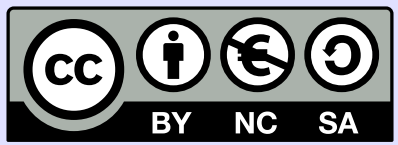
M1101
Initiation aux Réseaux

Sami Evangelista
IUT de Villetaneuse
Département Réseaux et Télécommunications
2017–2018

http://www.lipn.univ-paris13.fr/~evangelista/cours/M1101

Sources des images utilisées dans ce document:

http://www-lipn.univ-paris13.fr/~evangelista/cours/credits.html



Contenu du module

M1101 — Initiation aux réseaux d'entreprise

- Objectif : donner un aperçu des techniques réseau et systèmes d'exploitation étudiées en DUT.
- Module de culture générale : beaucoup de notions abordées mais pas forcément en détails.
- Notions approfondies dans d'autres modules.
- 4 parties :
  1. Codage de l'information
  2. Introduction au réseau IP
  3. Les services réseau
  4. Administration des systèmes

Organisation du module

Trois enseignants :

- Hervé Costantini (TD + TP)
- Sami Evangelista (Cours + TP)
- Giulio Manzonetto (TD + TP)

Répartition des heures :

- 4 cours de 1h–1h30.
- 4 séances de TD de 3h.
- 7 séances de TP de 3h.
- 1 examen court de 1h30–2h
- 1 examen long de 3h

Notation :

- 3 séances de TP notées (avec compte-rendu à rendre en fin de séance)
- 2 examens

Autres informations

- Mon adresse électronique : sami.evangelista@iutv.univ-paris13.fr
- La page du cours :

  `http://www.lipn.univ-paris13.fr/~evangelista/cours/M1101`

- Vous y trouverez :
  - transparents des cours
  - sujets des TD et TP
  - correction des TP

Plan

- L'informatique est le traitement automa**tique** de l'**informa**tion.
  - automatique = réalisable par une machine
  - information = données de l'utilisateur (des nombres, du son, des images, du texte, ...)
- Pour travailler sur des données l'ordinateur a besoin de les représenter dans un langage qu'il comprend : le langage machine basé sur le système binaire.
- On appelle **codage** le procédé qui consiste à traduire les données en langage machine et **décodage** le procédé inverse.

Le bit

- Le bit est la plus petite unité de représentation de l'information.
- bit = contraction de **bi**nary digi**t** (chiffre binaire)
- Un bit a pour valeur 0 ou 1.
- Dans certains cas (voir les opérateurs logiques) on associe aussi :
  - 0 à faux
  - 1 à vrai
- Une suite de 8 bits est appelée un **octet**.
  Une suite de 4 bits est appelée un **quartet**.
- En informatique, on code les données avec des séquences de bits (ou nombres binaires).

Notation

- Pour préciser le système (binaire ou décimal) dans lequel est écrit un nombre on utilise un indice (2 ou 10).
- Exemples :
  - 1001_2 est un nombre binaire
  - 1001_{10} est un nombre décimal
- Si l'indice n'est pas précisé, le nombre est écrit dans le système décimal.

Bits de poids fort/faible

- Dans une séquence de bits $b_{n-1}b_{n-2}\ldots b_0$, on appelle :
  - bit de **poids faible**, le bit le plus à droite (b_0)
  - bit de **poids fort**, le bit le plus à gauche (b_{n-1})
  - bit de **poids** p, le bit b_p
- Par exemple, dans la séquence $\underline{1}0111\underline{0}01\underline{0}_2$:
  - Le bit de poids faible est 0.
  - Le bit de poids fort est 1.
  - Le bit de poids 3 est 0.

Unités de représentation de l'information

Basées sur les bits		
kilo-bit	Kb	10^3 bits
mega-bit	Mb	10^6 bits
giga-bit	Gb	10^9 bits
tera-bit	Tb	10^{12} bits
peta-bit	Pb	10^{15} bits
exa-bit	Eb	10^{18} bits

Basées sur les octets		
kilo-octet	Ko	10^3 octets
mega-octet	Mo	10^6 octets
giga-octet	Go	10^9 octets
tera-octet	To	10^{12} octets
peta-octet	Po	10^{15} octets
exa-octet	Eo	10^{18} octets

Quelques ordres de grandeurs :

