M1 RES - Architecture des réseaux 3/10
Application (2)

... DNS, SNMP et Peer-to-peer

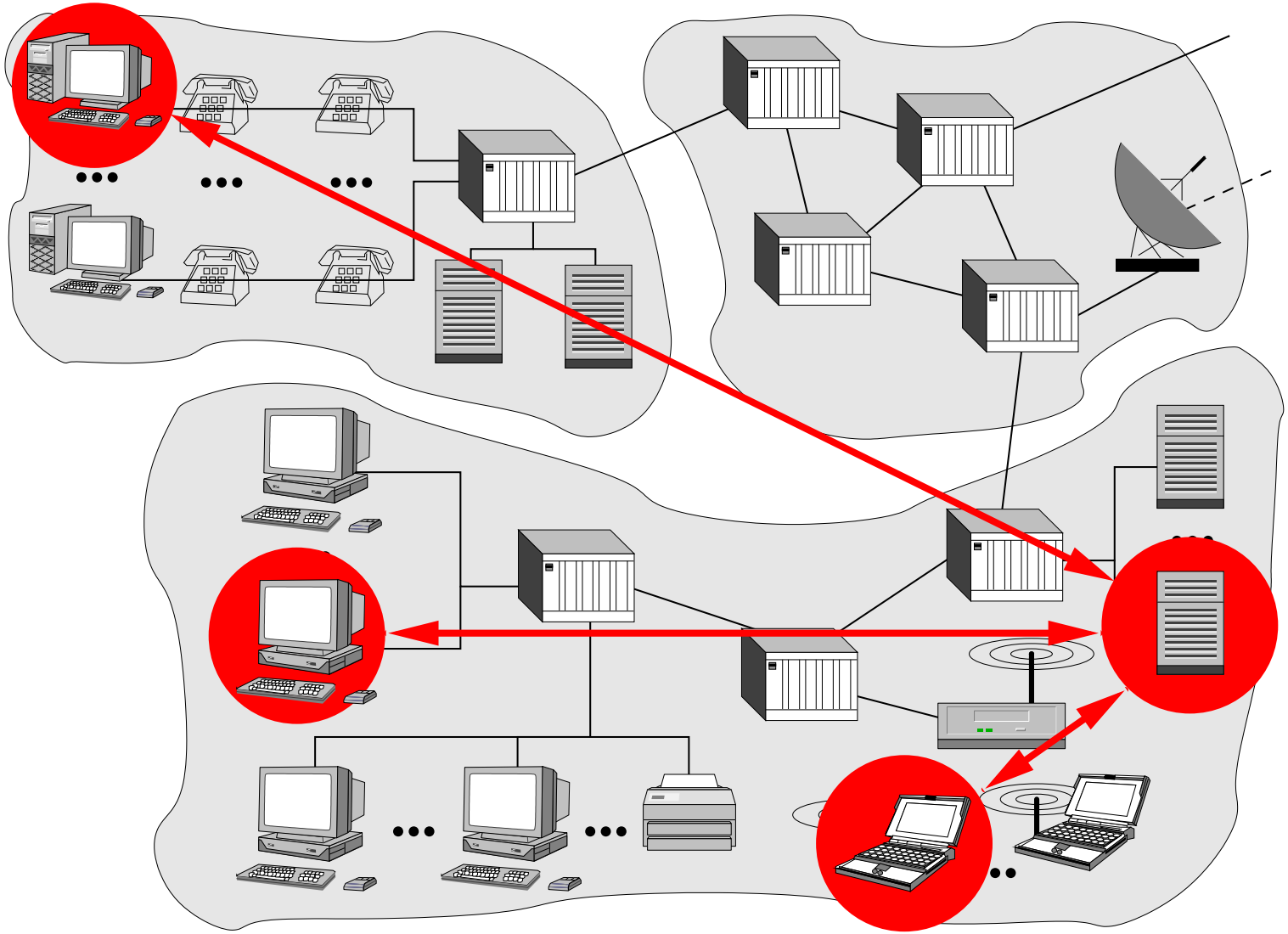
Olivier Fourmaux

olivier.fourmaux@lip6.fr

Version 4.c, septembre 2004

UNIVERSITE
PIERRE & MARIE CURIE
LA SCIENCE APARIS

Applications



Couche Application

Définition :

La couche application gère les logiciels utilisateurs (applications) en s'appuyant les services de bout-en-bout définis dans les couches de niveau inférieur.

Cours précédent :

- connexion à distance
- tranfert de fichiers
- messagerie électronique
- *World Wide Web*

➠ comment nommer les machines

➠ comment administrer

Plan

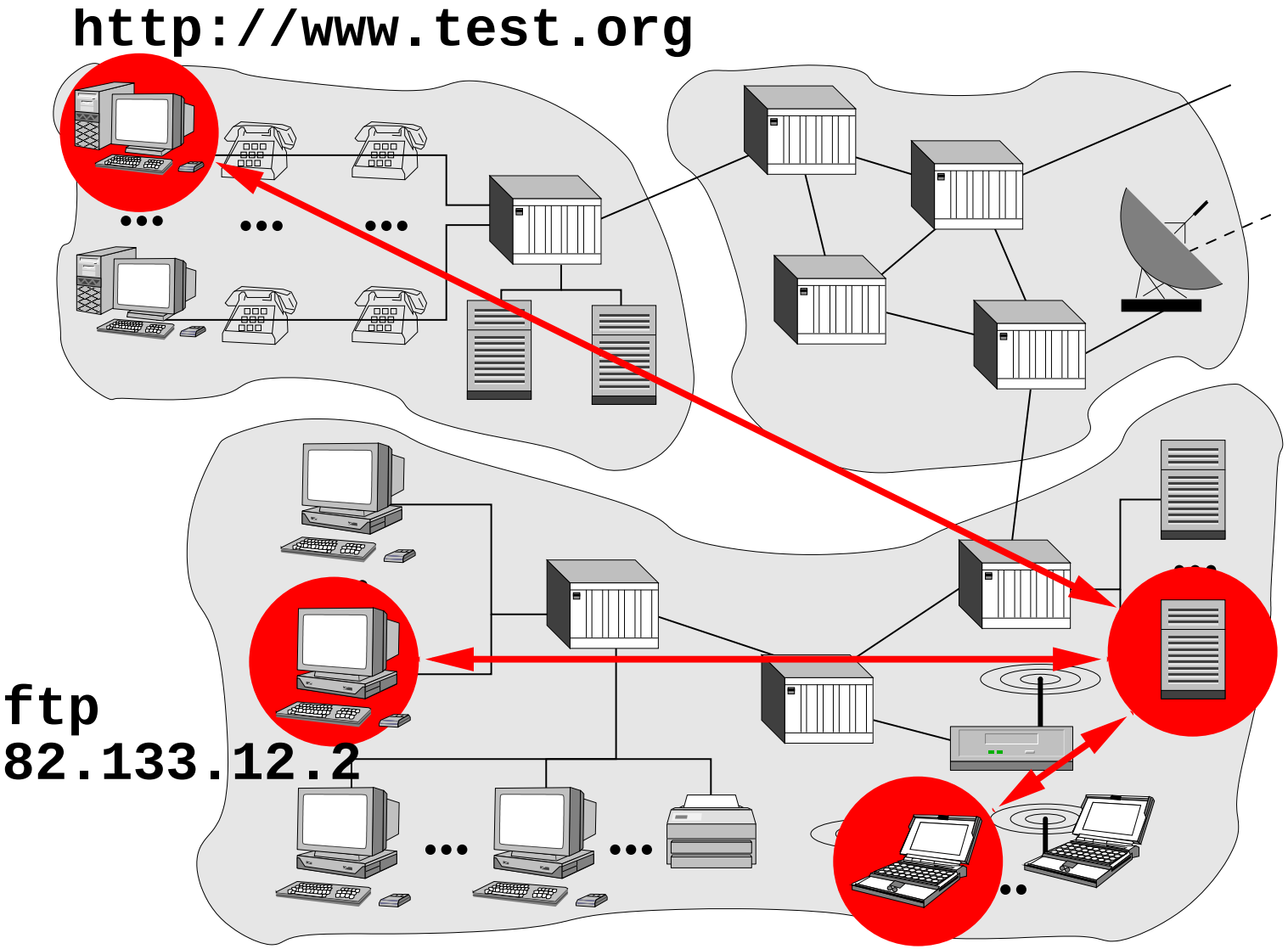
...

Annuaire

Administration

Peer-to-peer

Correspondance noms – adresses



Annuaire

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

- dans l'Internet :
  - ✓ initialement
    - ☞ **un** fichier géré par le *NIC*
    - ☞ "nommage" à plat
  - ✓ actuellement : **DNS**
    - ☞ système de "nommage" hiérarchique
    - ☞ base de données **distribuée**
    - ☞ protocole d'échange
    - ☞ contrôlé par l'*InterNIC* et ses délégués

DNS : principe

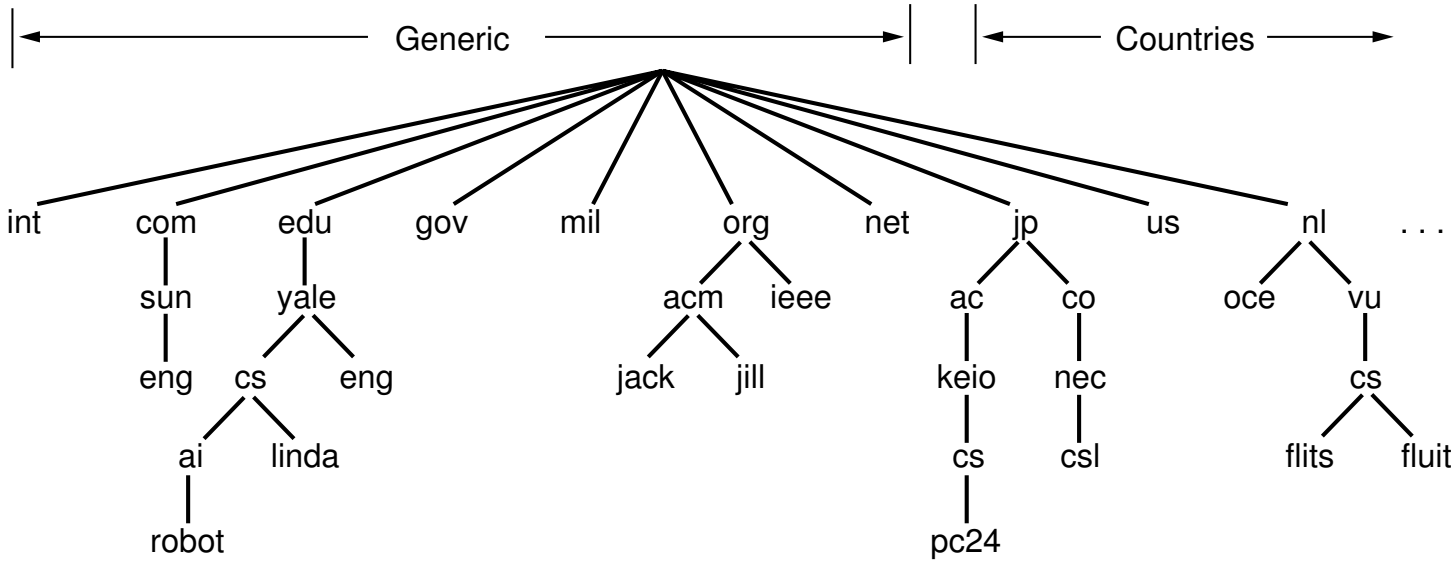
Domain Name System

- annuaire standard de l'Internet (RFC 1034 et RFC 1035)
- **serveurs de noms** (serveurs DNS)
  - ✓ composants de la **hiérarchie** supportant la b.d. 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 disribution...*
- exemple :
  - ✓ BIND (*Berkeley Internet Name Domain*)
    - ☞ `named` sous Unix

DNS : Espace de "nommage"

Système de "nommage" hiérarchique

- structure arborescente ($\sim$ système de fichier Unix)
- label d'un nœud : 63 caractères (majuscules ou minuscules)
- **domain name** = liste des labels en parcourant l'arbre vers la racine ("**.**" séparateur) :
  - ✓ relatif : `pc24.CS.keio`
  - ✓ absolu (**FQDN**) : `pc24.CS.keio.ac.jp.`



DNS : TLD

Top Level Domain

- ICANN (*Internet Corporation for Assigned Names and Numbers*)
  - ✓ partage du premier niveau et délégation à des *registrars*

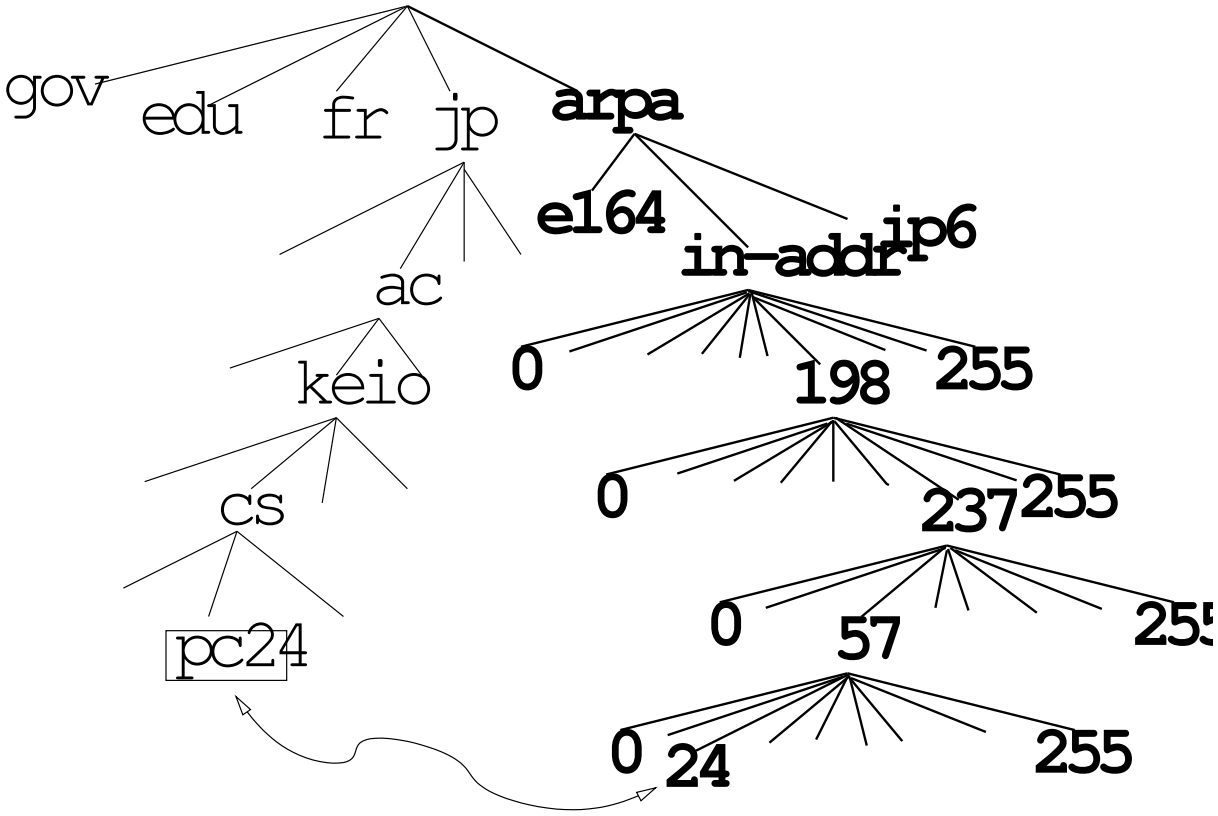
gTLD	intro.	description	operator
.aero	2001	Air-transport ind. *	SITA
.biz	2001	Businesses	NeuLevel
.com	–	Unrestricted	VeriSign
.coop	2001	Cooperative *	DotCooperation
.edu	–	US educ. inst. *	EDUCAUSE
.gov	–	US government *	US Admin.
.info	2001	Unrestricted	Afilias
.int	1998	Int. organisations	ICANN
.mil	–	US military *	US DoD NIC
.museum	2001	Museums *	MuseDoma
.name	2001	Individuals	GNR
.net	–	Unrestricted	VeriSign
.org	–	Unrestricted	PIR
.pro	2001	Professionals	RegistryPro
.arpa	–	**special TLD**	ICANN

ccTLD ISO 3166	240 countries and external territories
.ac	Ascension Island
.ad	Andorra
.ae	United Arab Emirates
.af	Afghanistan
...	...
.fr	France
...	...
.uk	United Kingdom (gb)
.us	United States
...	...
.yu	Yugoslavia
.za	South Africa
.zm	Zambia
.zw	Zimbabwe

DNS : Domaine arpa.

