurs proxys.
- l'appelé envoie la réponse à travers le(s) même proxy(s)

⇒ ensuite, le proxy de l'appelé renvoie une réponse SIP à l'appelant contenant son l'adresse IP

Le service fournis par le proxy SIP est similaire à celui d'un serveur DNS local



## SIP : Registrar

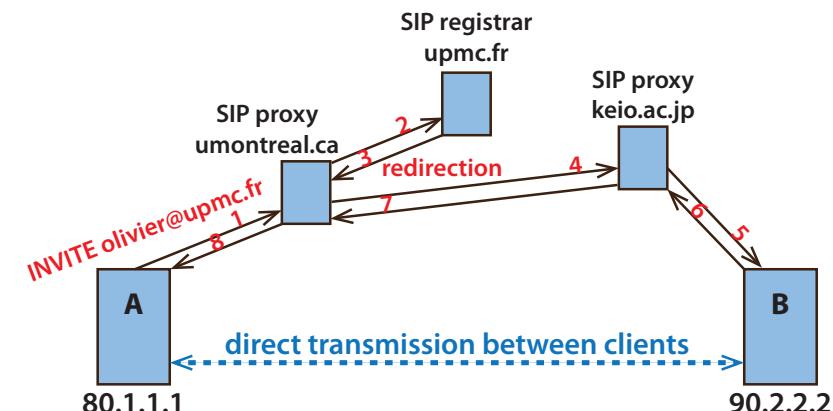
Au démarrage du client SIP de l'utilisateur :

⇒ émission d'un message REGISTER vers le **serveur SIP Registrar** de l'utilisateur.

```
REGISTER sip:upmc.fr SIP/2.0
Via: SIP/2.0/UDP 132.227.61.205:5061;rport;branch=z9hG4bKDC8595CD770E4317ACBC3
From: Olivier <sip:olivier@upmc.fr>;tag=1516659370
To: Olivier <sip:olivier@upmc.fr>
Contact: "Olivier" <sip:olivier@132.227.61.205:5061>
Call-ID: 46E1C3CB36304F84A020CF6DD3F96461@upmc.fr
CSeq: 37764 REGISTER
Expires: 1800
Max-Forwards: 70
User-Agent: LIP6-SIP-Phone v0.9
Content-Length: 0
```



## SIP : exemple



## SDP

### Session Description Protocol (RFC 4566)

- **description des sessions multimedia** pour leur initialisation
- présente les détails du média à transmettre/recevoir aux participants
  - adresses, identifiants, codecs, métadata, etc.
- ce n'est pas un protocole de transmission/transport
  - **seulement de la description**
- peut être utilisé sur un protocole de transport quelconque
- peut être intégré à tout protocoles d'initialisation/signalisation
  - RTSP, SIP, etc.
- **format standard** de présentation
  - indication de contenu d'une description SDP avec Content-Type: application/sdp
  - la description SDP est une suite de ligne de texte de type <type>=<value>



## SDP : exemple

```
v=0
o=fourmaux 2990844526 2990842807
    IN IP4 132.227.63.51
s=SDP Seminar
i=A Seminar on SDP
u=http://www.upmc.fr/sem/sdp.pdf
e=olivier.fourmaux@upmc.fr
c=IN IP4 224.2.17.12/127
t=2973397496 2973404696
a=recvonly
m=audio 49170 RTP/AVP 0
m=video 51372 RTP/AVP 99
a=rtpmap:99 h263-1998/90000
```

### Types SDP possibles :

v= (protocol version)  
o= (originator and session identifier)  
s= (session name)  
i= (session information)  
u= (URI of description)  
e= (email address)  
p= (phone number)  
c= (connection information)  
b= (0 or more bandwidth information lines)  
t= (time the session is active)  
r= (0 or more repeat times)  
z= (time zone adjustments)  
k= (encryption key)  
a= (0 or more session attribute lines)  
 0 or more media descriptions  
 m= (media name and transport address)  
 i= (media title)  
 c= (connection information)  
 b= (0 or more b.w. information lines)  
 k= (encryption key)  
 a= (0 or more media attribute lines)



## Video



## Début de la vidéo sur Internet

### avant 1995 : Internet académique

- début des recherches sur la vidéo

### 1995 - 2005 : Internet du web

- 1<sup>re</sup> génération d'applications commerciales (téléchargement progressif via un lecteur dédié)

### 2005 - 2010 : début de l'Internet de la vidéo

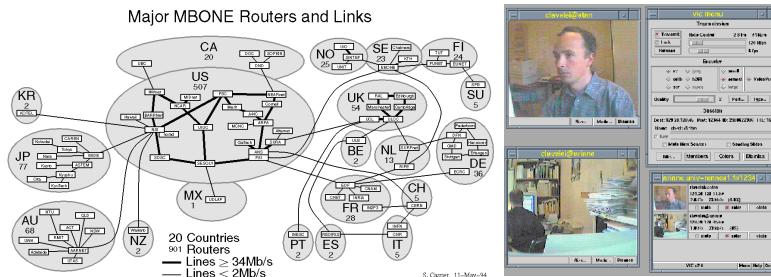
- 2<sup>me</sup> génération d'applications commerciales avec RTSP, et surtout RTMP (**flash** standard de facto)

### 2010... : Internet massivement pour la vidéo

- 3<sup>me</sup> génération d'applications commerciales avec le **HTTP Streaming**...



## Recherche sur la vidéo sur Internet - années 1990



- support de la qualité de service dans le réseau
  - signalisation, réservation de ressources et ordonnancement de paquet (RSVP/IntServ)
  - prioritisation de trafic (DiffServ)
- communications multicast
- ➡ application cible : **vidéoconférence multi-partenaire**

## 1<sup>re</sup> génération d'app. vidéo commerciales sur Internet (1)

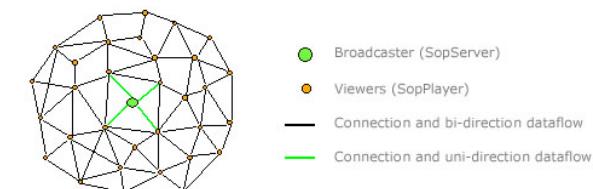


- lecture vidéo simple
- intégration avec le navigateur web limitée
- protocoles et serveurs propriétaires
- pas assez de contenu disponible sur Internet
- pas assez d'accès Internet à haut débit
- ➡ période de l'internet du web (1995-2005)

## Recherche sur la vidéo sur Internet - années 2000

Difficulté à introduire de nouveaux mécanismes dans les réseaux

- applications adaptatives
- approches P2P
  - multicast applicatif
  - BitTorrent
  - Coolstreaming, SopCast, PPLive, PPStream...



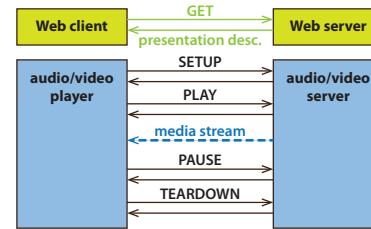
## 1<sup>re</sup> génération d'app. vidéo commerciales sur Internet (2)

- téléchargement complet
  - objets récupéré via HTTP puis lecture via *player* dédié
    - ✗ pas de *pipelining*
- téléchargement progressif
  - *metafile* récupérée et transmise au *player* dédié
  - le *player* établit une communication directe vers le serveur web
    - ✗ HTTP transfert les données aussi rapidement que possible
    - ✗ transmission importante même si arrêt de la lecture
    - ✗ inadaptation à l'usage de l'audio/vidéo (stop, lecture, accès direct)
    - ✗ surcoût de TCP
    - ✗ pas d'adaptation aux conditions du réseau

## 2<sup>me</sup> génération d'app vidéo commerciales sur Internet (1)

### Protocoles de streaming dédiés

- protocoles avec un état par session
- séparation des connexion contrôle et données
  - contrôle : lecture, pause, retour arrière, avance rapide...
  - données : UDP ou TCP (mais pas HTTP)
- exemple : RTSP, RTMP...



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Actuellement 2010 ➔



Nouveaux écrans : smart TV, smartphones, tablettes, consoles de jeux...

- nouveaux besoins (encodages multiples : + de 4 types d'écran)
- nouvelles plateformes où Flash n'est pas forcément disponible (iOS, Android, consoles...)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## 2<sup>me</sup> génération d'app vidéo commerciales sur Internet (2)



Flash : technologie principale pour le décodage de la vidéo

- *player* de facto coté client (95% PC, tous les OS, tous les navigateurs)
- environnement de programmation multi-plateforme (Action script)
- contenu multimédia riche (avec bonne intégration au navigateur)
- video FLV : codec Sorenson, VP6 puis H264
- protocole RTMP
  - ✗ problèmes de sécurité, filtrage des ports non standard
  - ✗ servers web plus rentables que serveurs dédiés



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## 3<sup>ème</sup> génération : HTTP Streaming

Adaptation de la distribution à l'Internet (plutôt que l'inverse)

- architecture **orientée client** (client avec état, serveur sans état)
  - serveur standard : serveurs web
  - protocole standard : HTTP
  - état de la session et régulation du flux du coté du client
- vidéo **découpée en multiples chunks**
  - un *chunk* démarre sur une image de référence
  - téléchargement progressif de chaque *chunk* via HTTP
  - lecture séquentielle des *chunk* ➔ vidéo continue
- adaptation aux différents débits
  - ➔ **adaptive HTTP streaming**



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## Protocole de découpage multi-chunk pour HTTP