- un document office = 10–100 Ko
- un fichier de musique = 1–10 Mo
- un DVD = 4–8 Go
- un disque dur standard = 1–10 To = 1 millier de DVD
- quantité de données que peut stocker la NSA au data center de Bluffdale (Utah, USA) = 3–12 Eo = 3 millions de disques durs de 2 To

Capacité de codage

- Combien de séquences binaires différentes peut-on écrire
  - avec 1 bit ? 2 séquences : 0 et 1.
  - avec 2 bits ? 4 séquences : $00, 01, 10$ et 11.
  - avec 3 bits ? 8 séquences : $000, 001, 010, 011, 100, 101, 110, 111$.
  - ...
  - avec n bits ? 2^n séquences.
- Autrement dit, l'utilisation d'un bit supplémentaire permet de doubler la capacité de codage.
- Chaque séquence binaire permet ensuite de coder une valeur parmi celles que l'on souhaite représenter (caractères, nombres, couleurs, ...).
- Inversement, pour trouver le nombre de bits nécessaires au codage de v valeurs, il faut trouver la plus petite puissance de 2 supérieure à v.
  - (On obtient cette puissance, avec l'opération $\log_2(v)$.)

Exemple : Codage de l'alphabet latin

- Pour représenter les 26 caractères de l'alphabet latin, il nous faut 5 bits.
  - (Car $2^4 = 16 < 26$ et $2^5 = 32 \geq 26$.)
- On peut ensuite utiliser la représentation suivante :
  - $00000 \rightarrow \mathrm{a}, 00001 \rightarrow \mathrm{b}, 00010 \rightarrow \mathrm{c}, \ldots, 11001 \rightarrow \mathrm{z}$.

Représentation des entiers naturels

- Pour compter en binaire c'est le même principe qu'en décimal :

  $0, 1, \underbrace{10}_{2}, \underbrace{11}_{3}, \underbrace{100}_{4}, \underbrace{101}_{5}, \underbrace{110}_{6}, \underbrace{111}_{7}, \underbrace{1000}_{8}, \underbrace{1001}_{9}, \ldots$

- On a donc :
  - $1_2 = 1 = 2^0$
  - $10_2 = 2 = 2^1$
  - $100_2 = 4 = 2^2$
  - $1000_2 = 8 = 2^3$

  ...
- Plus généralement, la valeur décimale associée au bit de poids n est 2^n.
- On en déduit donc la valeur décimale d'une suite binaire $b_{n-1}b_{n-2}\ldots b_0$:

$$b_{n-1} \times 2^{n-1} + b_{n-2} \times 2^{n-2} + \ldots + b_0 \times 2^0 = \sum_{i=0}^{n-1} b_i \times 2^i$$

- Exemple :

  $1001101_2 = 1 \times 2^6 + 1 \times 2^3 + 1 \times 2^2 + 1 \times 2^0 = 64 + 8 + 4 + 1 = 77$

Généralisation à une base B quelconque

- On connaît maintenant le système binaire basé sur les bits et le système décimal basé sur les chiffres arabes.
- On appelle base B un système dans lequel on dispose d'un alphabet de B chiffres (ou symboles) pour écrire des nombres.
  - système binaire = base 2
  - système décimal = base 10
- Ce que l'on a vu pour le système binaire peut être généralisé :
  - $x_{n-1}x_{n-2}\ldots x_0$ dans une base B vaut $x_{n-1} \times B^{n-1} + x_{n-2} \times B^{n-2} + \ldots + x_0$ en décimal.
  - Dans une base B, n chiffres permettent de coder B^n valeurs différentes.
- Exemples :
  - $5672_8 = 5 \times 8^3 + 6 \times 8^2 + 7 \times 8^1 + 2 \times 8^0 = 3002$
  - En base 7 on peut coder, avec 3 chiffres, $7^3 = 343$ valeurs différentes.