Résolution : `pc24.cs.keio.ac.jp.` ➠ ?

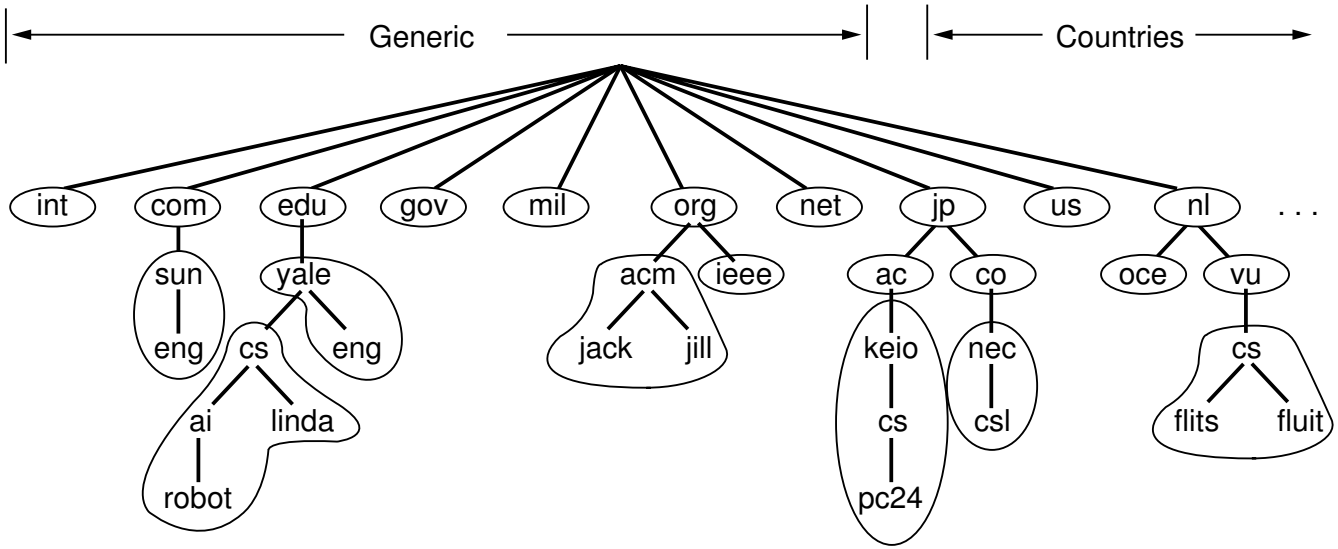
Résolution **inverse** : `24.57.237.198.in-addr.arpa.` ➠ ?



DNS : Zones

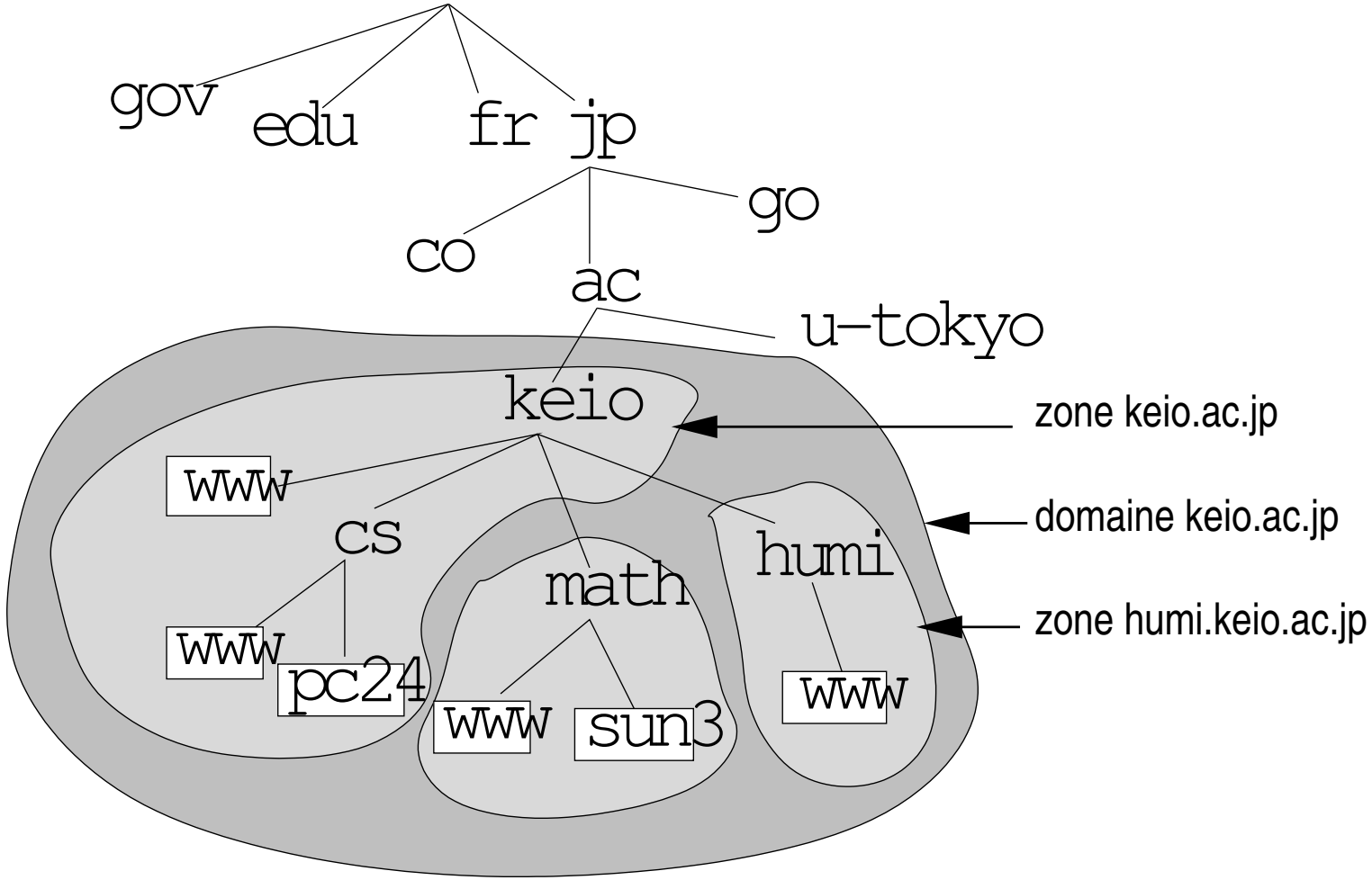
Sous-arbre de l'arbre DNS administré séparément

- ($\sim$ partitions physiques d'un système de fichier Unix)
- délégation des noms de sous-domaines correspondants
  ✓ exemple : `keio.jp.ac.`
- des **serveurs de noms** y sont associés



DNS : Zones

Ne pas confondre zone et domaine !

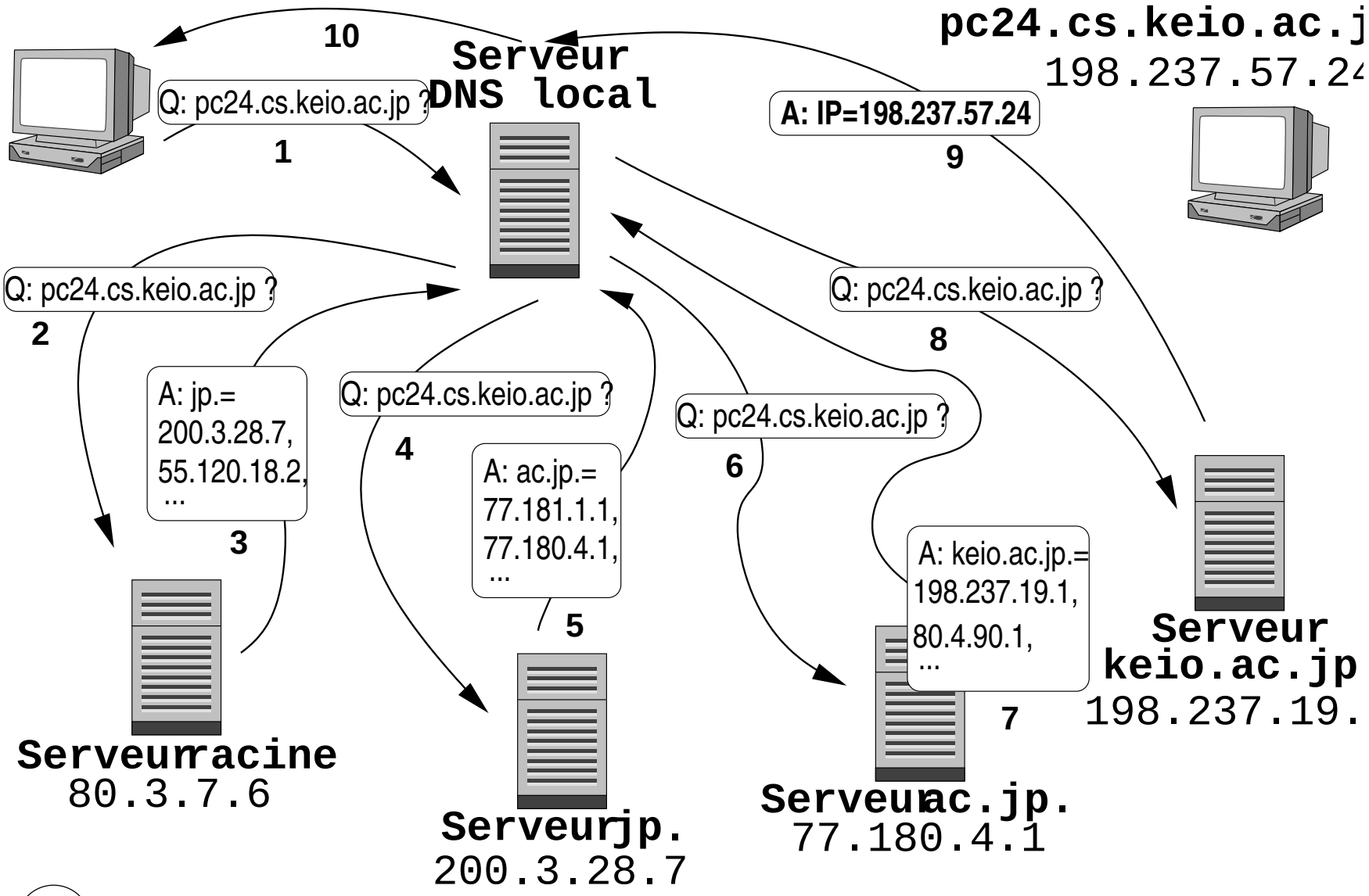


DNS : Serveur de noms

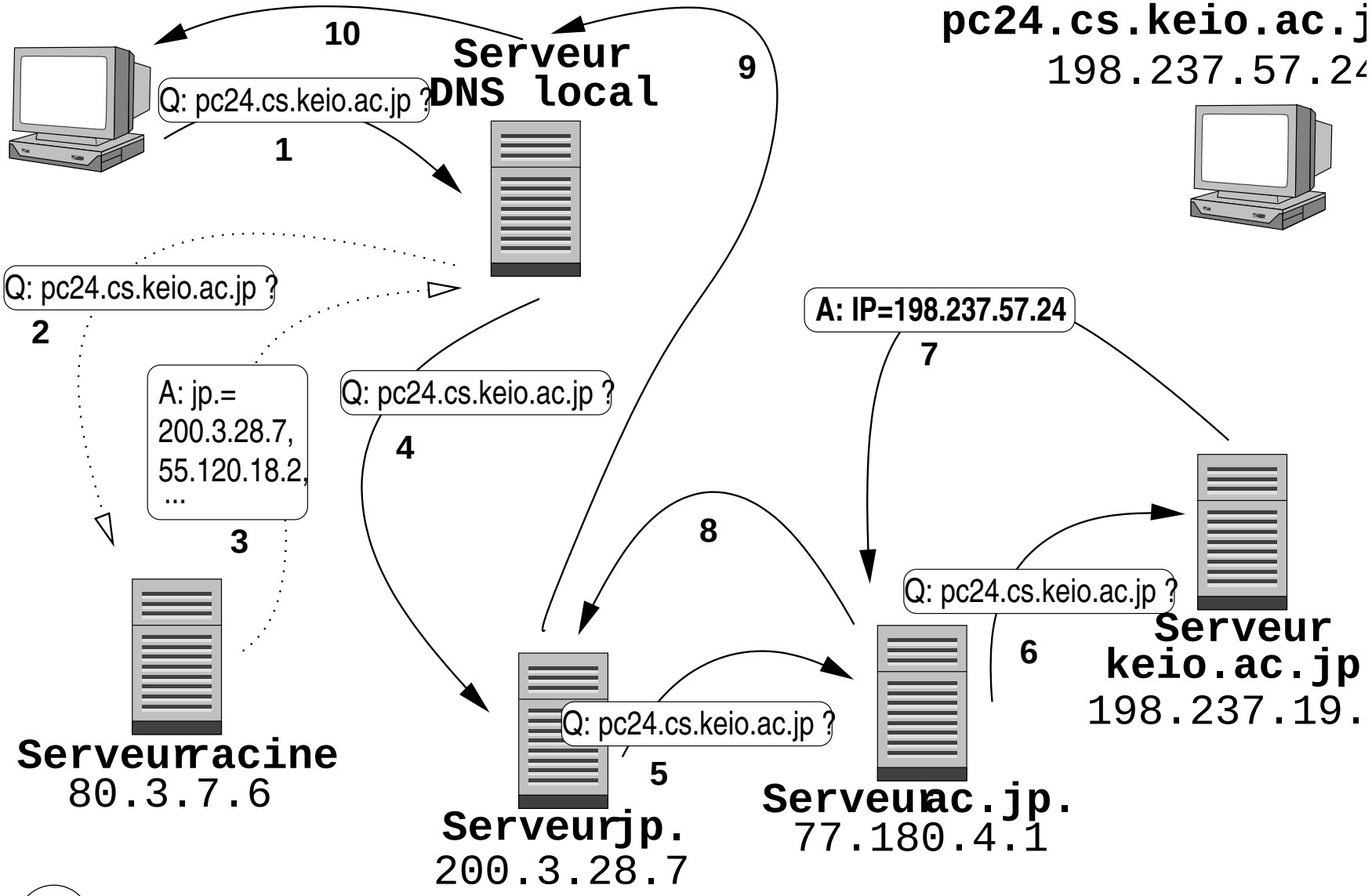
Différents types de serveurs de noms

- dans une zone :
  - ✓ un **primaire** (*primary name server*)
    - ☞ informations autoritaires sur la zone (*authoritative reccords*)
    - ☞ initialisation locale (disque)
  - ✓ un ou plusieurs **secondaire** (*secondary name server*)
    - ☞ redondance : indépendants du primaire (voire éloignés)
    - ☞ initialisation et m-à-j. à partir du primaire (**transfert de zone**)
- hors de sa zone :
  - ✓ 13 serveurs racines (*root name server*)
    - ☞ 1 primaire et 12 secondaires, haute disponibilité (anycast)
    - ☞ config. en dur (`ftp.rs.internic.net/domain/named.root`)
  - ✓ requêtes **récursives** ou **itératives**
  - ✓ utilisation de caches