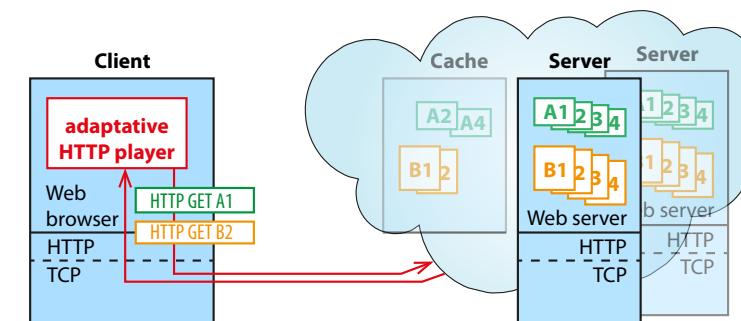


Mécanismes de découpage en multiple *chunk* pour HTTP

- encodage à différents niveaux de qualité
- adaptation par le client : demande du *chunk* avec l'encodage adapté
- ➡ tous les chunks encodent les mêmes segments de vidéo



## Avantages du streaming HTTP adaptatif



Motivation pour une large adoption

- orienté client ➡ commutation serveur/CDN
- passage des pare-feu et des boîtiers intermédiaires
- réutilise les infrastructure CDN existantes



## Exemples de protocoles de streaming HTTP adaptatif

- Apple HLS : HTTP Live Streaming
- Microsoft IIS Smooth Streaming
- Adobe HDS : HTTP Dynamic Streaming
- MPEG DASH : Dynamic Adaptive Streaming over HTTP



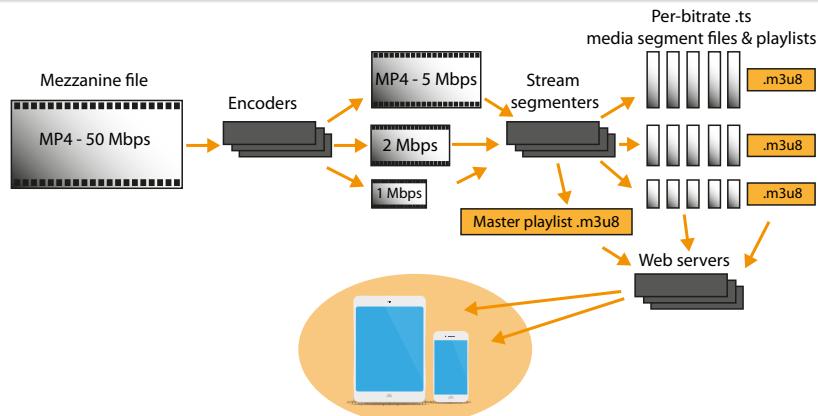
## Apple HLS (1)

*HTTP Live Streaming (2009)*

- segmentation du contenu en petits **objets HTTP standards** (10s)
- chargement initial d'une liste de lecture **M3U étendues**
- composants de la solution Apple :
  - **serveurs d'encodage** : H.264 + MP3, HE-AAC ou AC-3 dans un conteneur MPEG transport stream (.ts)
  - **serveurs de segmentation** : découpage en petits fragments de taille identique (+ création des index .m3u8)
  - **serveurs de diffusion** : serveur HTTP standard
  - **clients** : après avoir chargé les index, doivent pouvoir ré-assembler le flux adéquat
- DRM possibles
- **format public** (draft IETF informational RFC)
  - ➡ large support (la plupart des clients et serveurs multimédia implémentent HLS)



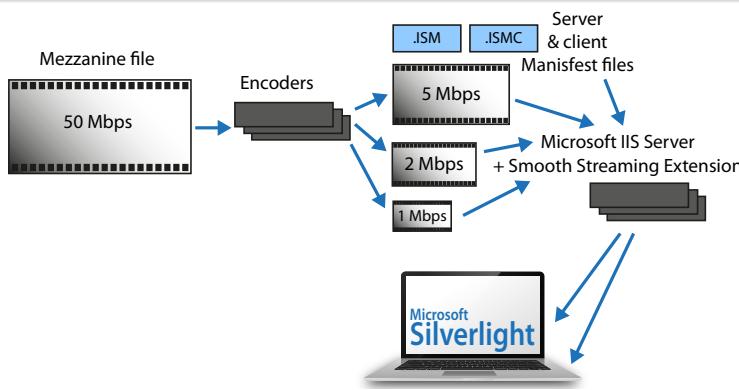
## Apple HLS (2)



Mécanismes de base :

- récupération des listes de lectures principales et spécifiques
- HTTP GET sur les différents segments .ts

## Microsoft Smooth Streaming (2)



Mécanismes de base :

- récupération du fichier "manifest" pour le client (.ismc)
- HTTP GET sur les différents fragments (url avec les informations de débit et de temps)

## Microsoft Smooth Streaming (1)

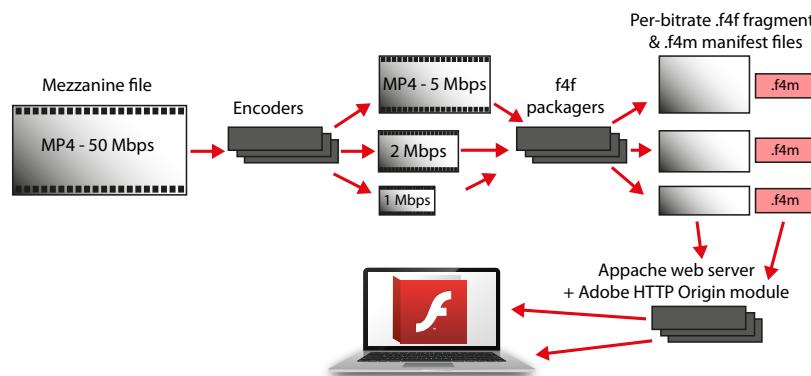
- vidéo "live" et à la demande avec Silverlight
- composants de la solution Microsoft :
  - **encodeurs** : Microsoft Expression ou autre (VC1, MP4...)
  - **serveur** : Microsoft IIS avec extension Smooth Streaming du IIS media service
  - **clients** : nécessite le player Microsoft Silverlight
- fichiers associés :
  - contenu intégré dans un conteneur ISO base media (MPEG-4 part 12) adapté au streaming (gestion de cache et accès aléatoire)
  - **ISM** : fichier "manifest" décrivant les fichiers média (codecs, débits et temps)
- requêtes de type : [http://video.foo.com/NBA.ism/QualityLevels\(400000\)/Fragments\(video=610275114](http://video.foo.com/NBA.ism/QualityLevels(400000)/Fragments(video=610275114)

## Adobe HDS (1)

### HTTP Dynamic Streaming

- vidéo "live" et à la demande avec un **client Flash** via **HTTP**
- composants de la solution Adobe :
  - **File Packager** : gère la fragmentation (6s) et les fichiers F4F
  - **HTTP Origin Module** : module pour les serveurs HTTP Apache
  - **OSMF Media Players** (inclus dans les lecteurs Flash)
- fichiers associés :
  - **F4F** : contenu FLV ou MP4 intégré dans un conteneur ISO base media (MPEG-4 part 12) adapté au streaming (gestion de cache et accès aléatoire)
  - **F4M** : fichier "manifest" décrivant les fichiers F4F assemblés (codecs, résolutions, débits et DRM)
- requêtes de type : <http://www.example.com/media/http-dynamic-StreamingSeg1-Frag1>

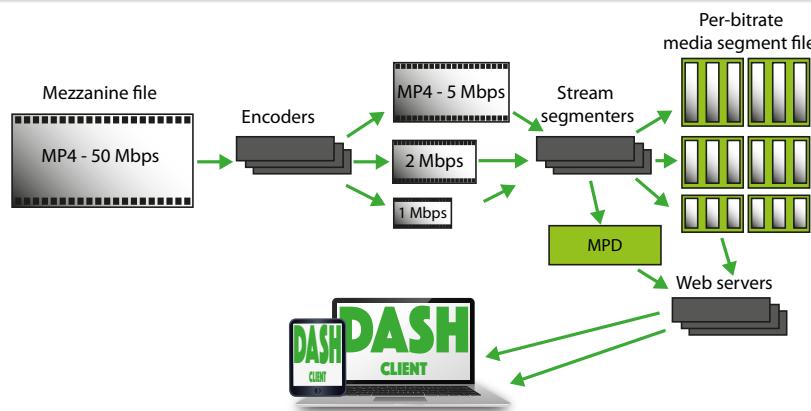
## Adobe HDS (2)



Mécanismes de base :

- récupération du fichier "manifest" .f4m
- HTTP GET sur les différents fragments (url avec les informations de débit et de temps)

## DASH : architecture



Mécanismes de base :

- récupération du fichier "manifest" MPD
- HTTP GET sur les différents segments nécessaires

## MPEG DASH ISO/IEC 23009-1

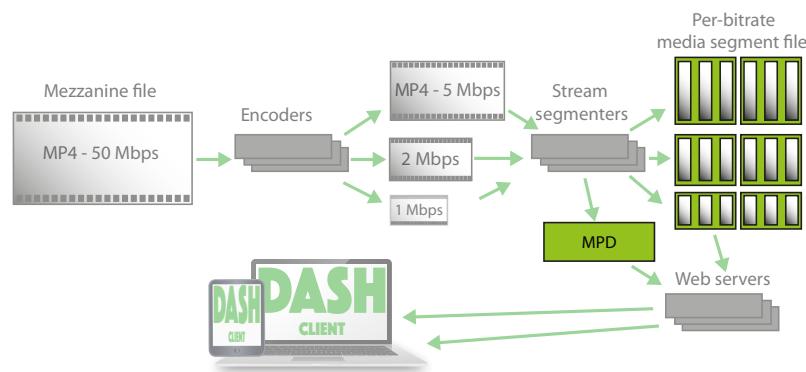
*MPEG's Dynamic Adaptive Streaming over HTTP (2012)*

- motivation
  - HTTP est nécessaire (caches/CDN, NAT/Firewalls, adaptation à l'état du réseau...)
  - nombreux types d'équipements
  - nombreuses solutions de streaming sur HTTP
- principes
  - transmission de petits morceaux de vidéo via HTTP
  - pas de modification de l'infrastructure de distribution
  - contrôle et réassemblage par le client
  - adaptation au client et aux conditions du réseau
- cible : OTT sur tout équipements