B	Nom	Chiffres	Exemple : écriture de 93
2	binaire	$0, 1$	1011101_2
8	octale	$0, 1, 2, 3, 4, 5, 6, 7$	135_8
10	décimale	$0, 1, 2, 3, 4, 5, 6, 7, 8, 9$	93_{10}
16	hexadécimale	$0, 1, 2, 3, 4, 5, 6, 7,$ $8, 9, A, B, C, D, E, F$	$5D_{16}$

Exemples d'utilisations :

- binaire : par l'ordinateur pour faire des calculs
- octale : pour représenter les droits sur un fichier (lecture, écriture, exécution)
- hexadécimale : pour représenter des séquences de bits de manière compacte

Problème : convertir un nombre décimal N en un nombre d'une base B

- On cherche à décomposer N en puissances de B :

$$N = r_{n-1} \times B^{n-1} + \ldots + r_0 \times B^0$$

- Méthode générale :
  - On divise N par B.
  - Le reste forme le chiffre de poids faible.
    (Le reste est forcément dans l'intervalle $[0..B-1]$.)
  - On recommence l'opération avec le quotient.
  - On s'arrête lorsque le quotient vaut 0.

conversion de 157 en binaire

$$
\begin{array}{lrlrcrcrcrl}
\text{division de} & 157 & \text{par 2 :} & 157 & = & 2 & \times & 78 & + & \boxed{1} & r_0 \\
\text{division de} & 78 & \text{par 2 :} & 78 & = & 2 & \times & 39 & + & \boxed{0} & r_1 \\
\text{division de} & 39 & \text{par 2 :} & 39 & = & 2 & \times & 19 & + & \boxed{1} & r_2 \\
\text{division de} & 19 & \text{par 2 :} & 19 & = & 2 & \times & 9 & + & \boxed{1} & r_3 \\
\text{division de} & 9 & \text{par 2 :} & 9 & = & 2 & \times & 4 & + & \boxed{1} & r_4 \\
\text{division de} & 4 & \text{par 2 :} & 4 & = & 2 & \times & 2 & + & \boxed{0} & r_5 \\
\text{division de} & 2 & \text{par 2 :} & 2 & = & 2 & \times & 1 & + & \boxed{0} & r_6 \\
\text{division de} & 1 & \text{par 2 :} & 1 & = & 2 & \times & 0 & + & \boxed{1} & r_7
\end{array}
$$

Le quotient vaut 0 $\Rightarrow$ on s'arrête

Conclusion : $157_{10} =$

1	0	0	1	1	1	0	1
r_7	r_6	r_5	r_4	r_3	r_2	r_1	r_0

conversion de 157 en hexadécimal

division de 157 par 16 : $157 = 16 \times 9 + \boxed{13}$ r_0

division de 9 par 16 : $9 = 16 \times 0 + \boxed{9}$ r_1

Le quotient vaut 0 $\Rightarrow$ on s'arrête

Conclusion : $157_{10} =$ 9 D

r_1 r_0

(13 est représenté par le caractère D en hexadécimal.)

Problème : convertir un nombre d'une base B en un nombre décimal

- Il suffit de décomposer le nombre en fonction des puissances de la base et d'additionner.
- Exemples :
  - $1011101_2 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 93_{10}$
  - $E07_{16} = 14 \times 16^2 + 0 \times 16^1 + 7 \times 16^0 = 3\ 591_{10}$

Problème : convertir un nombre binaire en un nombre hexadécimal

- Avec 4 bits on peut coder $2^4 = 16$ valeurs = 1 chiffre hexadécimal.
- Méthode générale :
  - On groupe les bits par 4 en commençant par les bits de poids faible.
  - On rajoute si nécessaire des bits de poids fort à 0 pour obtenir un nombre de bits multiple de 4.
  - On convertit chaque suite de 4 bits en hexadécimal comme suit :

0000	⇒	0	0100	⇒	4	1000	⇒	8	1100	⇒	C
0001	⇒	1	0101	⇒	5	1001	⇒	9	1101	⇒	D
0010	⇒	2	0110	⇒	6	1010	⇒	A	1110	⇒	E
0011	⇒	3	0111	⇒	7	1011	⇒	B	1111	⇒	F

- Exemples :
  - $10110011_2 = 1011\ 0011_2 = B3_{16}$
  - $11100_2 = 0001\ 1100_2 = 1C_{16}$

Problème : convertir un nombre hexadécimal en un nombre binaire

- Il suffit de coder chaque chiffre hexadécimal en une suite de 4 bits.
- Exemples :
  - $A7_{16} = 1010\ 0111_2$
  - $7EF_2 = 0111\ 1110\ 1111_{16}$

L'addition

- L'addition se fait (quelle que soit la base) comme en base 10.
- Exemple : addition de $149 = 10010101_2$ et $305 = 100110001_2$ en binaire

			1	1				1		retenues
		0	1	0	0	1	0	1	0	1
+		1	0	0	1	1	0	0	0	1
	1	1	1	0	0	0	1	1	0	

On trouve bien $111000110_2 = 454 = 149 + 305$

Le dépassement de capacité

- Avant de réaliser une opération, l'ordinateur stocke les opérandes dans des **registres** (zones mémoire très rapides d'accès).
- Ces registres ont des tailles limitées (8, 32 ou 64 bits par exemple).