DNS : Requête itérative

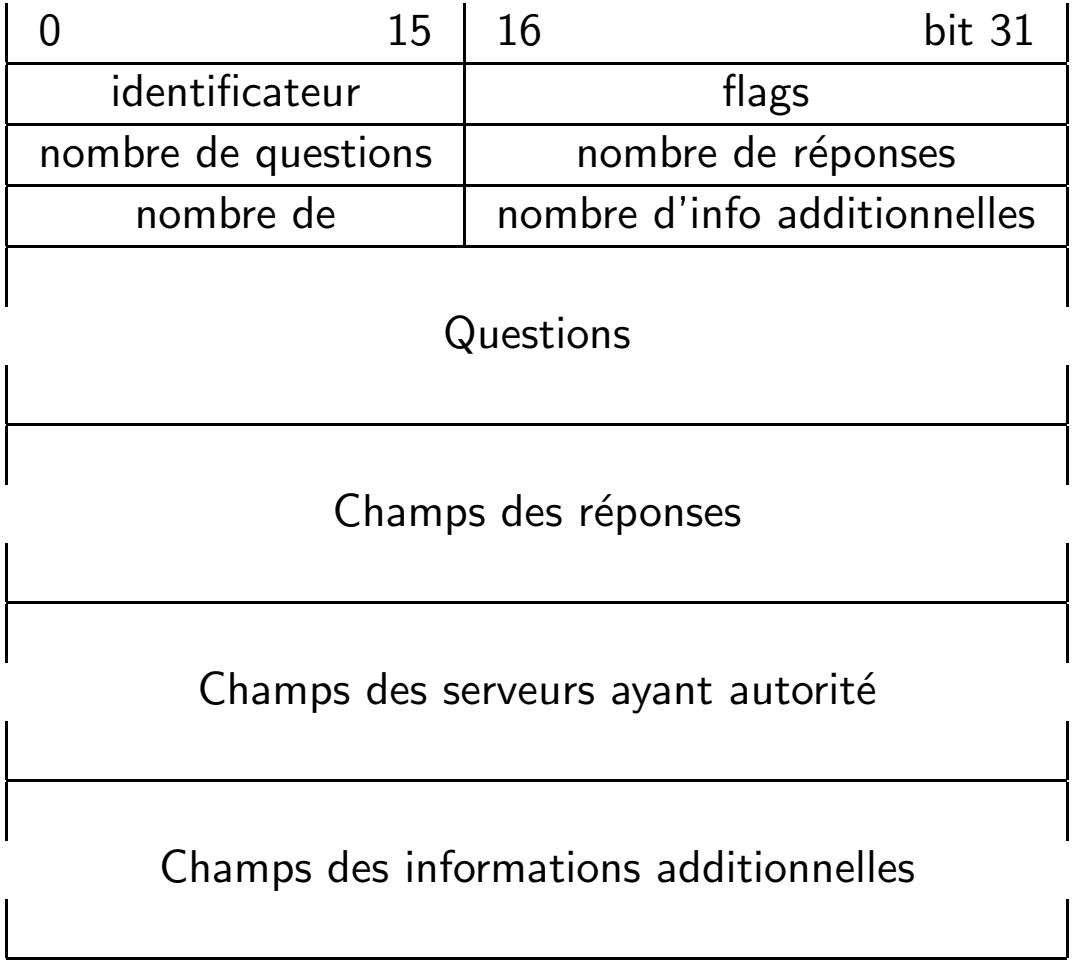


DNS : Requête récursive



DNS : Format du message (1)

Pour les questions et les réponses :

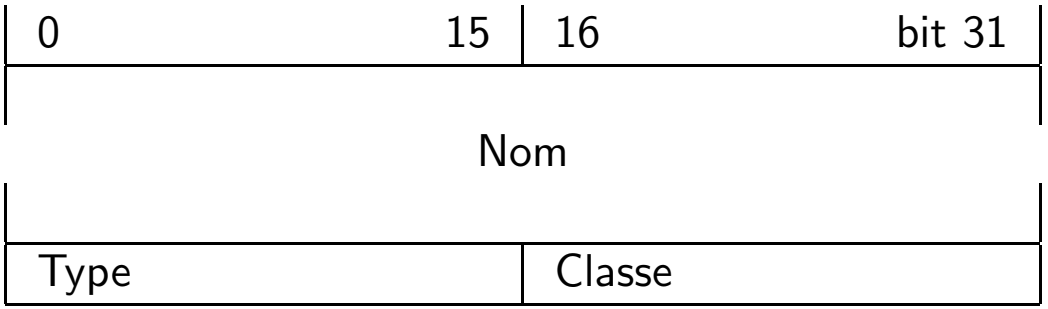


flags :

- **QR** (1 bit) : 0 = question, 1 = réponse
- **opcode** (4 bit) 0 = standard ...
- **AA** (1 bit) : 1 = réponse autoritaire
- **TC** (1 bit) : 1 = tronqué (datagramme UDP $<$ 512o)
- **RD** (1 bit) : 1 = demande récursion (indiqué par le client)
- **RA** (1 bit) : 1 = récursion disponible (indiqué par le serveur)
- **réservé** (3 bits) : 000
- **rcode** (4 bits) : 0 = pas d'erreur... 3 = erreur de nom...

DNS : Format du message (2)

Une question :



- **Nom** : N octets, chaque nom de label est précédé par un octet indiquant le nombre de caractères ASCII le suivant[1]. Terminé par 0x00[2].

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**	adresse IPv4	13	**HINFO**	info sur l’équipement
2	**NS**	nom de serveur	15	**MX**	serveur de messagerie
5	**CNAME**	alias	28	**AAAA**	adresse IPv6
6	**SOA**	zone DNS gérée		...	
12	**PTR**	pointeur de nom	255	*	tous les types (quest.)

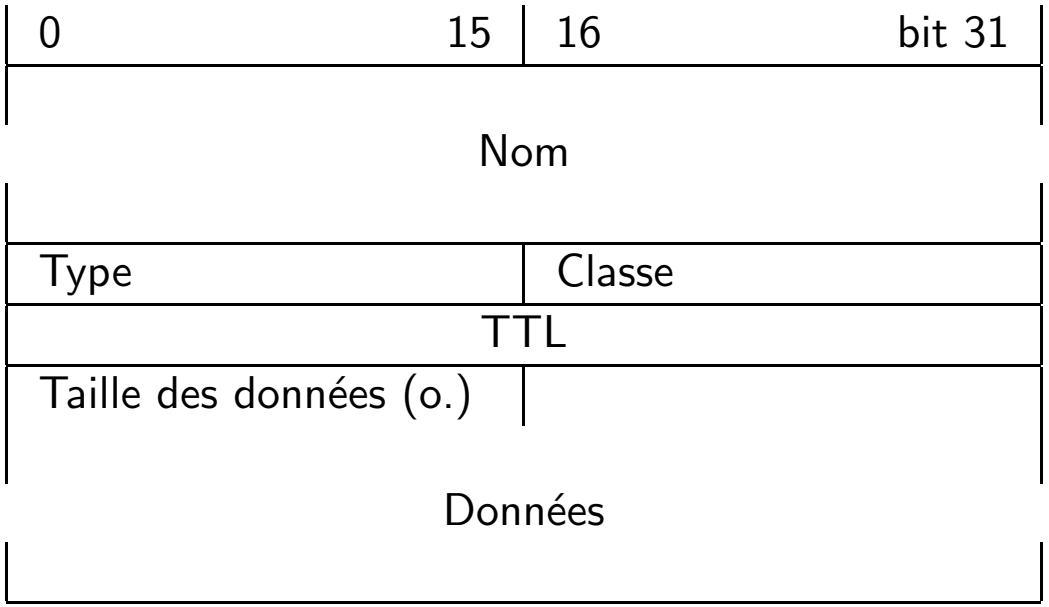
- **Classe** (16 bits) : 1 = Internet

[1] Si cet octet est > 63, alors renvoi : `0xC0 n` = le label se trouve à `n` octets du début du message DNS

[2] Pas de bourrage si les N octets ne sont pas aligné sur 4 octetss (32 bits)

DNS : Format du message (3)

Un champ réponse :