## DASH : principes de base

- ce que DASH n'est pas :
  - protocole, codec, middleware, spécification du client...
- DASH est un catalyseur :
  - définit les **formats** pour permettre une diffusion performante sur Internet
- DASH permet :
  - la réutilisation des techno existantes (conteneurs, codecs, DRM...)
  - l'utilisation de l'infrastructure Web (cache/CDN)
  - l'amélioration de l'expérience de l'utilisateur (démarrage rapide, limitation du rebuffing)
  - l'adaptation aux besoins de l'utilisateurs, la capacité de ses équipements et à l'état du réseau
  - le contrôle au niveau des clients (différenciation)
  - la commutation de qualité (switching) imperceptible
  - les flux "live" ou pré-enregistré (DVD-like)
  - l'interopérabilité entre flux

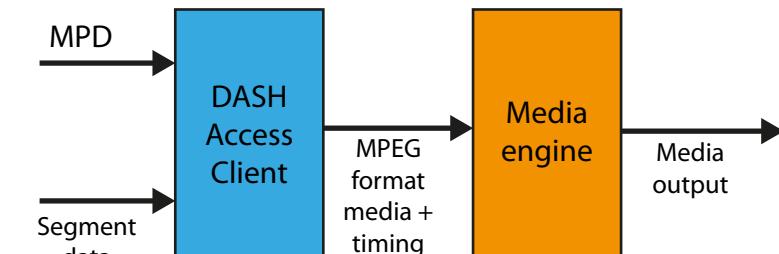
## DASH : ce qui est spécifié



- présentation des données sur le serveur HTTP
- fichier MPD (Media Presentation Description)
- structure des URL pour retrouver les segments via HTTP

## DASH : classification des informations

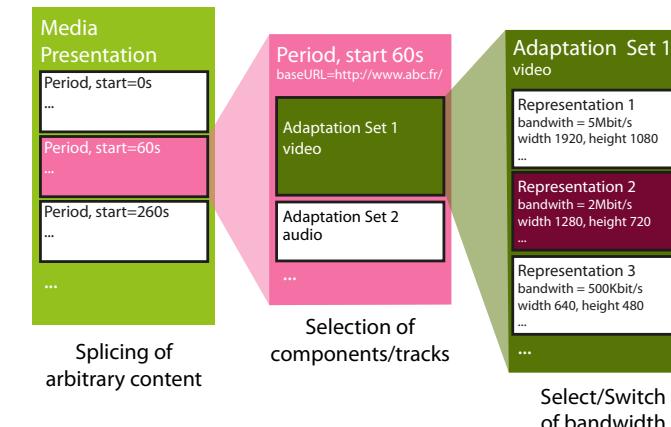
- MPD et informations indexées pour le client DASH
  - cœur des spécifications DASH
- initialisation et flux de contenu pour le lecteur
  - réutilisation des containers existants



## DASH : MPD

- information pour sélectionner les éléments des différentes représentations :
  - débit, résolution, codec, language, DRM...
- informations temporelles et d'accès :
  - URL et position dans chaque segment accessible (byte-range)
  - disponibilité des segments (début/fin en temps absolu)
  - temps de présentation initial et durée relative des segments
  - délais recommandé de lecture (live)
- information de découpage et de commutation entre les représentations

## DASH : Structure MPD



## DASH : exemple de MPD pour vidéo segmentée

```
<MPD xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" profiles="urn:mpeg:dash:profile:isoff-main:2011" type="static" mediaP
<BaseURL>http://www.itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_10s/</BaseURL>
<Period start="PT0S">
  <AdaptationSet bitstreamSwitching="true">
    <Representation id="0" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="45373">
      <SegmentBase>
        <Initialization sourceURL="bunny_10s_50kbit/bunny_50kbit_dash.mp4"/>
      </SegmentBase>
    <SegmentList duration="10">
      <SegmentURL media="bunny_10s_50kbit/bunny_10s1.m4s"/>
      <SegmentURL media="bunny_10s_50kbit/bunny_10s2.m4s"/>
      <SegmentURL media="bunny_10s_50kbit/bunny_10s3.m4s"/>
      ...
      <SegmentURL media="bunny_10s_50kbit/bunny_10s60.m4s"/>
    </SegmentList>
  </Representation>
  <Representation id="1" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="88482">
    <SegmentBase>
      <Initialization sourceURL="bunny_10s_100kbit/bunny_100kbit_dash.mp4"/>
    </SegmentBase><SegmentList duration="10">
      <SegmentURL media="bunny_10s_100kbit/bunny_10s1.m4s"/>
      <SegmentURL media="bunny_10s_100kbit/bunny_10s2.m4s"/>
      ...
    </SegmentList>
  </Representation>
  <Representation id="19" codecs="avc1" mimeType="video/mp4" width="1920" height="1080" startWithSAP="1" bandwidth="37924">
    <SegmentBase>
      <Initialization sourceURL="bunny_10s_8000kbit/bunny_8000kbit_dash.mp4"/>
    </SegmentBase>
    <SegmentList duration="10">
      ...
      <SegmentURL media="bunny_10s_8000kbit/bunny_10s60.m4s"/>
    </SegmentList>
  </Representation>

```



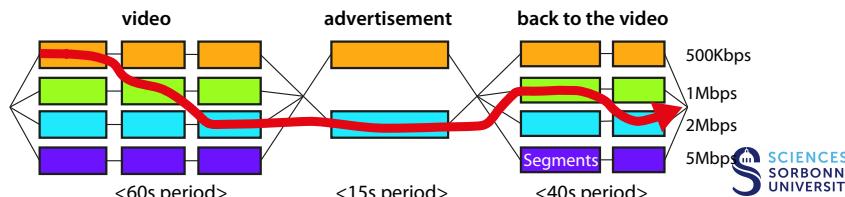
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

```
...
<SegmentURL media="bunny_10s_8000kbit/bunny_10s60.m4s"/>
</SegmentList>
</Representation>
```

## DASH : mécanismes coté client

- processus client :
  - téléchargement du **fichier MPD**
  - téléchargement **segment par segment** en fonction de la lecture
  - **détermination du débit disponible**
- facteur de choix pour la représentation :
  - état des tampons **mémoires**
  - état du **réseau**
  - état de l'**équipement**
  - changement de résolution de l'utilisateur



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## DASH : exemple de MPD pour vidéo non segmentée

```
<MPD xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" profiles="urn:mpeg:dash:profile:isoff-main:2011" type="static" mediaP
<BaseURL>http://www.itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_10s/</BaseURL>
<Period start="PT0S">
  <AdaptationSet bitstreamSwitching="true">
    <Representation id="0" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="45373">
      <SegmentBase>
        <Initialization sourceURL="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" range="0-864"/>
      </SegmentBase>
    <SegmentList duration="10">
      <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="865-57231"/>
      <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="57232-114771"/>
      <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="114772-171506"/>
      ...
      <SegmentURL media="bunny_10s_50kbit/bunny_50kbit_dashNonSeg.mp4" mediaRange="3344628-3383017"/>
    </SegmentList>
  </Representation>
  <Representation id="1" codecs="avc1" mimeType="video/mp4" width="320" height="240" startWithSAP="1" bandwidth="88482">
    <SegmentBase>
      <Initialization sourceURL="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" range="0-866"/>
    </SegmentBase><SegmentList duration="10">
      <SegmentURL media="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" mediaRange="867-112515"/>
      <SegmentURL media="bunny_10s_100kbit/bunny_100kbit_dashNonSeg.mp4" mediaRange="112516-229893"/>
      ...
    </SegmentList>
  </Representation>
  <Representation id="19" codecs="avc1" mimeType="video/mp4" width="1920" height="1080" startWithSAP="1" bandwidth="37924">
    <SegmentBase>
      <Initialization sourceURL="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" range="0-870"/>
    </SegmentBase>
    <SegmentList duration="10">
      ...
      <SegmentURL media="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" mediaRange="281330526-282757919"/>
    </SegmentList>
  </Representation>

```

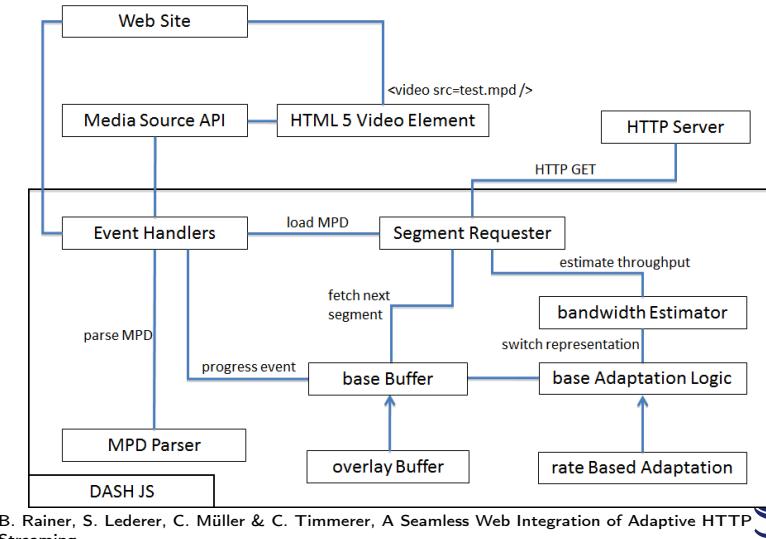


Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

```
...
<SegmentURL media="bunny_10s_8000kbit/bunny_8000kbit_dashNonSeg.mp4" mediaRange="281330526-282757919"/>
</SegmentList>
</Representation>
```

## DASH : exemple de client



B. Rainer, S. Lederer, C. Müller & C. Timmerer, A Seamless Web Integration of Adaptive HTTP Streaming,



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

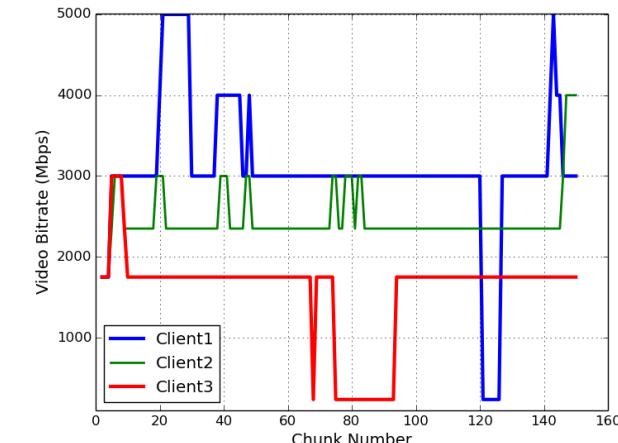
Architecture des Réseaux (ARes) 2/5 : Application

## DASH : commutation et segments

- supports pour la commutation
  - **alignement des segments** (permet un décodage non recouvrant entre différentes représentations)
  - **point d'accès du flux** (stream access points - SAP - position dans un segment où la commutation est préférable)
  - **encodages adaptés** à la concaténation de différentes présentation (structures similaires)
- formats des segments
  - **ISO base media FF et MPEG-2 TS**
  - possibilité d'intégrer d'autres formats
  - indépendant des codec



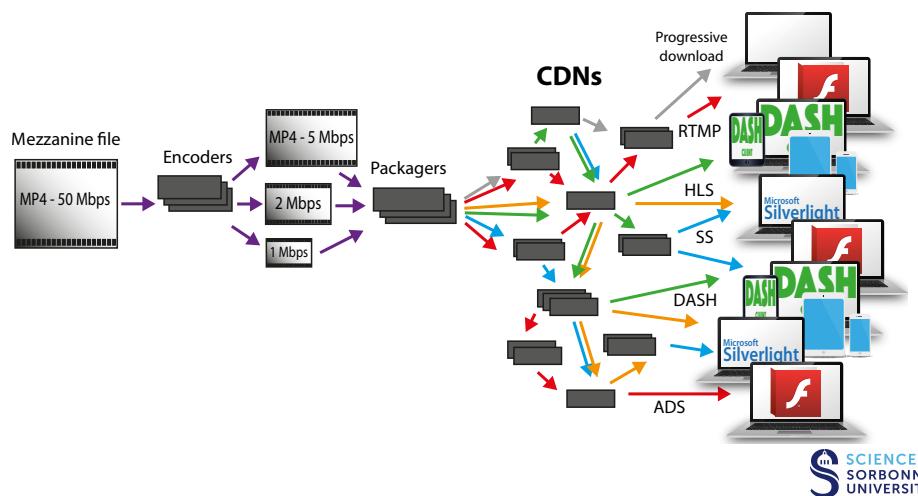
## DASH : stabilité



Courbe réalisée par Rabee Mustapha Abuteir (mustapha.abuteir@lip6.fr)



## Distribution vidéo actuelle



## Trafic de l'Internet vidéo

*A world full of elephants*

- croissance du trafic ➔ 95% vidéo !
  - vidéo : X 100
  - autres applications : X 10
- le segment HTTP (chunk) est le nouveau datagramme
  - support de HTTP par tous les équipements du réseau
- garantir la QoE
  - pas de support universel de la QoS dans l'Internet
  - adaptation aux besoins par mécanismes adaptatifs multi-débits
  - **le CDN est l'élément clé pour optimiser la performance**



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

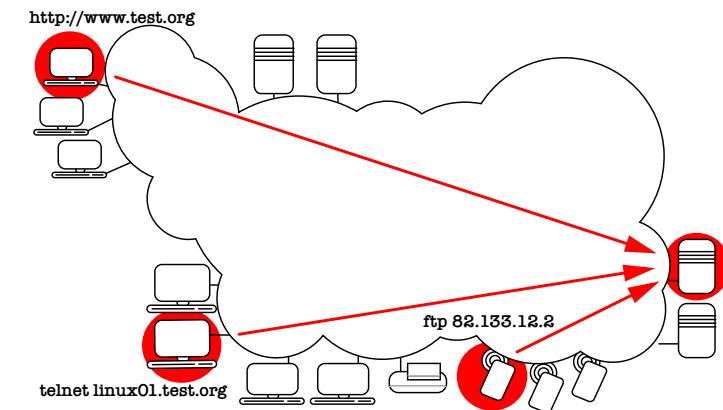
- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## Correspondance noms – adresses



## Annuaire

Conversion des noms littéraux des hôtes de l'Internet en adresses numériques

- initialement
  - un fichier
  - espace de "nommage" à plat
  - gestion centralisée par un *NIC* (*Network Information Center*)
- actuellement : **DNS**
  - base de données **distribuée**
  - espace de "nommage" **hiérarchique**
    - décorrélé de la topologie physique
  - système contrôlé par l'*InterNIC* (1992-1998) et puis l'*ICANN* (*Internet Corporation for Assigned Names and Numbers*) et ses nombreux délégues
    - délégation hiérarchique (proche de celle du "nommage")
    - taille des délégations raisonnables
  - protocole d'échange...



## DNS (*Domain Name System*)

Annuaire standard de l'Internet (RFC 1034 et RFC 1035)

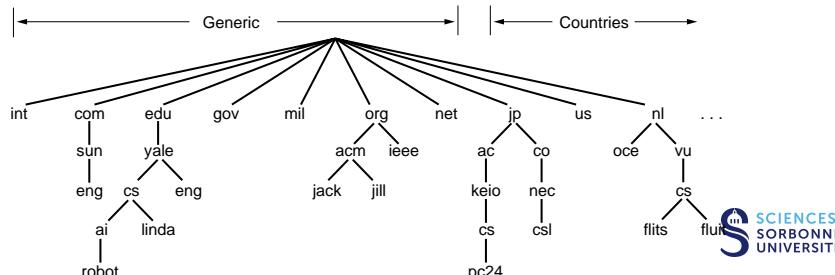
- espace de "nommage" hiérarchique et système de délégation
- **serveurs de noms** (serveurs DNS)
  - composants physiques de la **hiérarchie** supportant la base distribuée
  - gèrent les requêtes DNS
  - transport sur **UDP** ou **TCP**, port **53**
  - les applications y accèdent à travers le **resolver** (UNIX) :
    - `gethostbyname` (3), `gethostbyaddr` (3)
- services :
  - *name resolving*
  - *host aliasing*
  - *mail server aliasing*
  - *load distribution...*
- exemple :
  - BIND (*Berkeley Internet Name Domain*)
  - named (UNIX)



## DNS : Espace de "nommage"

Système de "nommage" hiérarchique

- structure arborescente (~ système de fichier Unix)
- label d'un nœud : 63 car. max. (A..Z0..9- insensible à la casse)
- **domain name** = liste des labels en parcourant l'arbre vers la racine (255 car. max. au total et "." séparateur de label) :
  - absolu (**FQDN**) : pc24.CS.keio.ac.jp.
  - les noms relatifs sont gérés localement (hôte)



## DNS : ccTLD (*country code Top Level Domain*)

ccTLD (ISO 3166)	240 countries and external territories
.ac	Ascension Island
.af	Afghanistan
.aq	Antarctica (-60°S)
.eu	European Union
.fr	France
.gf	French Guiana
.gp	Guadeloupe
.mq	Martinique
.pf	French Polynesia + Clipperton
.pm	Saint-Pierre and Miquelon
.re	Réunion
.tf	TAAF
.ru	Russia (+.su)
.tv	Tuvalu
.uk	United Kingdom (+.gb)
.us	United States
.zw	Zimbabwe

## DNS : gTLD (*generic Top Level Domain*)

.aero	2001	Air-transport industry *	SITA
.asia	2006	Asia-Pacific region *	Afilias
.biz	2001	Unrestricted	NeuLevel
.cat	2005	Catalan lingu. & cult.*	Asso. puntCAT
.com/.net	1985	Unrestricted	VeriSign
.coop	2001	Cooperative *	DotCooperation
.edu	1985	(US) educational inst. *	VeriSign
.gov	1985	US government *	US Admin.
.info/.org	01/85	Unrestricted	Afilias
.int	1988	Internat. organisations	ICANN
.job	2005	Human resrc. managment*	Employ Media
.mil	1985	US military *	US DoD NIC
.mobi	2005	Mobile device use *	Mobi JV
.museum	2001	Museums *	MuseDoma
.name	2001	Individuals	VeriSign
.pro	2001	Professionals	RegistryPro
.tel	2005	Internet Tel. serv.*	Telnic Limited
.travel	2005	Travel industry*	Tralliance Corp.

## DNS : Nouveau TLD depuis 2012

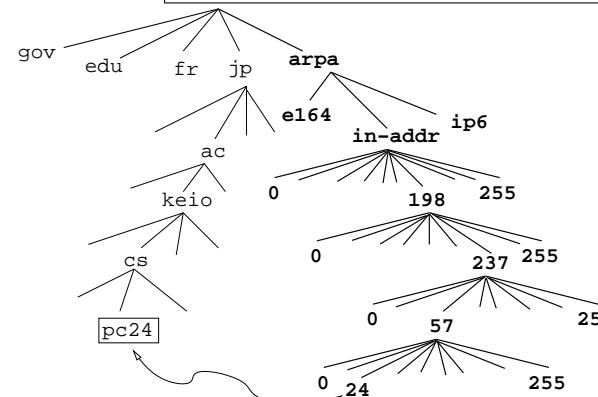
En 2012, l'ICANN, a autorisé ~2000 nouveaux TLD

- Grande diversité de TLD :
  - marques (.google, .danone, etc.)
  - communautés (.archi, .immo, etc.)
  - zones géographiques (.paris, .nyc, etc.)
  - jeux de caractères non ASCII recodée avec Punycode/RFC3492 (.移动 qui se code .xn-6frz82g, idéogramme pour "mobile", etc.)
  - inclassable (.photo, .wtf, etc.)