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

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
```

Plan

...

Annuaire

Administration

Peer-to-peer

Administration de réseau

Développement du réseau (nombreux équipements et machines à gérer)

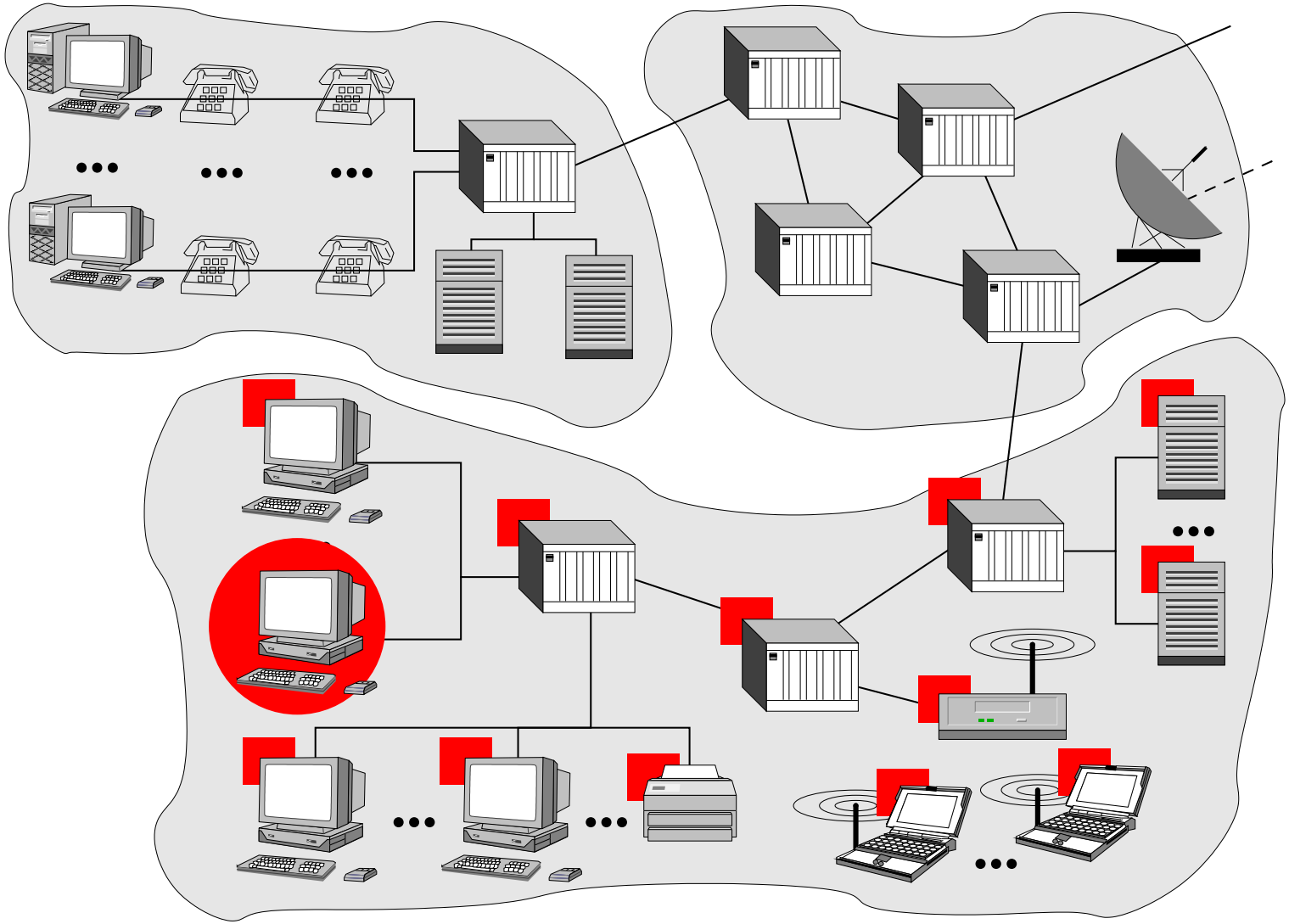
Besoins :

- 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

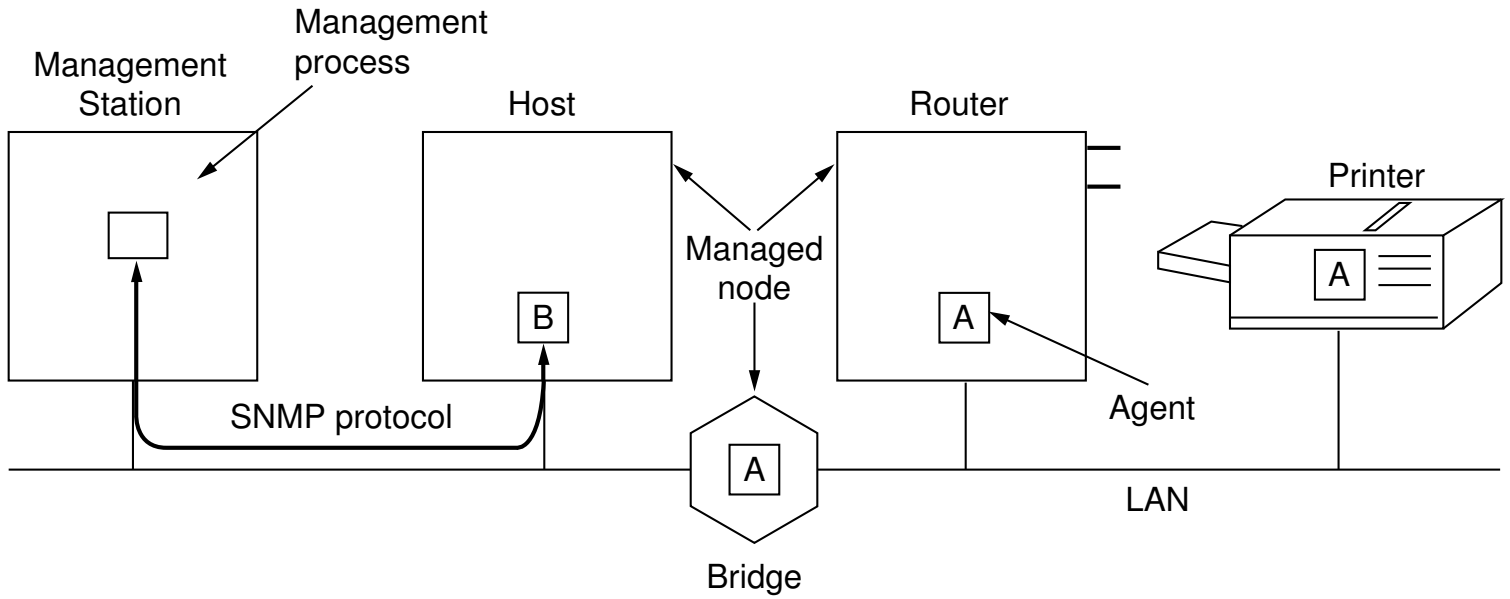
Equipements administrables



Administration TCP/IP

Comment gérer les machines en environnement TCP/IP ?

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



pictures from TANENBAUM A. S. *Computer Networks 3rd edition*

SNMP : principe

Informations réseau stockées dans deux types de bases :

- **bases agent** (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)

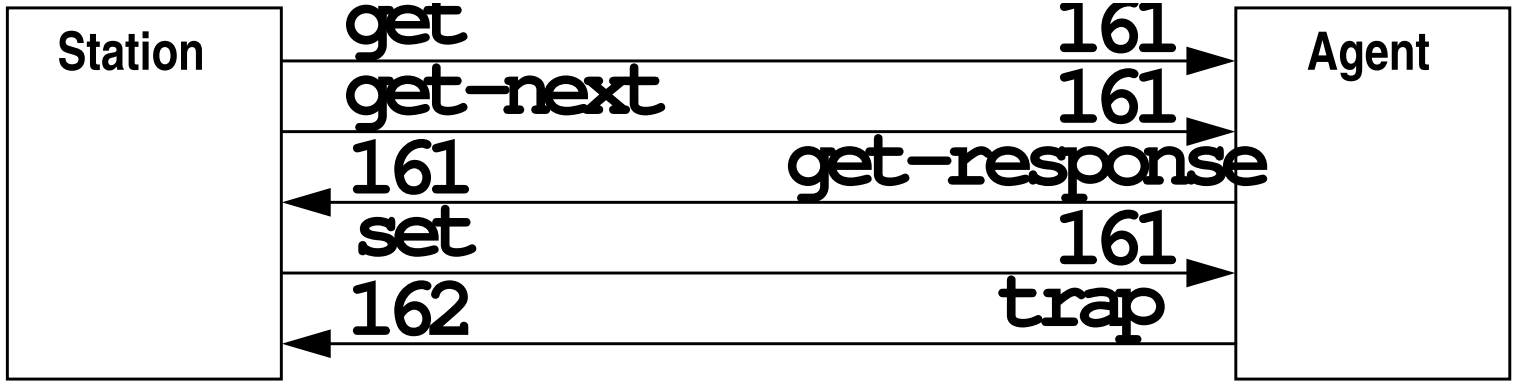
- désignation et type d'information définis par des **MIB**
- structures communes et nomenclature définies dans la **SMI**
- représentation des données en **ASN.1**
- protocole **SNMP** entre la station et les agents permettant :
  - ✓ **lecture/écriture** de variables sur des éléments gérés
  - ✓ **alarmes** 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 : Format des messages

- `version` : version SNMP - 1 (0 $\sim$ SNMPv1)
- `communauté` : chaine de caractères autorisant l'accès
  - ✓ généralement "`public`"
- `type PDU` : 0 (`get`), 1 (`get-next`), 2 (`set`), 3 (`get-response`)
  - ✓ le message de type 4 (`trap`) sera présenté dans la suite...
- `ident. req.` : permet de faire correspondre requêtes et réponses
- `erreur status` et `erreur index` : indique le type d'erreur concernant la variable référencée par l'indexage (0 $\sim$ 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.

SNMP : SMI

Structure for Management Information

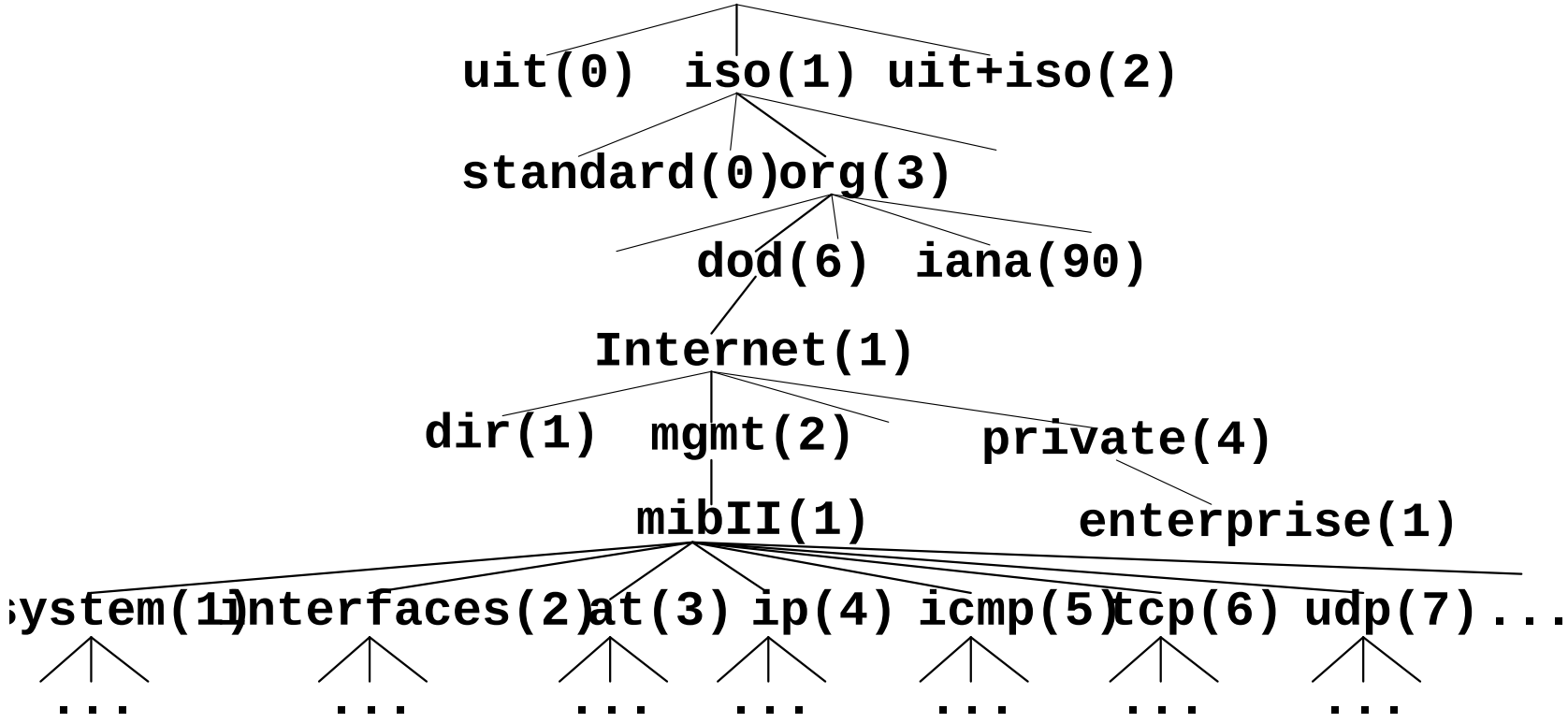
- les informations respectent les types de la SMIv1 (RFC 1155 et 1212)

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

OID

Object IDentifier

- arbre de nommage (référencement **unique** d'une structure de données)
  - ✓ les objets de l'Internet commencent par 1.3.6.1.



SNMP : MIB

Management Information Base

- les goupes 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
1.3.6.1.2.1.10.15    fddi
```

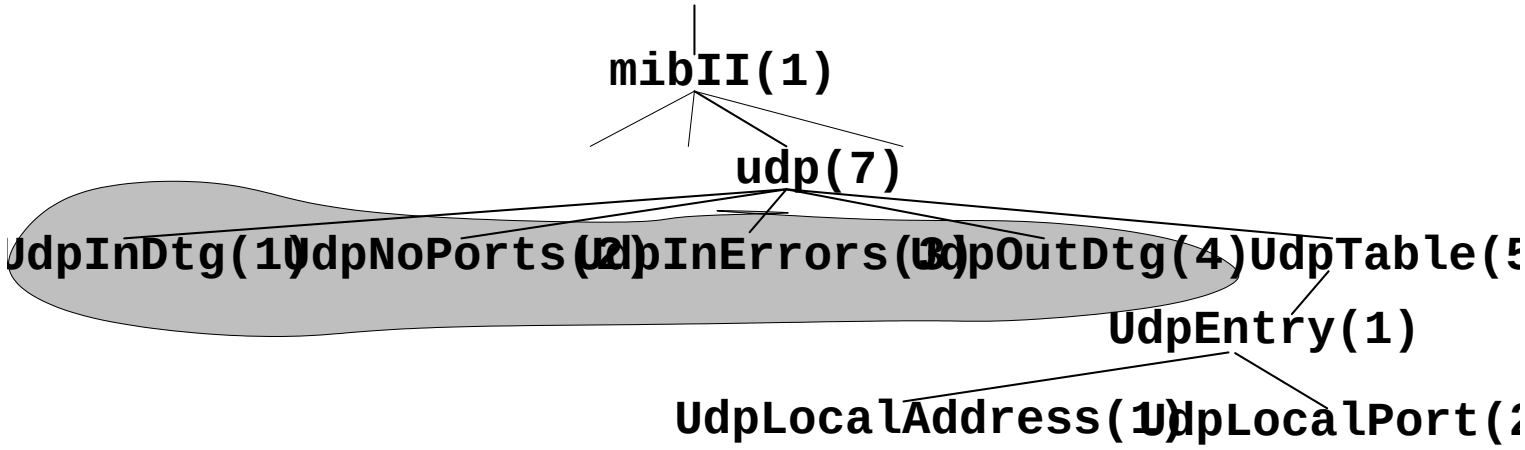
Ces groupes contiennent des variables **simples** ou **tables**

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

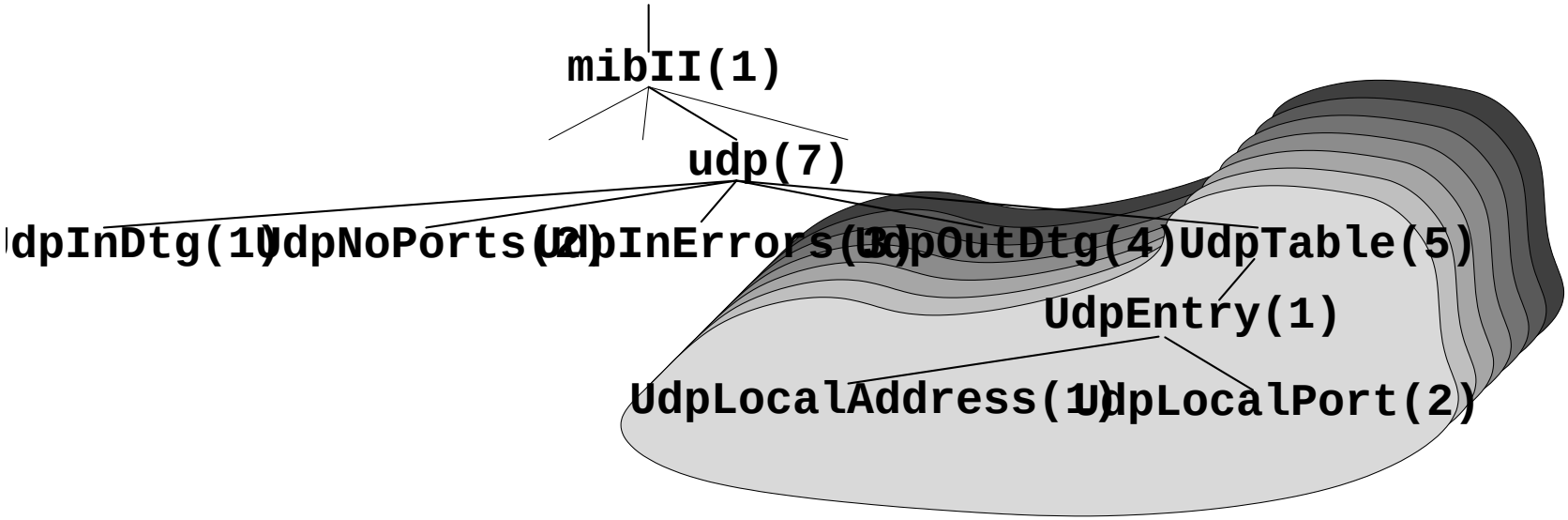


MIB : Variable table

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

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 basé sur 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
    ```

SNMP : Trap

Envoi d'un message SNMP de l'agent vers l'administrateur 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
- `type trap` :

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

- `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

Syntaxe abstraite ASN.1

Abstract Syntax Notation One

- 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
                }...
```

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 (1 octet) : les 2 bits de poids fort déterminent la catégorie
  - ✓ UNIVERSAL (00)

0x02	INTERGER
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 < 128, sinon voir la norme X.208)
- données (valeur)
  - ✓ Les OID (avec les valeurs entières successives A.B.C.D...) sont codés sur des 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...
```

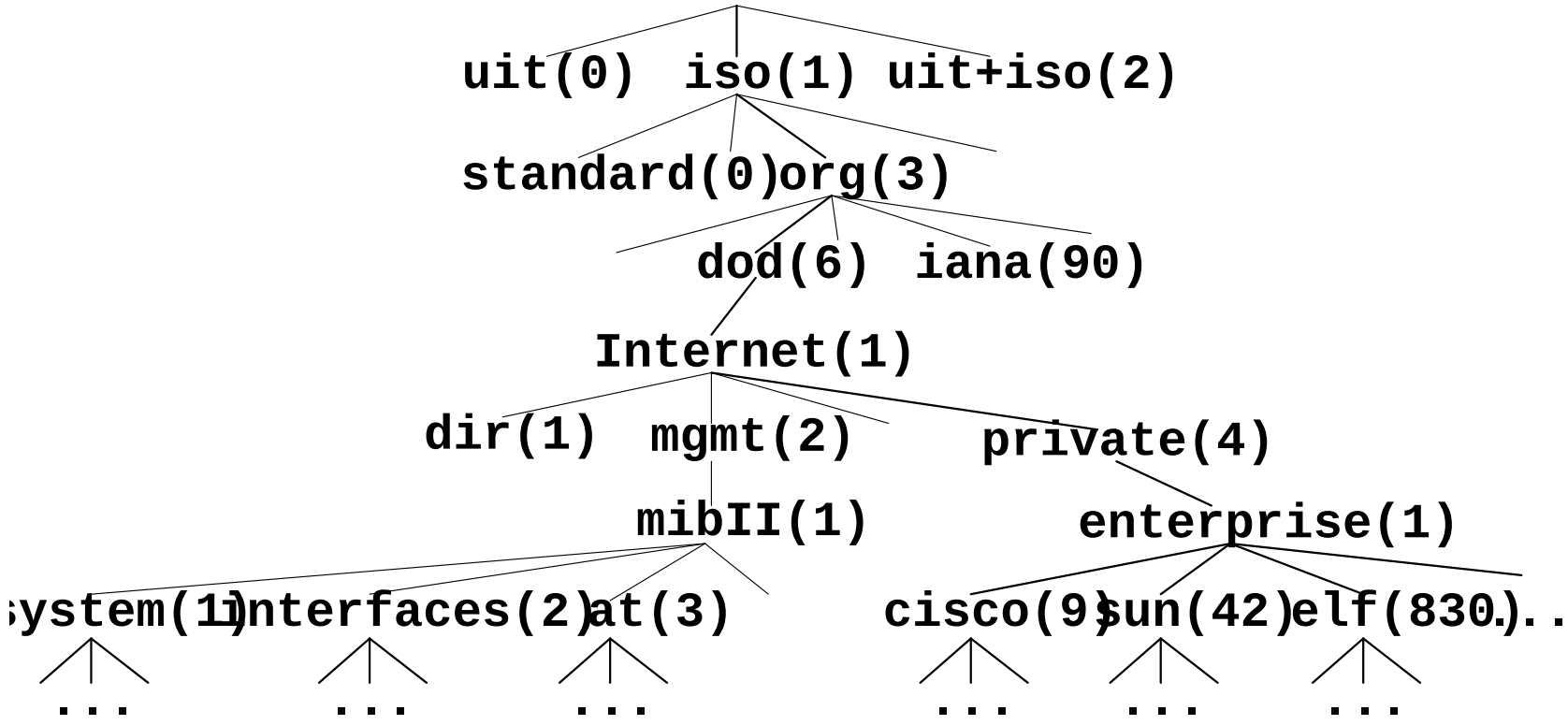
SNMP : MIB RMON

Remote MONitoring

Sonde pour obtenir des **statistiques** sur un réseau local (RFC 1757)

- MIB de taille importante
  - ✓ 9 groupes :
    - ☞ statistiques sur Ethernet (table de 21 attributs)
    - ☞ équipements du réseau (adresses observées...)
    - ☞ matrice de statistique (entre deux stations)
    - ☞ capture de trames
    - ☞ ...