## DNS : Domaine .arpa

Résolution : pc24.cs.keio.ac.jp. ➡ ?

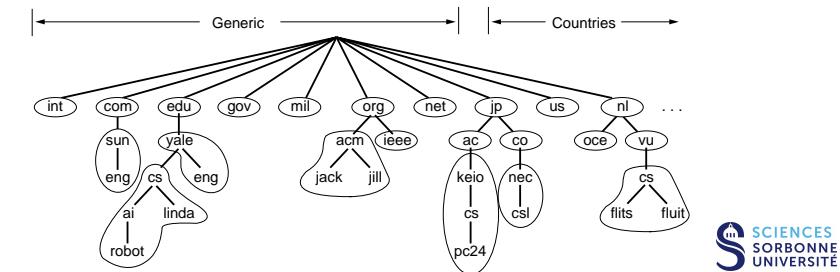
Résolution inverse : 24.57.237.198.in-addr.arpa. ➡ ?



## DNS : zones (1)

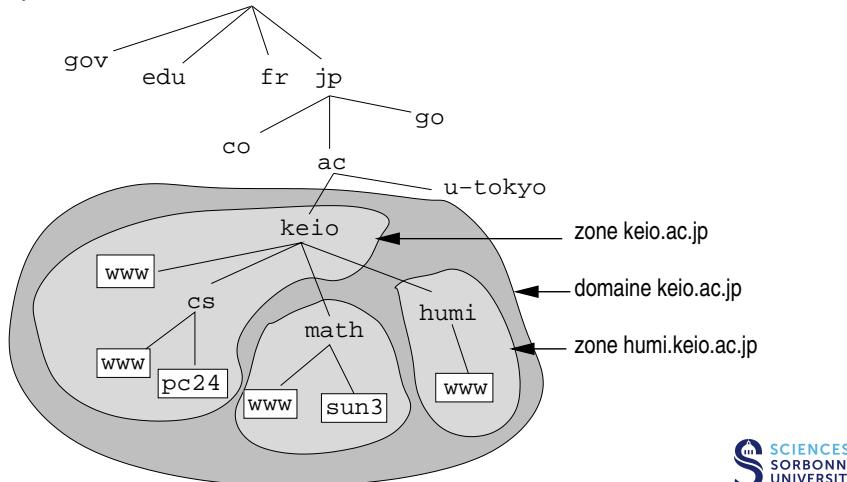
ICANN gère la racine et délègue les TLDs à des *domain name registry*

- zones (sous-arbres de l'arbre DNS) administrés séparément
  - (~ partitions physiques d'un système de fichier Unix)
  - délégation des noms de sous-domaines correspondants
  - exemple : keio.ac.jp.
- des serveurs de noms y sont associés



## DNS : zones (2)

Ne pas confondre zone et domaine !



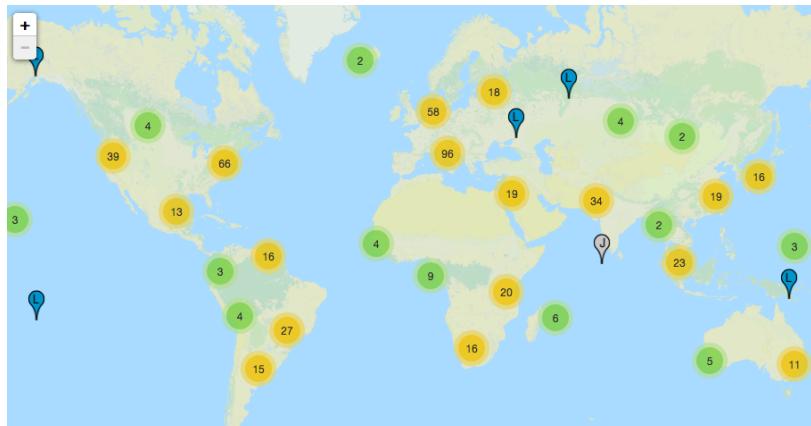
## DNS : serveurs de noms

Différents types de serveurs de noms

- serveurs de référence d'une zone :
  - un  **primaire (primary name server)**
    - informations de référence (*authoritative records*)
    - connaissance de ses descendants (délégations)
    - initialisation locale (disque)
  - un ou plusieurs **secondaire (secondary name server)**
    - redondance : complètement séparé du primaire
    - initialisation et m-à-j. à partir du primaire (**transfert de zone**)
  - physiquement indépendant de la zone
- serveurs locaux (accès au service)
  - résolution *top-down* (des TLD vers les sous-domaines)
  - connaissance des serveurs racines (*root name server*)
    - 1 primaire et 12 secondaires, haute disponibilité (anycast)
    - config. en dur ([ftp.rs.internic.net/domain/named.root](http://ftp.rs.internic.net/domain/named.root))
  - requêtes **récurssives ou itératives**



## Root-servers (1)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

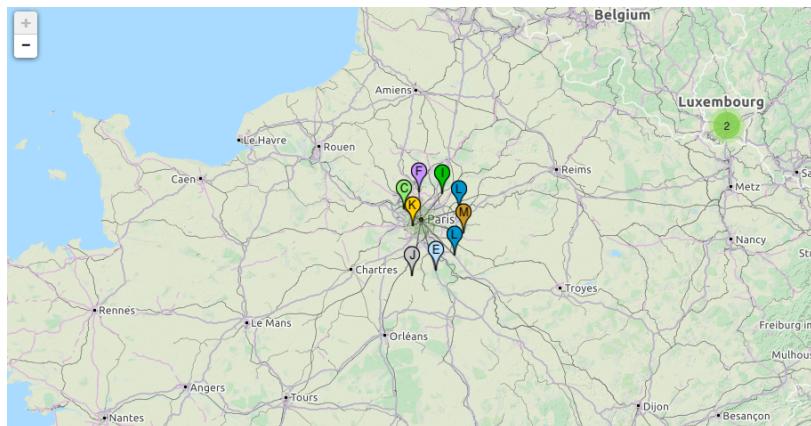
## Root-servers (2)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

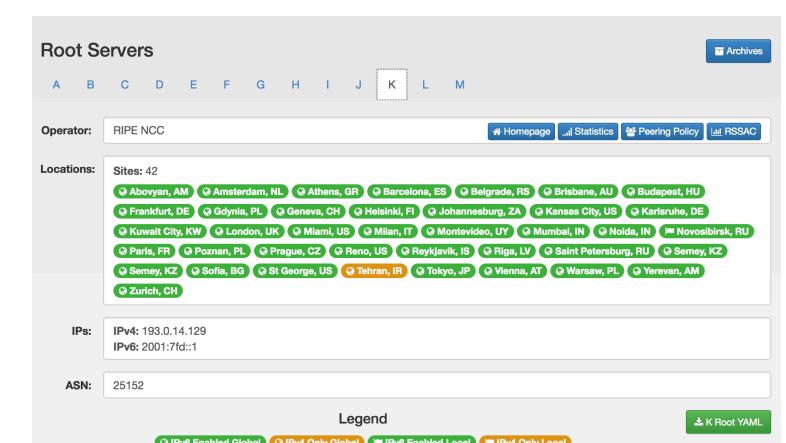
## Root-servers (3)



Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

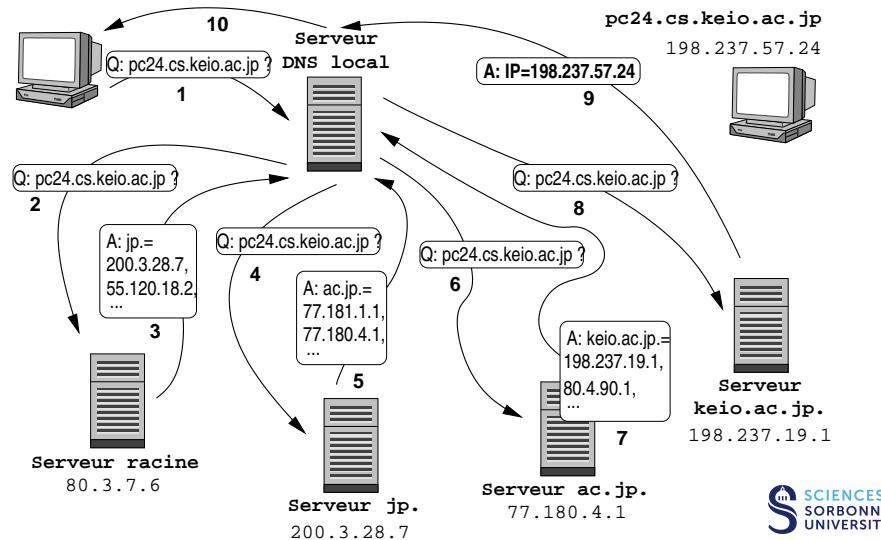
## Root-server K



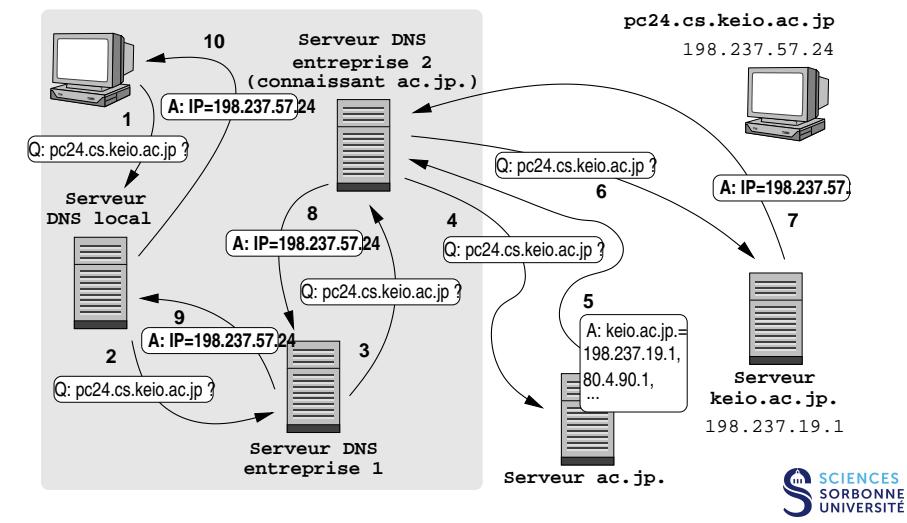
Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

Architecture des Réseaux (ARes) 2/5 : Application

## DNS : requête itérative



## DNS : requête récursive



## DNS : performances

### Capacité du système DNS à supporter la charge ?

- problèmes liés à la consultation systématique de la racine
  - ne tient pas compte de la localité des requêtes
    - serveur local généralement distinct du serveur de référence
  - charge sur les serveurs racines
    - combien de requêtes pour tout l'Internet ?
    - disponibilité des serveurs racines
      - passage obligé pour toute requête
- utilisation de cache
  - informations de seconde main (*non-authoritative records*)
  - réponses d'un serveur de référence inclue un délai de validité (TTL)
    - réponses pour les TLD sur les serveurs racines valide 48h
      - 100.000 requêtes par secondes (2005)

## DNS : format général du message

0	15	16	bit 31	flags :
identificateur				QR (1 bit) : 0 = question, 1 = réponse
nombre de questions				opcode (4 bit) 0 = standard ...
nombre de serveurs				AA (1 bit) : 1 = réponse autoritaire
Questions				
Champs des réponses				
Champs des serveurs de référence				
Champs des informations additionnelles				

## DNS : format d'une question

0	15	16	bit 31
---	----	----	--------

Nom (non aligné sur 32bits)

Type	Classe
------	--------

- **Nom** : N octets, chaque nom de label est précédé par un octet indiquant le nombre de caractères (si >0x3F alors si 0xCOZZ = renvoi à ZZ octets du début du message). Terminé par 0x00.

4, 'p', 'c', '2', '4', 2, 'c', 's', 4, 'k', 'e', 'i', 'o', 2, 'a', 'c', 2, 'j', 'p', 0

- **Type (16 bits)** :

val	nom	description	val	nom	description
1	A	adr. IPv4	13	HINFO	info sur l'équip.
2	NS	nom serv.	15	MX	serveur messag.
5	CNAME	alias	28	AAAA	adresse IPv6
6	SOA	zone gérée	...		
12	PTR	point. nom	255	*	tt types (quest.)



- **Classe (16 bits) : 1 = Internet**

## DNS : annuaire inversé

### Conversion des adresses numériques en noms littéraux

- requêtes de type **pointeur de nom (PTR)**
  - adresse IPv4
    - 198.237.57.24
  - conversion dans le domaine in-addr.arpa
    - 24.57.237.198.in-addr.arpa
      - ⇒ souvent utilisé pour vérifier les droits d'accès

## DNS : format d'un champ réponse

0	15	16	bit 31
---	----	----	--------

Nom (non aligné sur 32bits)

Type	Classe
------	--------

TTL

Taille des données  
(o.)

Données

- **Nom, Type, Classe** : idem
- **TTL** (32 bits) : validité en secondes
- **Taille des données (16 bits)** : en octets
- **Données** (N octets sans bourrage) :
  - Nom (chaîne codée comme pour une question) NS, CNAME sur 4 octets, AAAA sur 16
  - Adresses (valeur numérique) A sur 4 octets, AAAA sur 16



## DNS : obtention d'une délégation

Pour être référence pour un sous domaine officiel :

- réservation du nom du domaine auprès d'un *domain name registrar*
- mise en place de serveurs conformes à la norme DNS
  - information de référence de la zone
    - réplication dans au moins un serveur secondaire
  - si sous délégations :
    - connaissance des serveurs descendants
  - si gestion des adresses IP correspondantes :
    - information de référence des pointeurs de nom

## DNS : modification dynamique

### Dynamique DNS (RFC 2136)

- pour fonctionner avec l'auto-conf. des hôtes (DNS local) :
  - update
  - notification
- problèmes de sécurité...

### Service DNS dynamique (prestataire externe)

- pour fonctionner avec une adresse dynamique (accès résidentiels) :
  - serveur : dyndns.org, no-ip.org...
  - client spécifique indiquant le changement d'adresse (*host/setupbox*)
  - délégation virtuelle (sous domaine de 3ème niveau)
    - toto123.myftp.biz
    - toto123.blogspot.org
    - toto123.homenet.org
    - toto123.dyn-o-saur.com
    - toto123.endofinternet.net...



## DNS : exemple

```
Unix> dig www.math.keio.ac.jp

;; Got answer:
;; ->>HEADER<- opcode: QUERY, status: NOERROR, id: 11895
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 3, ADDITIONAL: 4

;; QUESTION SECTION:
;www.math.keio.ac.jp.      IN   A

;; ANSWER SECTION:
www.math.keio.ac.jp.    3600  IN   CNAME  sun3.math.keio.ac.jp.
sun3.math.keio.ac.jp.    3600  IN   A      131.113.70.3

;; AUTHORITY SECTION:
math.keio.ac.jp.        3600  IN   NS     relay.math.keio.ac.jp.
math.keio.ac.jp.        3600  IN   NS     ns.st.keio.ac.jp.
math.keio.ac.jp.        3600  IN   NS     ns0.sfc.keio.ac.jp.

;; ADDITIONAL SECTION:
relay.math.keio.ac.jp.  3600  IN   A      131.113.70.1
ns.st.keio.ac.jp.       127   IN   A      131.113.1.8
ns0.sfc.keio.ac.jp.    1199  IN   AAAA  3ffe:501:1085:8001::121
ns0.sfc.keio.ac.jp.    2358  IN   A      133.27.4.121

;; Query time: 577 msec MSG SIZE rcvd: 206
```



## DNS : sécurité

Pas de sécurité dans le protocole de base (RFC 3833)

- interception / modification de message DNS
- faux messages (*DNS cache poisoning*)
- déni de service...

DNSSEC (RFC 4033 à 4035 + RFC 4310 + RFC 4641)

- extension du système DNS permettant :
  - authentification de l'origine des données
  - authentification du déni d'existence
  - intégrité des données
- obligatoire pour sécuriser les *DNS update*
- limitations :
  - DNSSEC ne garantie pas la confidentialité des données
  - DNSSEC ne protège pas des attaques DDoS



## ARes : plan du cours 2/5

### 1 Applications historiques

- introduction
- connexion à distance
- transfert de fichiers

### 2 Applications principales

- messagerie électronique
- World Wide Web
- multimédia

### 3 Applications support

- annuaire (DNS)
- administration de réseau



## administration de réseau

Développement du réseau (nombreux équipements et machines à gérer)

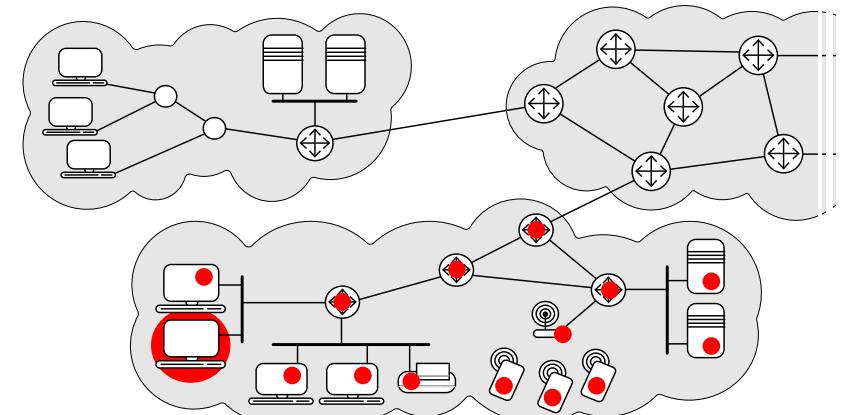
Besoins :

- surveillance du réseau
  - détection de pannes
  - mesure de performance
- intervention sur le matériel
  - activation (interface...)
  - configuration (table de routage...)
- poste de contrôle centralisé

Contraintes :

- matériels hétérogènes
  - routeurs, hubs, switchs...
  - ordinateurs, imprimantes, sondes...
- constructeurs multiples
- localisation géographique distante

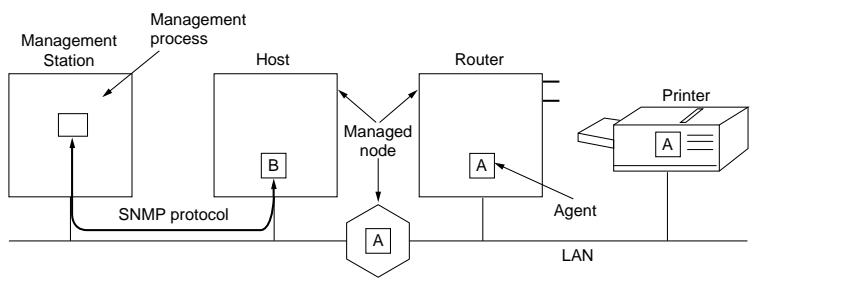
## Equipements administrables



## Administration TCP/IP

Comment gérer les machines en environnement TCP/IP ?