SNMP : MIB constructeur



SNMP : Versions

Plusieurs versions ont été standardisées :

- **SNMP v1** définie dans le RFC 1157 (1990) simple et non sécurisé mais le plus utilisé
- **SNMP v2** 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
  - ✓ **SNMP v2c** réduite aux nouvelles fonctionnalités mais sans la sécurité
- **SNMP V3** définie dans les RFC 2570 à 2576, réintègre la sécurité
  - ➠ Complexe et peu utilisée

SNMP : limitations

- la mesure ne doit pas perturber le réseau
- latence
- MIB propriétaires
- sécurité
  - ➠ quelques améliorations avec **SNMPv3**

Plan

...

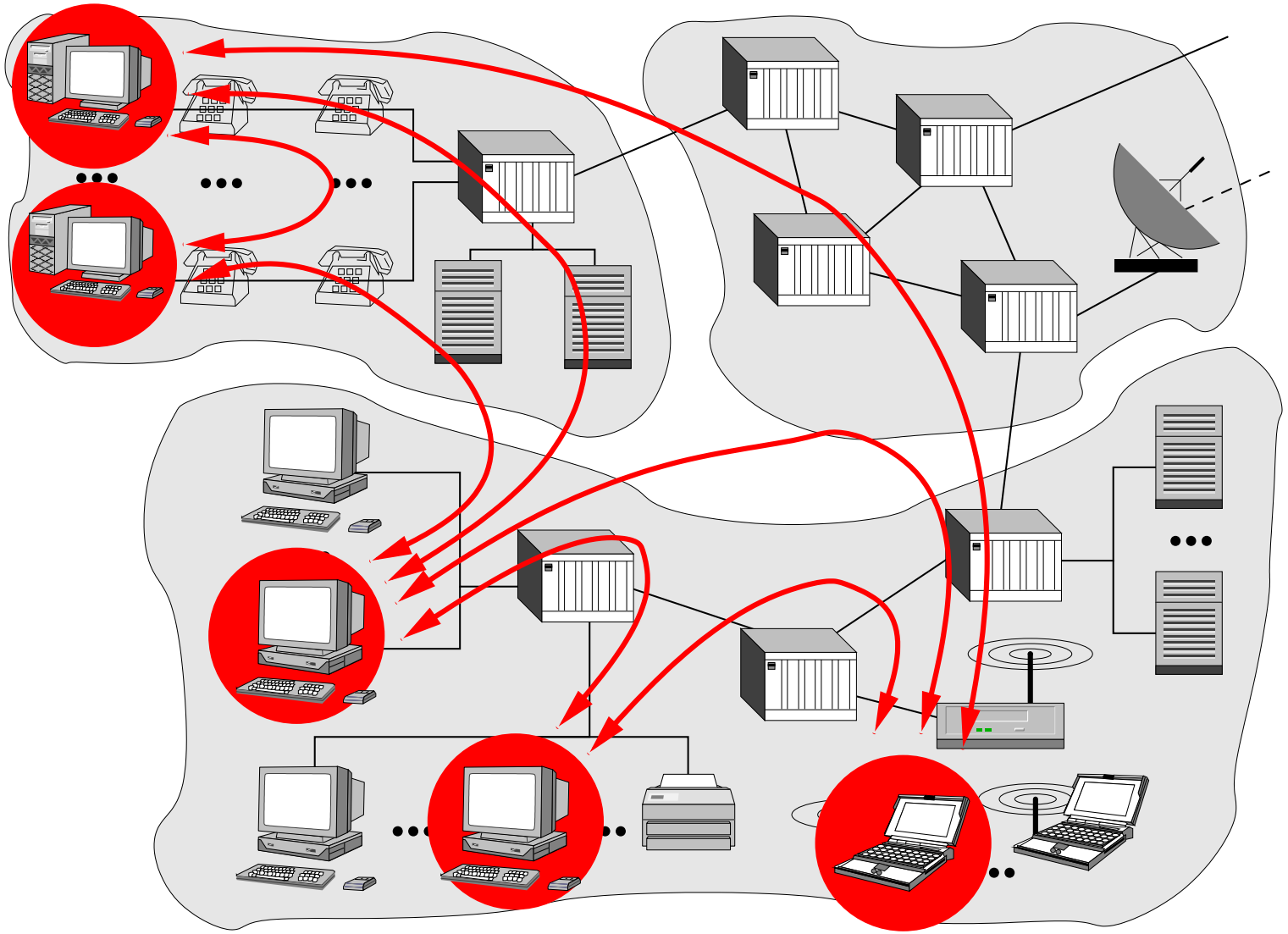
Annuaire

Administration

Peer-to-peer

- Napster
- Gnutella
- Freenet
- ...

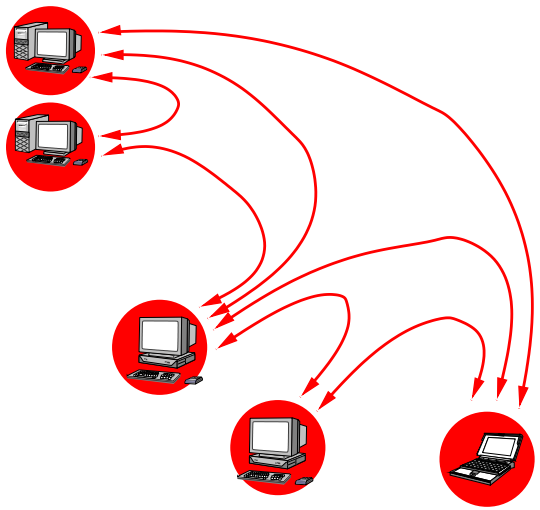
Peer-to-peer



Peer-to-peer

Applications *peer-to-peer*

- **partage de fichiers**
  - ✓ Napster, DirectConnect, FastTrack, eDonkey...
  - ✓ Gnutella, Overnet, Freenet...
- autres services réseau "remontés" au niveau applicatif
  - ✓ réseaux ad-hoc
  - ✓ communications multipoints
    - ☞ domaine de la recherche

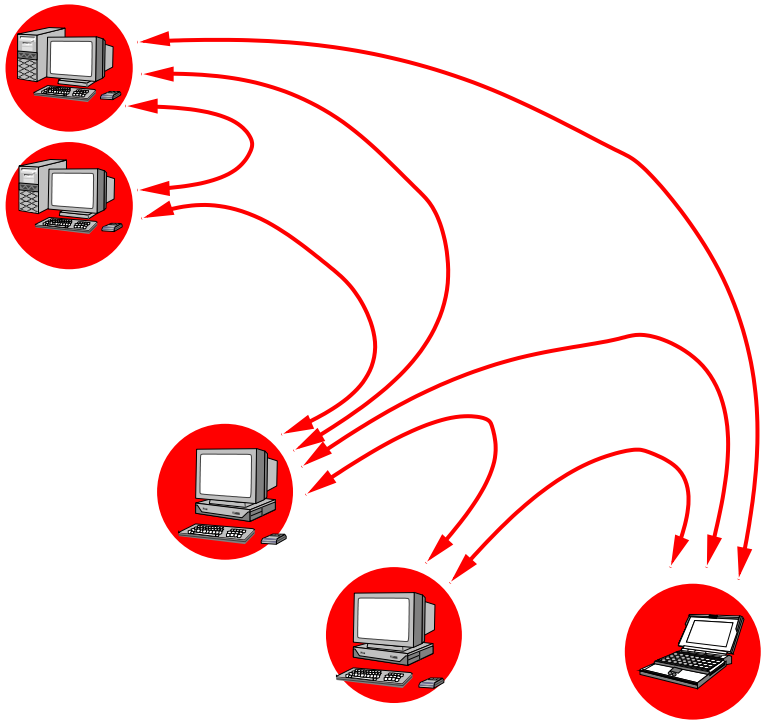


Des standards ?

P2P : Questions

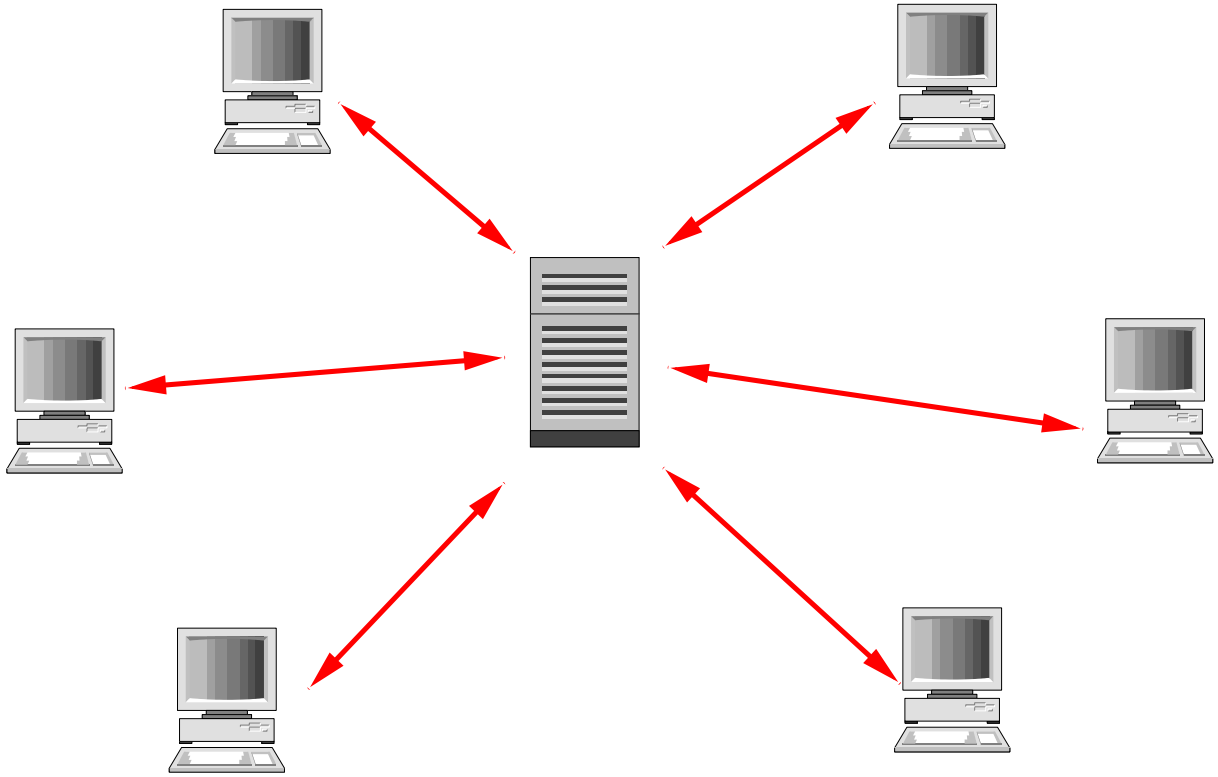
Ne peut-on pas tout faire en client/serveur ?

- est-ce juste du "réseau au niveau applicatif" ?
- quels nouveaux types de services ? d'applications ?
- quels sont les nouveaux challenges techniques ?



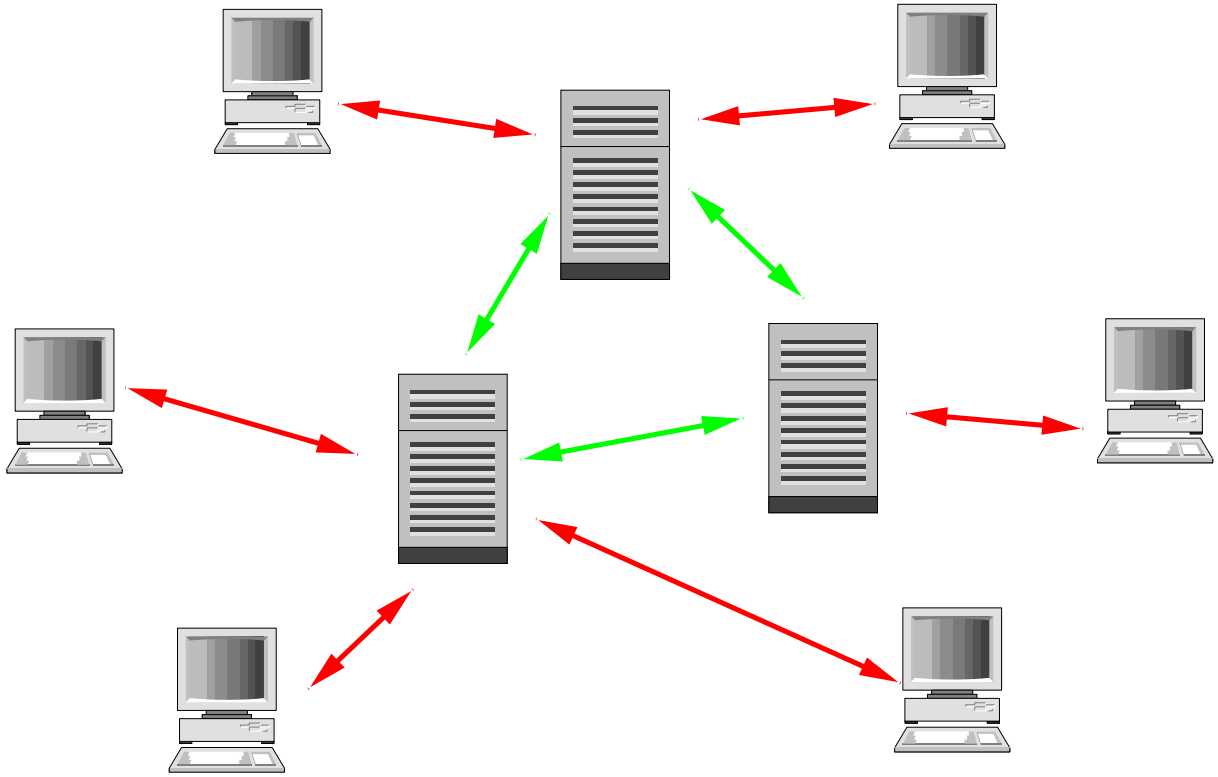
P2P : Architectures (1)