- instrumentation des équipements (**agents**)
- logiciels de supervision (HP Openview, Cisco Works, Nagios...)
- protocole de gestion ➡ **SNMP**



## SNMP : principe

Informations réseau stockées dans deux types de bases :

- **bases agents** (dans les équipements) : Les valeurs sont directement couplées avec les registres internes
- **base centralisée** (plateforme de supervision) : dernières valeurs transmises et historique (statistiques)

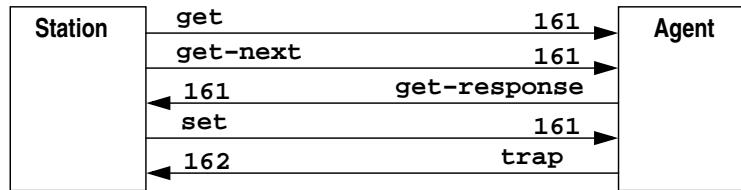
Standardisation (pour échange en milieu hétérogène)

- désignation et type d'information définis par des **MIB**
- structures communes et nomenclature définies dans la **SMI**
- représentation des données en **ASN.1**
- protocole **SNMP** entre la station et les agents permettant :
  - **lecture/écriture** de variables sur des éléments gérés
  - **alarmes** non sollicitées
  - **parcours** de listes de variables dans les éléments gérés ➡ **vision agrégée globale**

## SNMP : commandes

La richesse est dans la MIB !

- seulement 5 commandes **simples**
- utilisation sur **UDP port 161 et 162**



## SNMP : SMI (*Structure for Management Information*)

- les info. respectent les types de la SMIV1 (RFC 1155 et 1212)

NULL	pas de valeur
INTEGER	entier signé non limité
Counter	entier positif (0 à $2^{32} - 1$ ) bouclant
Gauge	entier positif (0 à $2^{32} - 1$ ) borné
TimeTicks	durée en centième de secondes
OCTET STRING	chaine d'octets non limitée
DisplayString	chaine codée en NVT de 255 car. max.
IpAddress	chaine de 4 octets
PhyAddress	chaine de 6 octeys
OBJECT ID.	identifiant numérique...
SEQUENCE	structure d'éléments nommés
SEQUENCE OF	vecteur d'éléments identiques



## SNMP : format des messages

version	communauté	type PDU	ident req.	erreur status	erreur index	nom	valeur	nom	valeur	...
---------	------------	----------	------------	---------------	--------------	-----	--------	-----	--------	-----

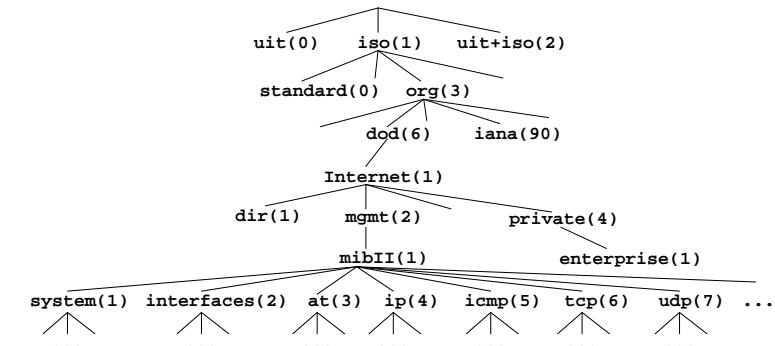
- version : version SNMP - 1 (0 ~ SNMPv1)
- communauté : chaîne de caractères autorisant l'accès
  - généralement "public"
- type PDU : 0 (get), 1 (get-next), 2 (get-response), 3 (set)
  - le message de type 4 (trap) sera présenté dans la suite...
- ident. req. : fait correspondre requêtes et réponses
- erreur status et erreur index : type d'erreur concernant la variable référencée par l'indexage (0 ~ pas d'erreur)
- nom et valeur : variables transportées

Les tailles des champs ne sont pas précisées car la structure du message est décrite en ASN.1 avec encodage BER.



## OID (*Object IDentifier*)

- arbre de "nommage" (référencement **unique** d'un objet)
  - les objets de l'Internet commencent par 1.3.6.1.



## SNMP : MIB (*Management Information Base*)

- les groupes d'objets définis dans la MIB II (RFC 1213) :

```
1.3.6.1.2.1.1    system
1.3.6.1.2.1.2    interfaces
1.3.6.1.2.1.3    at
1.3.6.1.2.1.4    ip
1.3.6.1.2.1.5    icmp
1.3.6.1.2.1.6    tcp
1.3.6.1.2.1.7    udp
1.3.6.1.2.1.8    egp
1.3.6.1.2.1.10   transmission
1.3.6.1.2.1.11   snmp
```

- d'autres groupes, ou sous-groupes sont définis (autres RFC) :

```
1.3.6.1.2.1.17   bridge
1.3.6.1.2.1.43   printer ...
```

Ces groupes contiennent des variables **simples** ou **tables**

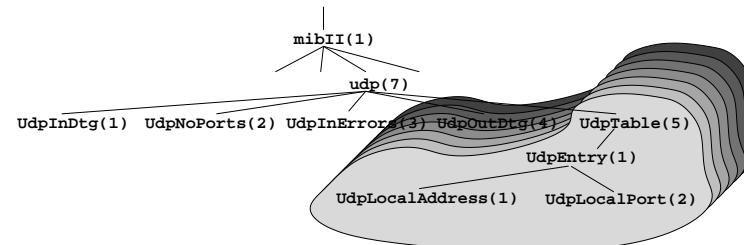


## MIB : variable table

Dans le groupe UDP, 1 variable table :

- udpTable indique les ports scrutés sur l'équipement
- udpTable est un **vecteur** de structures udpEntry

```
udpLocalAddress  IpAddress  ro  adresse IP locale
udpLocalPorts    [0..65535]  ro  port correspondant
```



- l'**index** dans la table est ici

udpLocalAddress.udpLocalPorts

- l'index est précisé à la conception de la MIB

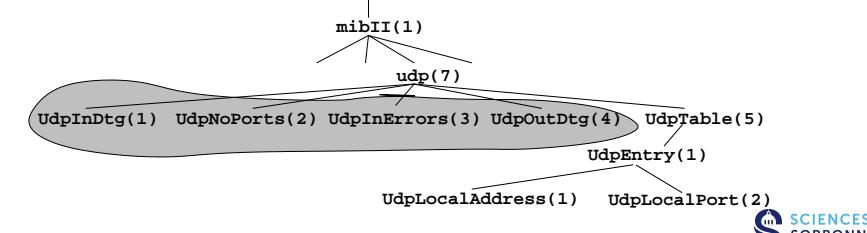


## MIB : variable simple

Dans le groupe UDP, 4 variables simples :

- la MIB II fait correspondre des types SMI

udpInDatagrams	Counter	ro	nb datagrammes délivrés aux applications
udpNoPorts	Counter	ro	nb datagrammes sans application en attente
udpInErrors	Counter	ro	nb datagrammes non délivrables
udpOutDatagrams	Counter	ro	nb datagrammes émis



## SNMP : référencement des variables

Référencement des variables :

- simples : ajout de ".0" à la fin
- tables : ajout des valeurs des champs index
- parcours des OID de la table dans l'ordre **lexicographique**

nom abrégé	OID	valeur
udpInDatagrams.0	1.3.6.1.2.1.7.1.0	17625
udpLocalAddress.0.0.0.0.53	1.3.6.1.2.1.7.5.1.1.0.0.0.0.53	0.0.0.0
udpLocalAddress.0.0.0.0.161	1.3.6.1.2.1.7.5.1.1.0.0.0.0.161	0.0.0.0
udpLocalPort.0.0.0.0.53	1.3.6.1.2.1.7.5.1.2.0.0.0.0.53	53
udpLocalPort.0.0.0.0.161	1.3.6.1.2.1.7.5.1.2.0.0.0.0.161	161

- le référencement permet de spécifier les objets dans les messages UDP
  - seuls les OID et les valeurs sont transportées



## SNMP : commande get-next

Opérateur de parcours dans l'ordre **lexicographique** des OIDS :

- renvoie la prochaine référence terminale
  - `get-next udp` ➔ `udpInDatagrams.0 = 17625`
- permet le parcours des variables...
  - `get-next udpInDatagrams.0` ➔ `udpNoPorts.0 = 0`
- ... et des tables
  - `get-next udpTable`  
 ➔ `udpLocalAddress.0.0.0.0.53 = 0.0.0.0`
  - `get-next udpLocalAddress.0.0.0.0.53`  
 ➔ `udpLocalAddress.0.0.0.0.161 = 0.0.0.0`
  - `get-next udpLocalAddress.0.0.0.0.161`  
 ➔ `udpLocalPort.0.0.0.0.53 = 53 ...`
- fin du tableau lors du changement de nom :
  - `get-next udpLocalPort.0.0.0.0.161`  
 ➔ `snmpInPkts.0 = 12`



## Syntaxe abstraite ASN.1

Couche 6 de l'OSI (définie par l'UIT, recommandation X.680)

- propriétés :
  - représentation universelle d'informations
  - type associé aux données
  - désignation par un identificateur unique (OID)
  - notation de type BNF
- description des informations échangées par SNMP :

```
RFC1157-SNMP DEFINITIONS ::= BEGIN
  Message ::= SEQUENCE {
    version     INTEGER {version-1(0)},
    community   OCTET STRING,
    data        ANY
  }
  PDUs ::= CHOICE {
    get-request  GetRequest-PDU,
    get-next-request GetNextRequest-PDU,
    get-response  GetResponse-PDU,
    set-request   SetRequest-PDU,
    trap         Trap-PDU
  }...
```



## SNMP : Trap

Envoi d'un message SNMP de l'agent vers l'admin. sur le **port 162**

version	communauté	type = 4	entreprise	adr. agent	type trap	code entr.	estamp. temp.	nom	valeur	...
---------	------------	----------	------------	------------	-----------	------------	---------------	-----	--------	-----

- entreprise : identificateur du créateur de l'agent
  - OID débutant par 1.3.6.1.4.1.
- adr. agent : adresse IP de l'agent

0	coldStart	agent initialisé
1	warmStart	agent réinitialisé
2	linkDown	interface désactivée
3	linkUp	interface activée
...		
6	entr. specific	voir le champ code entr.

- type trap :
- code entr. : sous-code du trap spécifique à l'entreprise
- estamp. temp. : valeur indiquant le nombre de centièmes de secondes depuis le démarrage de l'agent



## ASN.1 : PDU

Message get écrit en ASN.1 :

```
getRequest-PDU ::= [0]
  IMPLICIT SEQUENCE {
    request-id  INTEGER,
    error-status INTEGER {
      noError(0), tooBig(1),
      noSuchName(2), badValue(3),
      readOnly(4), genErr(5), -- always 0
    }
    error-index  INTEGER, -- always 0
    variable-bindings SEQUENCE OF
      SEQUENCE {
        name    ObjectName,
        value   ObjectSyntax
      }
  }
```



## SNMP : encodage BER

Encodage **TLV** (Type, Longueur, Valeur)

- types (1o) : les 2 bits de poids fort déterminent la catégorie

• UNIVERSAL (00)	0x02 INTEGER
	0x04 OCTET STRING
	0x05 NULL
	0x06 OBJECT IDENTIFIER
	0x30 SEQUENCE
• APPLICATION (01)	0x40 IpAddress
	0x41 Counter
	0x42 Gauge
	0x43 TimeTicks

- CONTEXT (10)
- PRIVATE (11)
- longueur des données (1 octet si < 0x80, sinon norme X.208)
  - longueur 49 ➔ 0x31, longueur 242 ➔ 0x8200F2...

- données (valeur)
  - les OID (avec les valeurs entières successives A.B.C.D...) sont codés en octets avec les 2 premiers agrégés : A\*40 + B, C, D

## SNMP : exemple

```
0020          30 82 00 f2 02 01 J...D... .0.....
0030 00 04 06 70 75 62 6c 69 63 a2 82 00 e3 02 01 01 ...publi c.....
0040 02 01 00 02 01 00 30 82 00 d6 30 82 00 0d 06 08 .....0. ..0.....
0050 2b 06 01 02 01 02 01 00 02 01 03 30 82 00 0f 06 +..... .0....
0060 0a 2b 06 01 02 01 02 02 01 08 01 02 01 01 30 82 .+..... ....0.
0070 00 0f 06 0a 2b 06 01 02 01 02 02 01 08 02 02 01 ....+.... .....
0080 02 ...
0100          ... 30 82 00 10 ..... C.,,0...
0110 06 0a 2b 06 01 02 01 02 02 01 09 01 43 02 01 2c ..+.... .C...
```

## MIB RMON

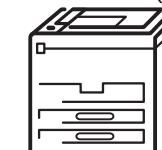
Remote MONitoring (RFC 2819 - STD 59)

Sonde pour obtenir des **statistiques** sur un réseau administré

- 9 groupes :
  - statistiques sur Ethernet (table de 21 attributs)
  - équipements du réseau (adresses observées...)
  - matrice de statistiques (entre deux stations)
  - capture de trames
  - ...
- nombreuses extensions
  - identification de protocoles pour RMON (RFC 2895, 2896)
  - RMON pour réseaux commutés (SMON : RFC 2613)
  - gestion des interface pour RMON (IFTOPN : RFC 3144)
  - RMON pour les services différenciés (DSMON : RFC 3287) ...

## Autres MIB IETF (1)

MIB Imprimante Printer MIB (RFC 1759 - RFC 3805)



- 274 Objets (228 OID dont 16 tables)
  - 20 groupes :
    - groupe général
    - groupe des entrées
    - groupe des sorties
    - groupe des dimensions de sortie
    - groupe de la couverture
    - groupe des fournitures
    - groupe des colorants ...

## Autres MIB IETF (2)

RFC1230 : IEEE 802.4 Token Bus MIB  
 RFC1381 : MIB Extension for X.25 LAPB  
 RFC1559 : DECnet Phase IV MIB Extensions  
 RFC1593 : SNA APPN Node MIB  
 RFC1611 : DNS Server MIB Extensions  
 RFC1612 : DNS Resolver MIB Extensions  
 RFC1696 : Modem MIB  
 RFC1697 : Relational DB Mngmnt System MIB  
 RFC1724 : RIP Version 2 MIB  
 RFC1748 : IEEE 802.5 MIB  
 RFC2020 : IEEE 802.12 Interface MIB  
 RFC2320 : Classical IP and ARP Over ATM MIB  
 RFC2564 : Application Management MIB  
 RFC1792 : TCP/IPX Connection MIB  
 RFC2605 : Directory Server Monitoring MIB  
 RFC2707 : Job Monitoring MIB  
 RFC2720 : Traffic Flow Measurement : Meter MIB  
 RFC2788 : Network Services Monitoring MIB  
 RFC2789 : Mail Monitoring MIB  
 RFC2790 : Host Resources MIB  
 RFC2863 : The Interfaces Group MIB  
 RFC2922 : Physical Topology MIB  
 RFC2932 : IPv4 Multicast Routing MIB  
 RFC2933 : IGMP MIB  
 RFC2934 : PIM MIB for IPv4  
 RFC2981 : Event MIB  
 RFC2982 : Distributed Management Expression MIB  
 RFC3014 : Notification Log MIB  
 RFC3144 : RMon MIB Extensions for Interface  
 RFC3287 : RMon MIB Extensions for DiffServ...

RFC6765 : xDSL Multi-Pair Bonding (G.Bond) MIB  
 RFC6766 : xDSL Multi-Pair Bonding TDIM MIB  
 RFC6767 : Ethernet-Based xDSL Multi-Pair Bonding MIB  
 RFC6768 : ATM-Based xDSL Bonded Interfaces MIB  
 RFC6779 : Definition of Managed Objects for the NDP  
 RFC6825 : TE Database MIB for MPLS-TE/GMPLS  
 RFC6933 : Entity MIB (Version 4)  
 RFC7052 : Locator/ID Separation Protocol (LISP) MIB  
 RFC7124 : Ethernet in the 1st Mile Copper (EFMCu) MIB  
 RFC7147 : Definitions of Managed Objects for iSCSI  
 RFC7184 : Def. of Managed Objects for the OLSRv2  
 RFC7330 : Def. of TCs for BFD  
 RFC7331 : BFD MIB  
 RFC7367 : Def. for the MANET Simplified Multicast  
 RFC7388 : Def. for 6LoWPANs  
 RFC7420 : PCEP MIB Module  
 RFC7453 : MPLS-TP TE MIB  
 RFC7460 : Monitoring & Control MIB for Power & Energy  
 RFC7461 : Energy Object Context MIB  
 RFC7577 : Def. of Managed Objects for Battery Monit.  
 RFC7659 : Def. of Managed Objects for NATs  
 RFC7666 : MIB for VMs Controlled by a Hypervisor  
 RFC7697 : MPLS-TP OAM Identifiers MIB  
 RFC7784 : TRILL OAM MIB  
 RFC7856 : Software Mesh MIB  
 RFC7870 : Dual-Stack Lite MIB for AFTRs  
 RFC7939 : Def. of Managed Objects for NDR  
 RFC8038 : Exporting MIB Variables Using INFLX Proto  
 RFC8150 : MPLS Transport Profile Linear Proto

Olivier Fourmaux (olivier.fourmaux@sorbonne-universite.fr)

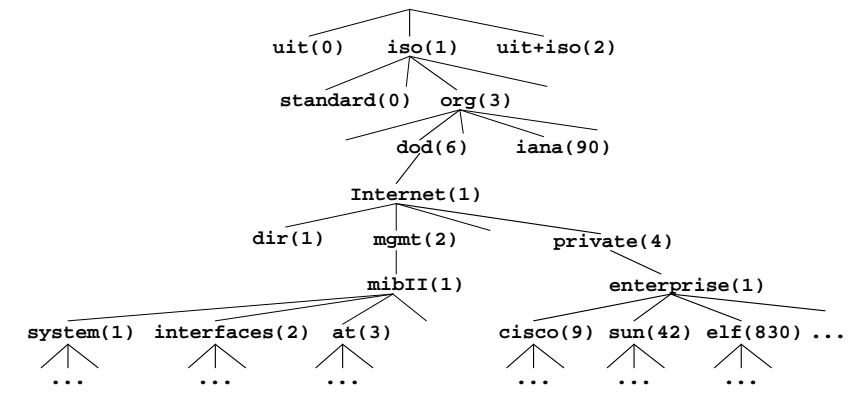
Architecture des Réseaux (ARes) 2/5 : Application

## SNMP : versions

Plusieurs versions ont été standardisées :

- **SNMPv1** définie dans le RFC 1157 (1990) simple et non sécurisée ➡ encore très utilisée
- **SNMPv2** définie dans les RFC 1901 à 1908 avec extensions (requêtes get-bulk et inform, MIB SNMPv2 et SNMPv2-M2M) et sécurisation mais pas de consensus des industriels
  - **SNMPv2c** réduite aux nouvelles fonctionnalités mais sans la sécurité (*Community-Based*)
  - **SNMPv2u** nouveau mécanisme de sécurité simplifié (*User-Based*)
- **SNMPv3** définie dans les RFC 3410 à 3418, réintègre la sécurité
  - seule la v3 est un standard IETF (STD-62)
  - Utilisation de multi-version : RFC 3584

## MIB constructeur



## SNMP : limitations

- la mesure ne doit pas perturber le réseau
- latence
- MIB propriétaires
- sécurité
  - écoute sur le réseau (*packet sniffing*) pour connaître la communauté
  - usurpation d'identité (*IP spoofing*) facilité par UDP

➡ améliorations avec **SNMPv3**