Client/serveur centalisé classique



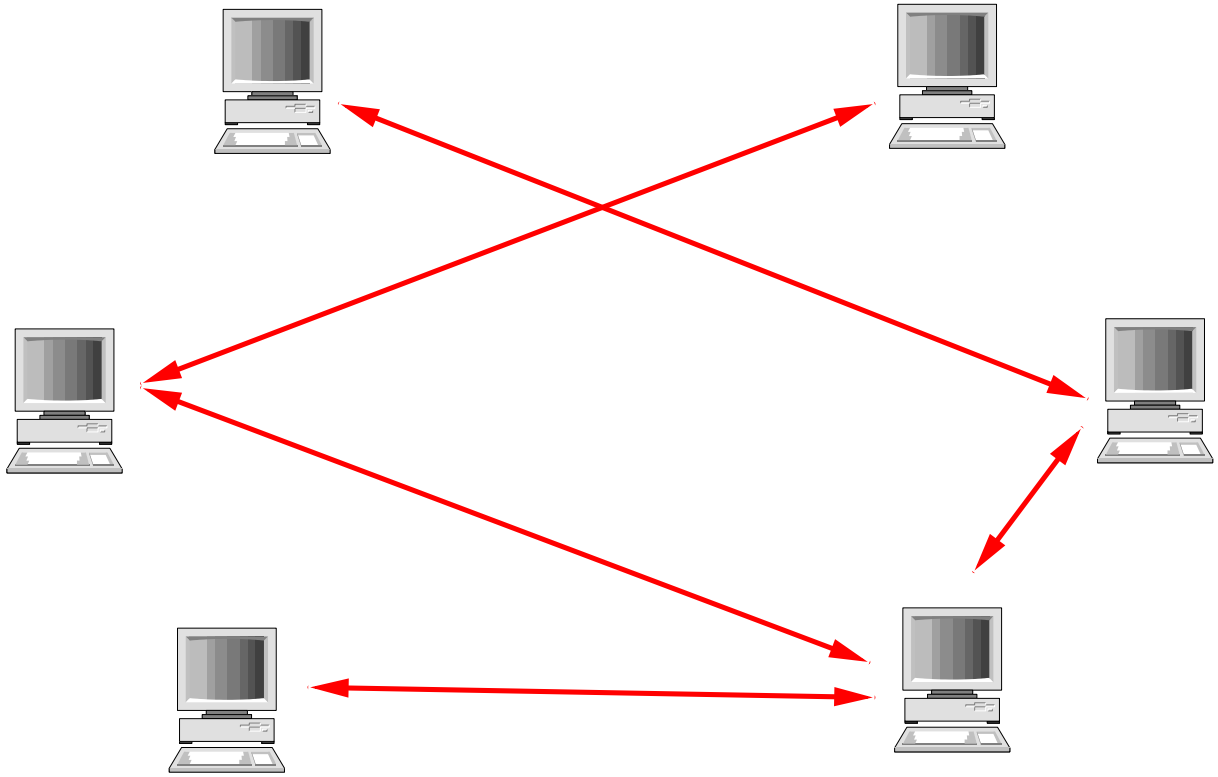
P2P : Architectures (2)

Client/serveur avec réplication des serveurs



P2P : Architectures (3)

Peer-to-peer classique



P2P : Comparaison Client/serveur

RPC/RMI

- synchrones
- asymétriques
- orientés langage
- identification
- authentification

```
Client_call(args)

Server_main_loop()
  while (true)
    await(call)
    switch(call.procid)
      case 0: call.ret=proc0(call.arg)
      case 1: call.ret=proc1(call.arg)
      ...
      default: exception
```

Messages P2P

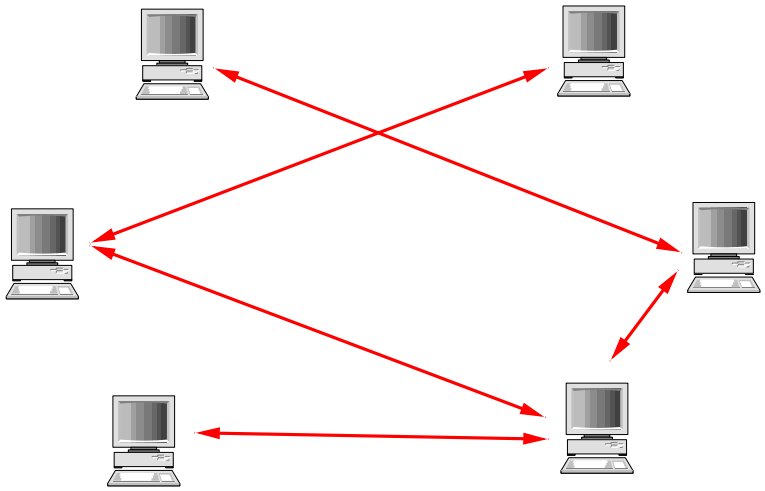
- asynchrones
- symétriques
- orientés service
- anonymat
- haute disponibilité

```
Peer_main_loop()
  while (true)
    await(event)
    switch(event.type)
      case timer_expire:
            do_some_P2P_work()
            randomize_timers()
      case inbound_mesg:
            handle_mesg()
```

P2P : Fonctionnalités

Caractéristiques des systèmes *peer-to-peer*

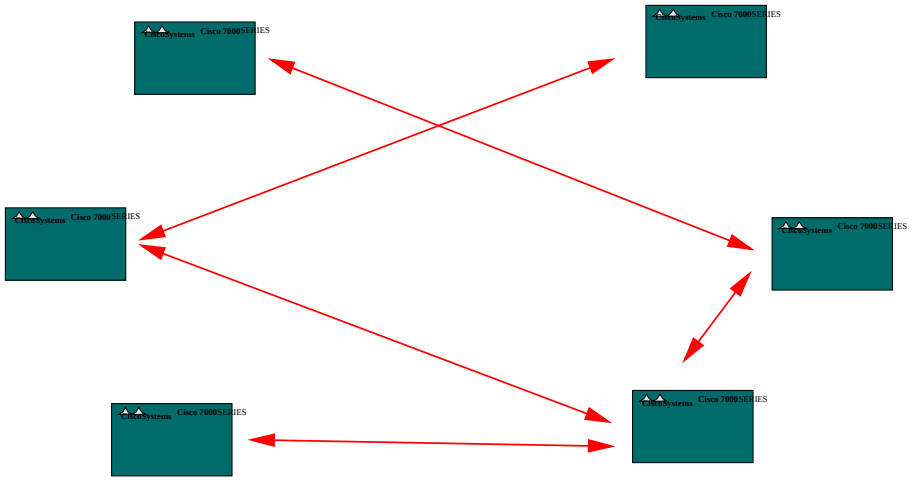
- pas de rôle distinct
  - ✓ évite les points de congestion ou les nœuds défaillants
  - ✓ besoin d'algorithmes distribués
    - ☞ découverte de service (nommage, adressage, calcul de métriques)
    - ☞ maintien d'un état du voisinage
    - ☞ routage au niveau applicatif (lié au contenu)
    - ☞ robustesse,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



P2P : Applications existantes

Le *peer-to-peer* n'est pas nouveau :

- Routeurs IP
  - ✓ découverte de la topologie
  - ✓ maintien de l'état du voisinage
  - ✓ autonome et tolérance aux fautes
  - ✓ algorithme distribué de routage
  - ✓ ...



Plan

...

Annuaire

Administration

Peer-to-peer

- **Napster**
- Gnutella
- Freenet
- ...

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 dérangeante ?
  - ✓ historique
    - ☞ fin 98 : Shawn Fanning (19 ans) débute le developpement
    - ☞ 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...

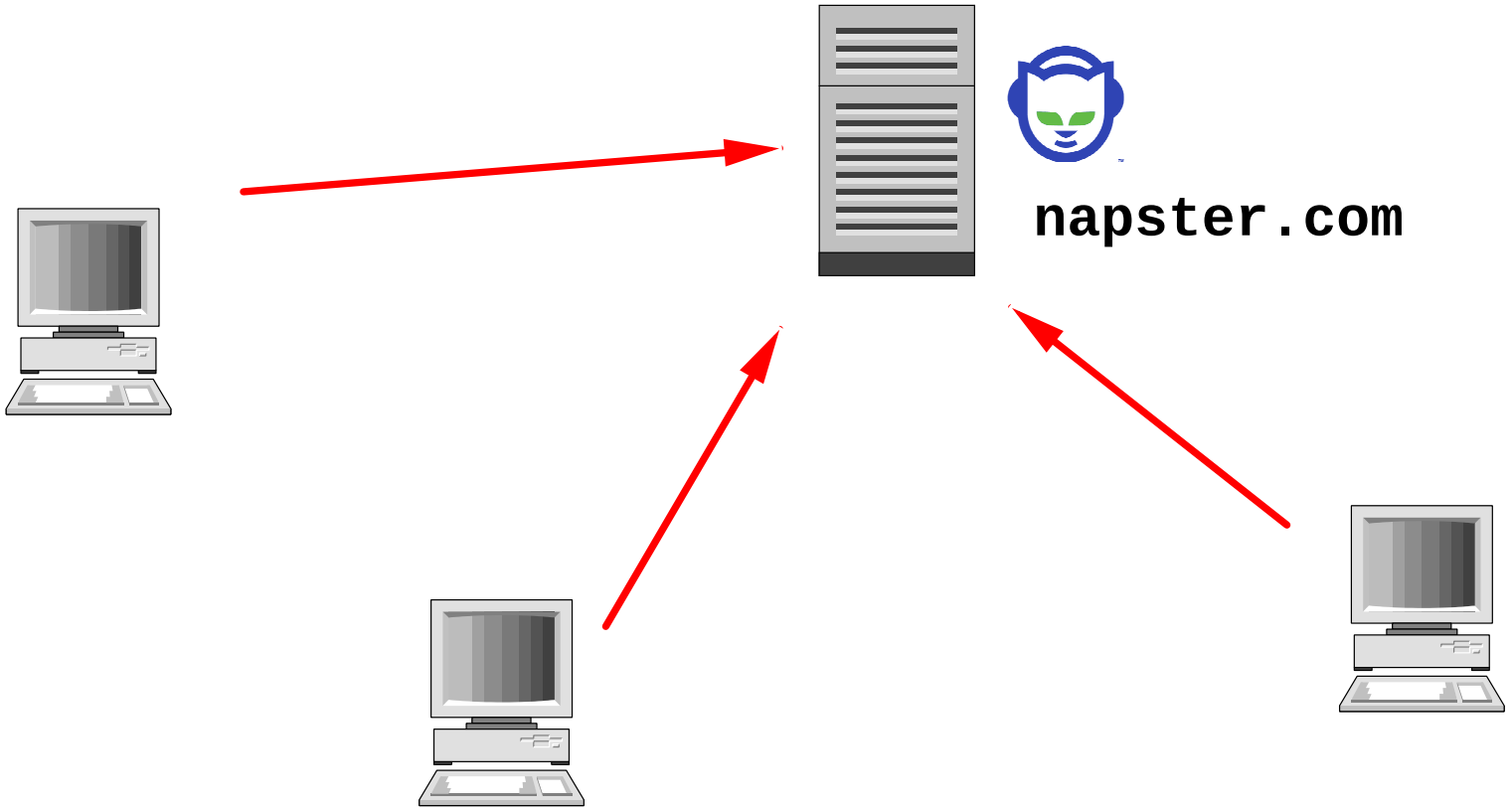
Napster : Principe

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
  - ✓ Selection du meilleur pair (*pings*)

Napster : *upload*

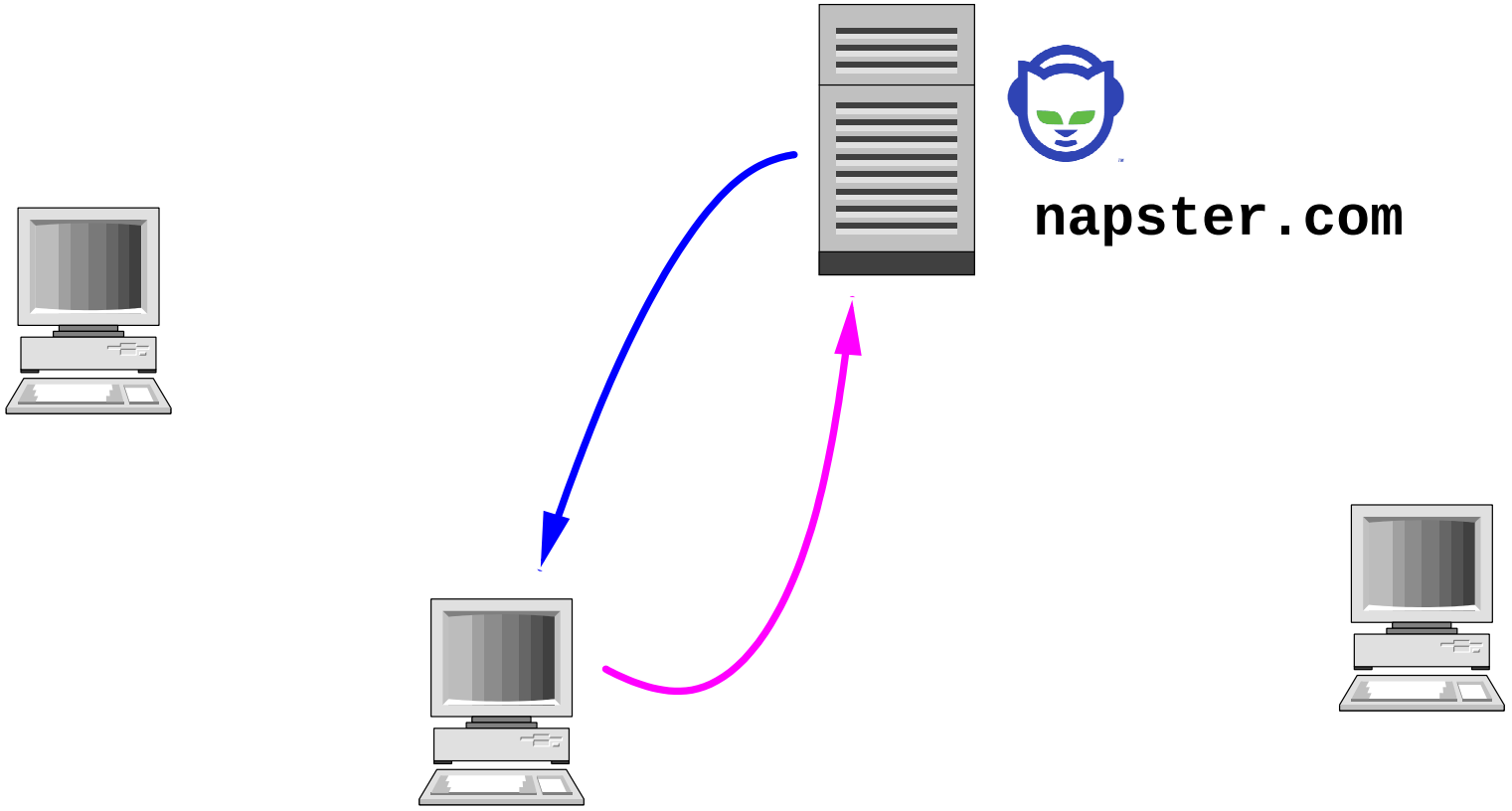
Les utilisateurs chargent la liste des fichiers à partager



Napster : *search*

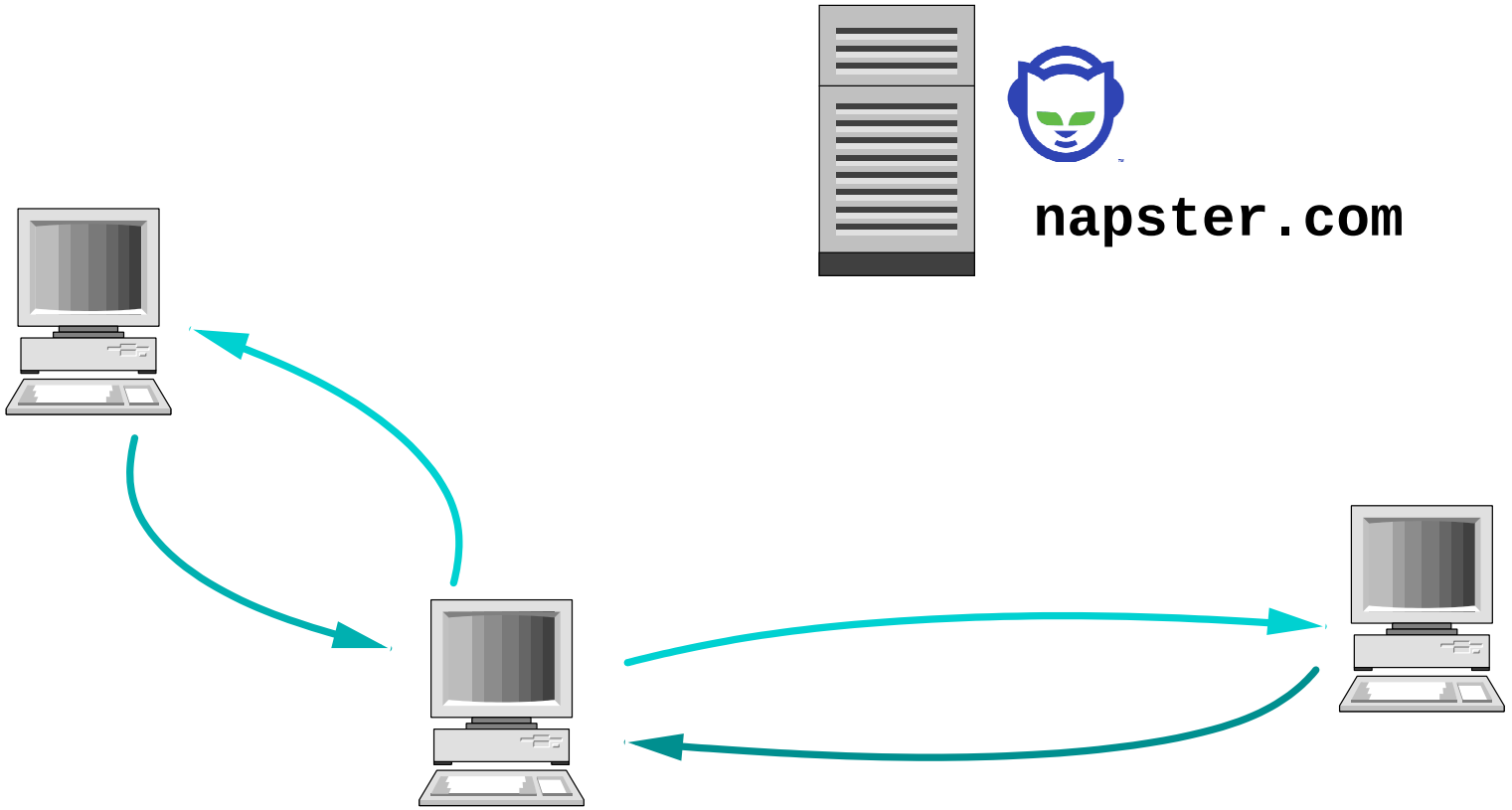
Un utilisateur émet une requête de recherche

Le serveur indique les localisations potentielles



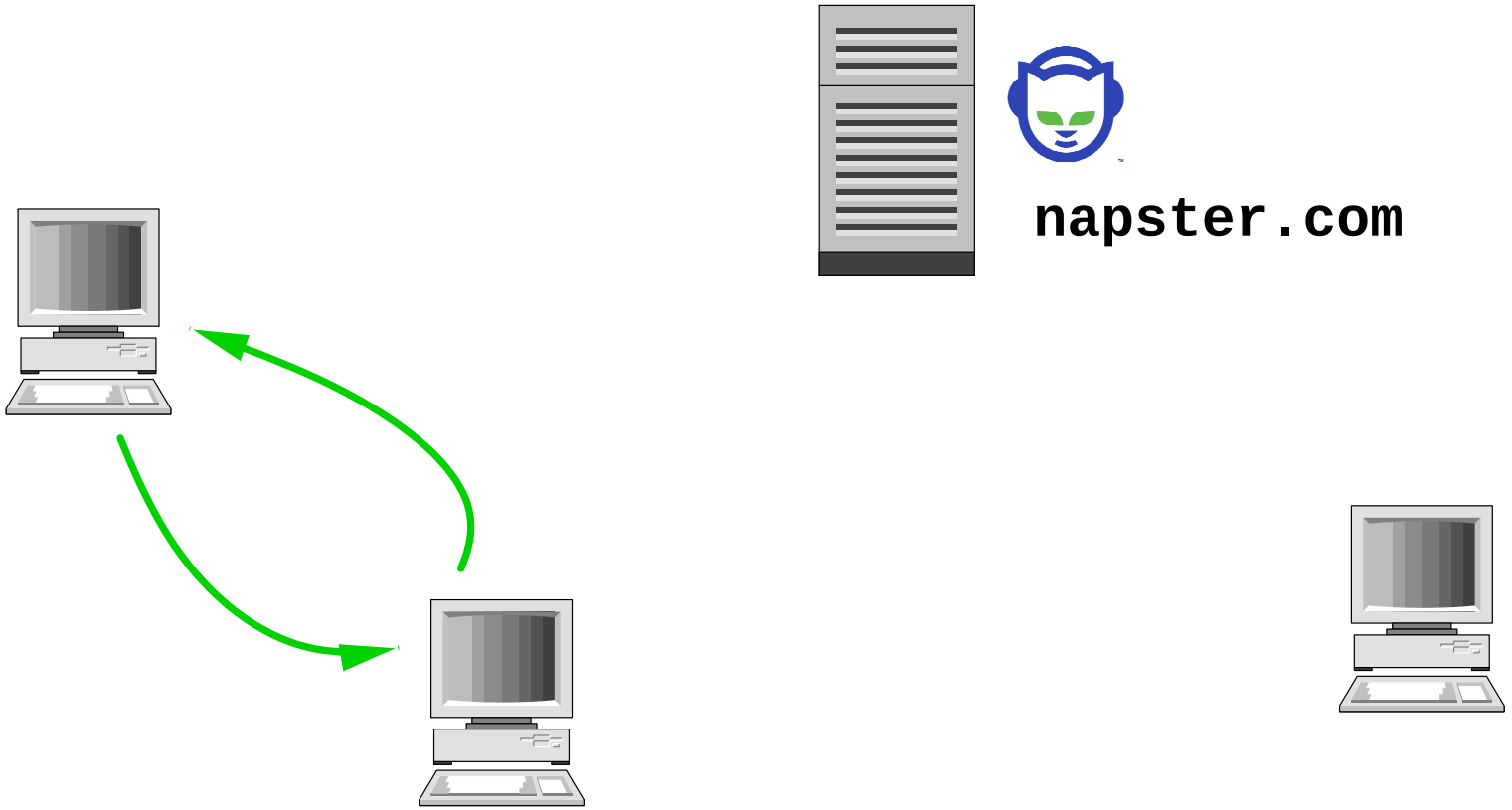
Napster : *pings*

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



Napster : *upload*

L'utilisateur récupère directement le fichier chez le pair choisi



Napster : Messages

Structure du message Napster :

Longueur (16 bits)	**Type** (16 bits)	**Données**...

Longueur : taille du bloc **donnée** en octets

- Attention : encodage en *little-endian*

Type : action à réaliser (voir suite)

- Attention : encodage en *little-endian*

Données Chaine ASCII

- généralement les arguments sont séparés par des espaces (ASCII 32)
- les chaines sont encadrées de guillemets (")
  - ✓ `"Om Lounge 1 - Jazzanova - bohemian sunset.mp3"`
  - ✓ `"nap v0.8"`

Napster : Types

Type	code	description
0	0000	error message [S]
2	0200	login [C]
3	0300	login ack [S]
4	0400	version check [C]
5	0400	auto-upgrade [S]
6	0600	new user login [C]
7	0700	nick check [C]
8	0800	nick not regist. [S]
9	0900	nick already regist. [S]
10	0A00	invalid nick [S]
14	0E00	login options [C]
100	6400	notif. of shared file [C]
102	6600	remove file [C]
110	6E00	unshare all files [C]
200	C800	client search req. [C]
201	C900	search response [S]
202	CA00	end of search resp. [S]
203	CB00	download request [C]
204	CC00	download ack [S]
205	CD00	private message [C, S]
206	CE00	get error [S]
207	CF00	add hotlist entry [C]
208	D000	hotlist [C]
209	D100	user signon [S]
210	D200	user signoff [S]
211	D300	browse a user's files [C]
212	D400	browse response [S]
213	D500	end of browse list [S]
214	D600	server stats [C, S]
215	D700	request resume [C]
216	D800	resume search resp. [S]
217	D900	end of res. search list [S]
218	DA00	downloading file [C]
219	DB00	download complete [C]
220	DC00	uploading file [C]
..	....	...

Napster : Exemple

Demande d'un fichier à `napster.com`

- requête au serveur :
  `26 00 CB 00 monSurnom "C:\MP3\Iggy Pop - Brick By Brick - Candy.mp3"`
- réponse du serveur :
  `51 00 CC 00 monSurnom 2965119704` (adresse IP = A.B.C.D)
  `6699` (port) `"C:\MP3\Iggy Pop - Brick By Brick - Candy.mp3"`
  `1477918779` (checksum sur 32 bits) `10` (vitesse)

Connexion au client A.B.C.D :6699

- reçu du client :
  `31 00 00 00 00 00`
- envoyé au client :
  `GET`
- reçu du client :
  `00 00 00 00 00 00`
- envoyé au clien :t
  `monSurnom "C:\MP3\Iggy Pop - Brick By Brick - Candy.mp3"`
- recu du client :
  `5673288` (taille en octets)
- reçu du client :
  données...

Napster : Remarques

- serveur **centralisé**
  - ✓ un point de défaillance
  - ✓ risque de congestion
  - ✓ partage de charge en utilisant la rotation DNS
  - ✓ controlé par une entreprise
- absence de **sécurité**
  - ✓ mot de passe en clair
  - ✓ pas d'authentification
  - ✓ pas d'anonymat
  - ✓ code propriétaire
  - ✓ téléchargement de mises-à-jour
- évolution :
  - ✓ **OpenNap** :
    - ☞ *open source*
    - ☞ communications entre serveurs
    - ☞ tous types de données

Plan

...

Annuaire

Administration

Peer-to-peer

- Napster
- **Gnutella**
- Freenet
- ...

Gnutella

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

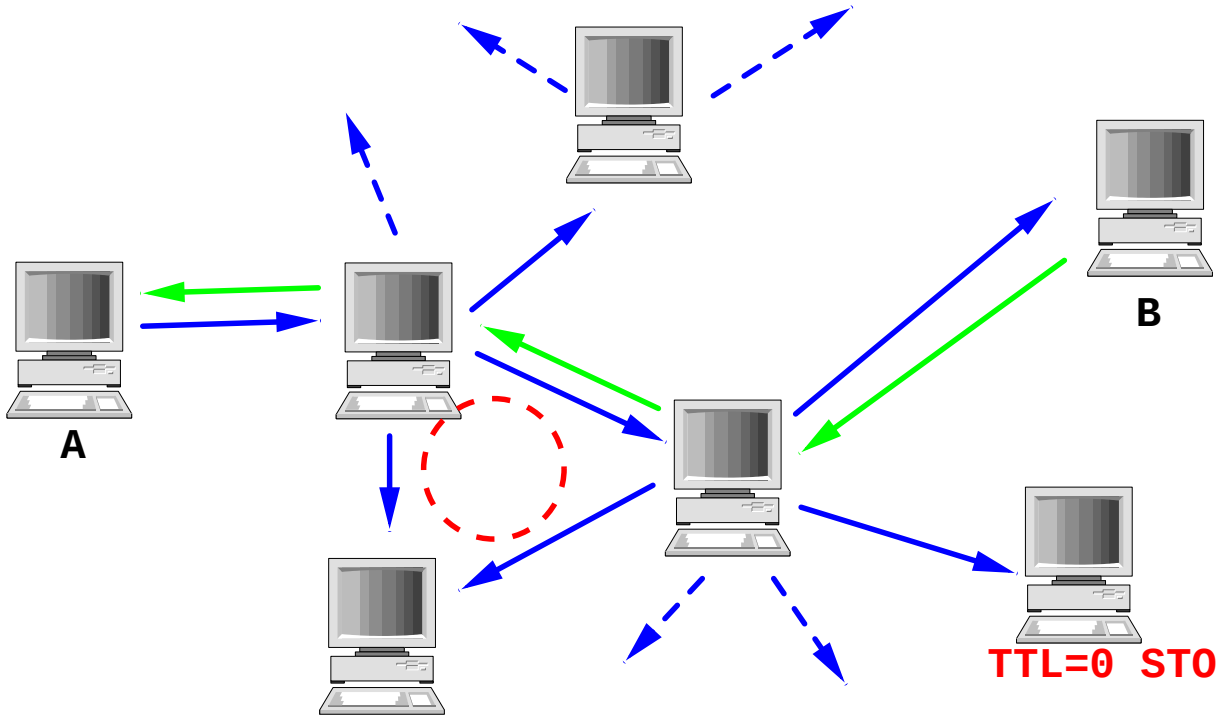
Peer-to-peer Networking

- 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...

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 : Messages

Structure du message de contrôle Gnutella :

Gnode ID (16 octets)	**Type** (1 octet)	**TTL** (1 octet)	**Sauts** (1 octet)	**Longueur** (4 octets)	**Données...**

Gnode ID : identification du nœud dans le réseau gnutella

Type : action à réaliser

- `Ping` (recherche de pair)
- `Pong` (réponse à un `Ping`, adresse IP)
- `Query` (recherche de fichier selon certains critères)
- `Query-Hit` (réponse à un `Query`, avec liste des fichiers et adresse IP)
- `Push` (mécanisme de passage de *firewall*)

TTL : nombre de sauts encore réalisables

Sauts : nombre de sauts réalisés

- $TTL_n + Sauts_n = TTL_{initial}$

Longueur : taille du bloc **données** en octets

Données : peuvent être vides

Gnutella : Identification des pairs (1)

Détection active des pairs

- requête `Ping`
  - ✓ pas de données
  - ✓ limitations des envois pour ne pas saturer le réseau
  - ✓ crée un état dans la table de routage (retour des `Pong`)
  - ✓ répondre et relayer aux pairs connectés (limite du TTL)
- réponse `Pong`
  - ✓ données :

Port (2 octets)	Adresse IP (4 octets)	Nb de fichiers (4 octets)	Nb de Ko partagés (4 octets)

Port : port sur lequel le pair est en attente

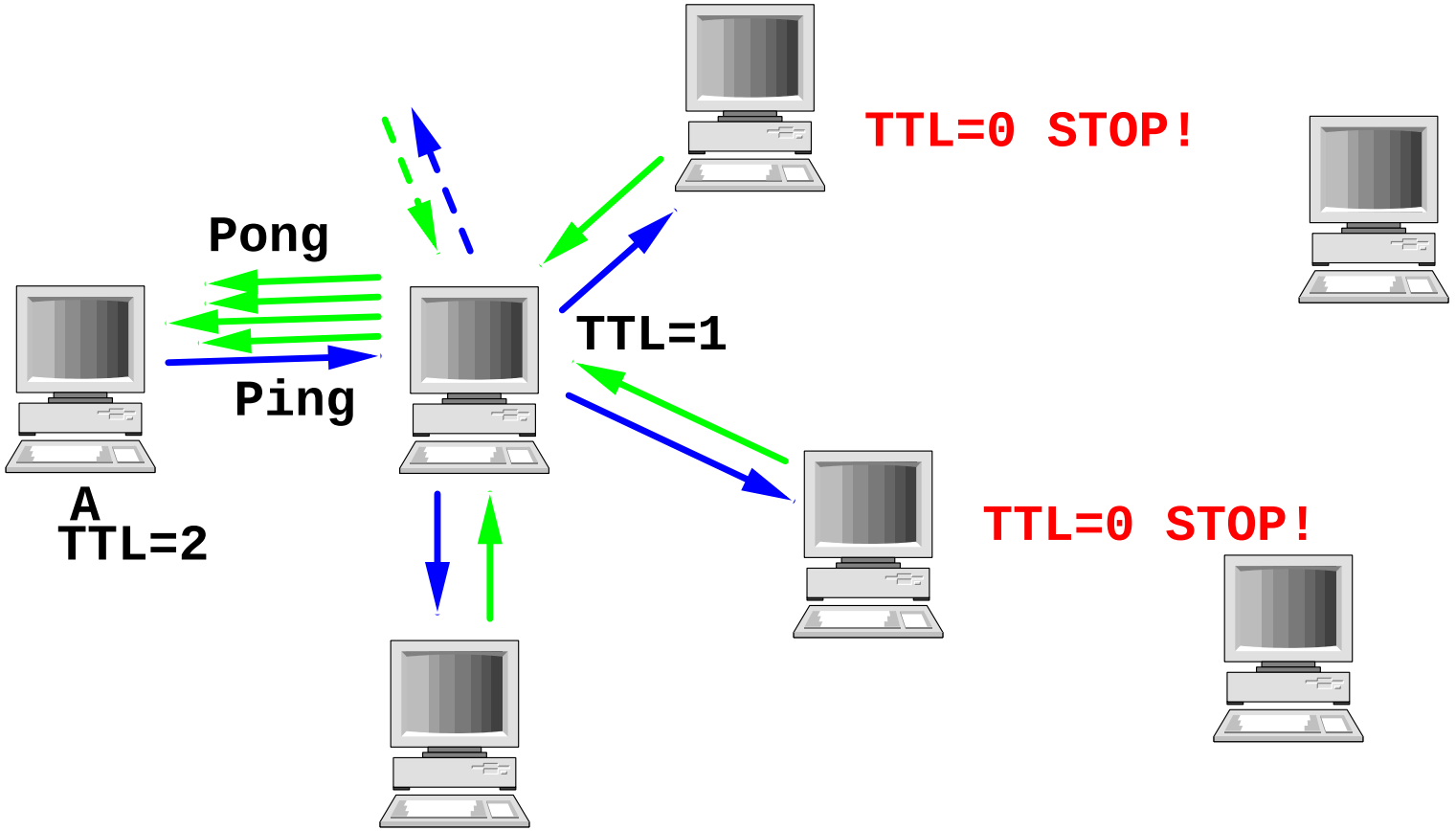
Adresse IP : adresse à laquelle le pair est atteignable

Nb de fichiers : nombre de fichiers partagés par le pair

Nb de Ko partagés : quantité de données partagées par le pair

Gnutella : Identification des pairs (2)

Détection active des pairs



Gnutella : Recherche de fichier (1)

Requête pour trouver une information

- requête `Query`
  - ✓ données :

Vitesse minimum (2 octets)	Critères de recherche (N octets)

Vitesse : débit minimum pour qu'un pair réponde (Ko/s)

Critères : chaine de caractères terminée par `0x00`

  - ✓ crée un état dans la table de routage (retour des `Query-Hit`)
  - ✓ relayer aux pairs connectés (limite du TTL)
- réponse `Query-Hit`...

Gnutella : Recherche de fichier (2)

- requête `Query`
- réponse `Query-Hit`
  - ✓ données :

Nb Hit (1 octet)	Port (2 octets)	Adresse IP (4 octets)	Vitesse (4 octets)	Résultats (N octets)	GID Pair (16 octets)

Nb Hit : Indique le nombre de champs des **Résultats**

Port : Port sur lequel le pair est en attente

Adresse IP : adresse à laquelle le pair est atteignable

Vitesse : débit minimum pour qu'un pair réponde (Ko/s)

Résultats : ensemble de **Nb Hit** champs :

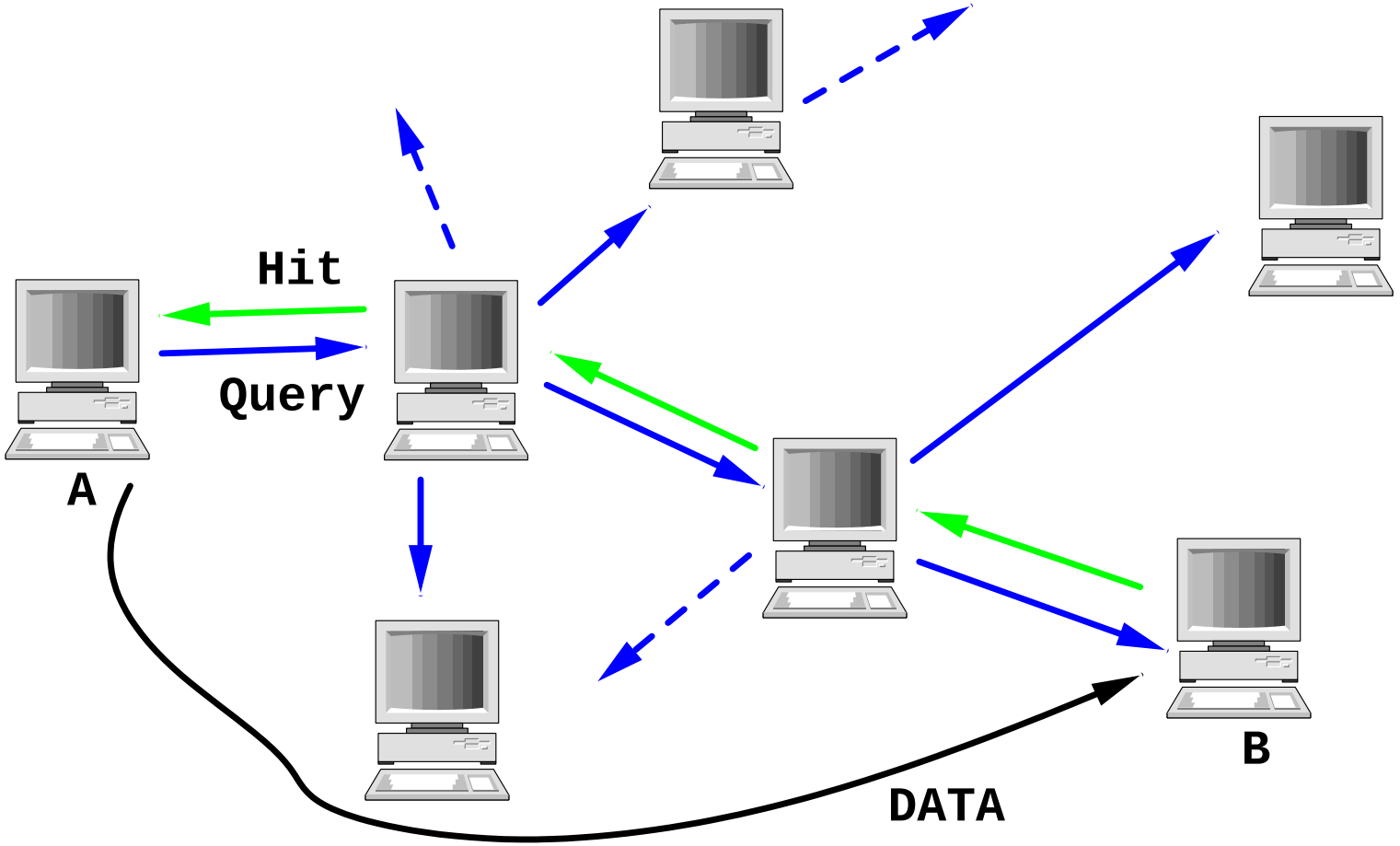
Index du fichier (4 octets)	Taille du fichier (4 octets)	Nom du fichier chaine terminant par 0x0000

GID Pair : identification pour un `Push`

  - ✓ sont routées selon le chemin inverse suivi par reqêtes `Query`

Gnutella : Recherche de fichiers (3)

Requête pour trouver une information



Gnutella : *Firewall* (1)

Retourner la connexion des données

- requête `Push`
  - ✓ données :

GID Pair (16 octets)	Index du fichier (4 octets)	Adresse IP (4 octets)	Port (4 octets)

GID Pair : identification du pair

Index : Identifiant unique du fichier sur le pair

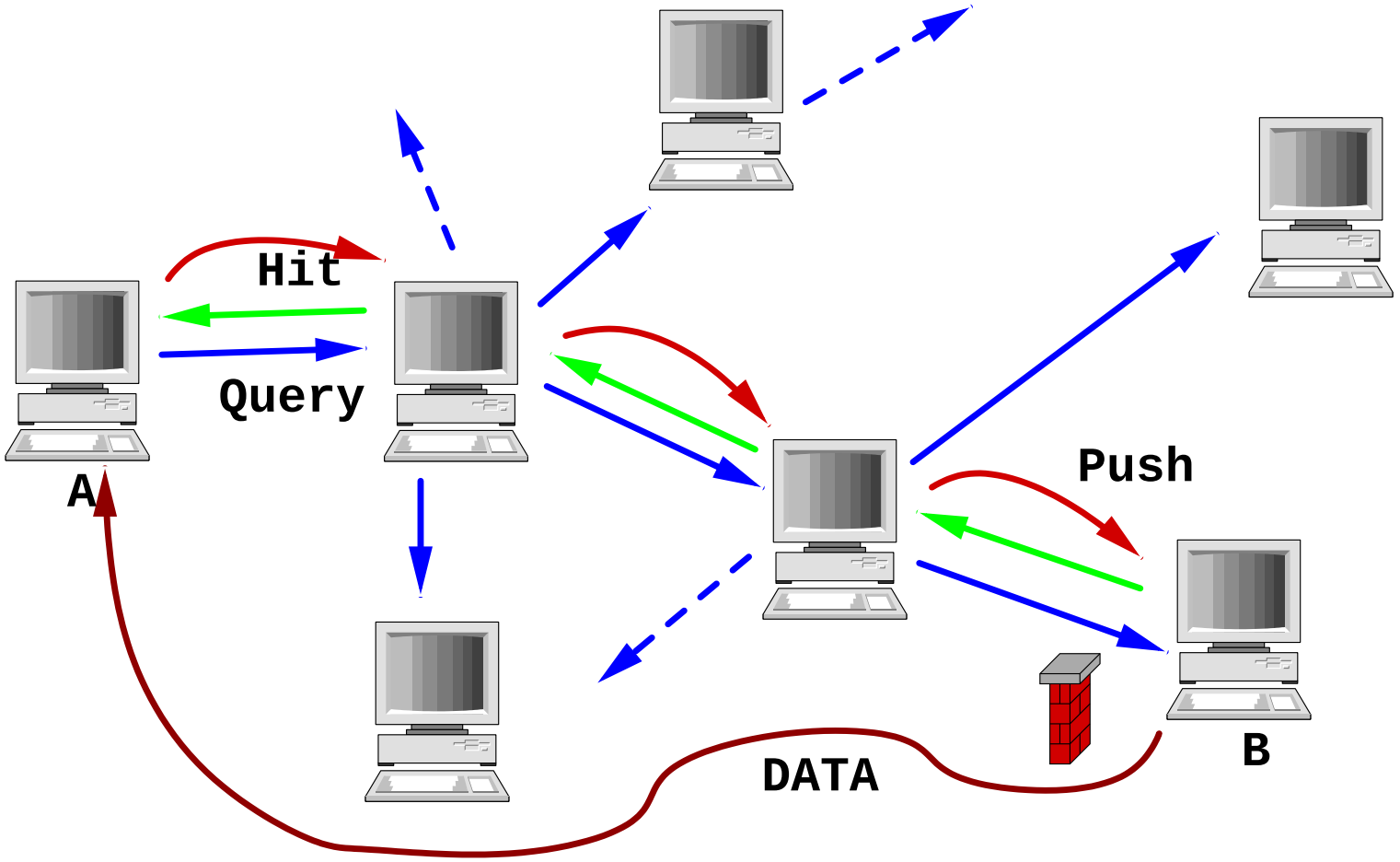
Adresse IP : adresse à laquelle le fichier doit être envoyé

Port : Port sur lequel le destinataire est en attente

  - ✓ sont routées selon le chemin inverse suivi par réponses `Query-Hit`
  - ✓ permet la création de la connexion donnée à partir du pair

Gnutella : *Firewall* (2)

Retourner la connexion des données



Gnutella : Gestion des connexions

Connexion de contrôle sur TCP

- demande de connexion :
  ```
  GNUTELLA CONNECT/0.6
  Node: 201.33.182.178:6346
  User-Agent: gtk-gnutella/0.80 beta2 - 22/01/2002
  ```
- réponse du pair :
  ```
  GNUTELLA/0.6 200 OK
  User-Agent: Morpheus 2.0.1.7
  Remote-IP: 181.185.36.178
  ```
- confirmation :
  ```
  GNUTELLA/0.6 200 OK
  ```

Récupération des données par **HTTP**

- séparée du réseau Gnutella :
  - ✓ connexion directe entre pairs et envoi d'un `GET`

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 hiérachie de pairs
- **anonymat ?**
  - ✓ le pair où l'on récupère le fichier nous connait
    - ☞ $\exists$ protocoles permettant de ne pas connaitre le destinataire

Plan

...

Annuaire

Administration

Peer-to-peer

- Napster
- Gnutella
- **Freenet**
- ...

Freenet

Partage de fichier **complètement anonyme**

- *peer-to-peer* décentralisé (comme Gnutella)
  - ✓ connaissance locale seulement
  - ✓ accès aux ressources de proche en proche (routage)
  - ✓ producteur anonyme
  - ✓ consommateur anonyme
  - ✓ résistance aux tentatives de limitation d'accès
- historique
  - ✓ 1999 projet de fin d'étude de Ian Clarke (Université d'Edimbourg)
  - ✓ 2000 projet *SourceForge* très actif :
  - ✓ 03/00 v 0.1
  - ✓ 07/03 v 0.5.2.1
    - ☞ à voir sur `http://sourceforge.net/projects/freenet/`

Plan

...

Annuaire

Administration

Peer-to-peer

- Napster
- Gnutella
- Freenet
- ...

P2P : Autre

Autre systèmes *peer-to-peer* de recherche par le contenu :

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

Fin

Document réalisé avec LaTeX.
Généré le 2 novembre 2004.
Classe de document `foils`.
Dessins réalisés avec `xfig`.

Olivier Fourmaux, `olivier.fourmaux@lip6.fr`
`http://www-rp.lip6.fr/~fourmaux`

Ce document est disponible en postscript compressé avec gzip à
`http://www-rp.lip6.fr/~fourmaux/res/res4c3c-.pdf`