$\Rightarrow$ L'intervalle des valeurs que l'on peut y stocker est limité.

- Si le résultat d'une opération ne tient pas dans le registre, certains bits seront « oubliés » et le résultat sera incorrect.
- On appelle ce phénomène **dépassement de capacité** ou **overflow**.
- Exemple : addition sur 8 bits de $77 = 01001101_2$ et $201 = 11001001_2$.

	1	1			1			1	retenues
		0	1	0	0	1	1	0	1
+		1	1	0	0	1	0	0	1
	~~1~~	0	0	0	1	0	1	1	0

$\Rightarrow$ Lorsque l'ordinateur effectue $77_{10} + 201_{10}$ avec des registres de 8 bits, il trouve $00010110_2 = 22_{10}$.

- Le dépassement de capacité est la source de nombreux bugs.
  Exemple : crash d'ariane 5 en 1996 dû à l'utilisation de registres de 16 bits pour des opérations nécessitant 64 bits.

Opérateurs logiques

- En plus des opérateurs arithmétiques $(+, -, \times, /)$ les programmes ont parfois besoin d'effectuer des opérations logiques.
- On ne voit plus alors la séquence binaire comme un nombre mais comme une suite de valeurs **vrai** (1) ou **faux** (0).
- Les opérateurs logiques sont des opérateurs bit à bit : dans le résultat, le bit de poids p dépend uniquement du (ou des) bit(s) de poids p dans le(s) opérandes.
- On utilise pour cela une **table de vérité** qui donne, en fonction des bits des opérandes, le bit résultant de l'opération.
- Nous allons voir 4 opérateurs logiques : ET, OU, OU exclusif et NON.
- Exemples d'utilisation de ces opérateurs :
  - ET, OU et NON utilisés pour certains calculs sur les adresses IP (voir cours et TD 2)
  - OU exclusif utilisé par des algorithmes de cryptage (voir TD 1)

Le ET logique

- Les deux bits doivent être à vrai.
- Table de vérité du ET :

a	b	a **ET** b
0	0	0
0	1	0
1	0	0
1	1	1

- Exemple :

$$\begin{array}{rcccccccc} & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ \textbf{ET} & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ \hline & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{array}$$

Le OU logique

- Au moins une des opérandes doit être à vrai.
- Table de vérité du OU :

a	b	a **OU** b
0	0	0
0	1	1
1	0	1
1	1	1

- Exemple :

	1	1	0	1	1	0	0	1
OU	0	1	1	1	0	0	1	1
	1	1	1	1	1	0	1	1

Le OU excusif (XOR)

- Une des deux opérandes doit être à vrai mais pas les deux.
- Table de vérité du XOR :

a	b	a **XOR** b
0	0	0
0	1	1
1	0	1
1	1	0

- Exemple :

	1	1	0	1	1	0	0	1
XOR	0	1	1	1	0	0	1	1
	1	0	1	0	1	0	1	0

Le NON logique

- Inversion du bit de l'opérande.
- Table de vérité du NON :

a	**NON** a
0	1
1	0

- Exemple :

NON	0	1	1	1	0	0	1	1
	1	0	0	0	1	1	0	0

Comment représenter du texte ?

- Une **table de codage** associe une séquence binaire à chaque caractère de l'alphabet que l'on veut coder.
- L'ordinateur traduit ensuite chaque caractère du texte en la séquence de bits correspondante en utilisant cette table.
- Il y a différentes tables de codage.
- Elles se différencient par :
  - l'ensemble des caractères qu'elles permettent de coder
  - et le nombre de bits qu'elles utilisent pour coder les caractères.
- Nous allons en voir 2 : l'ASCII et l'UTF-8.
- Pour voir le codage utilisé pour un fichier : `file <nom-du-fichier>`.
- Exemple :

```
$ file message.txt
message.txt: UTF-8 Unicode text
```

Le codage ASCII

- ASCII = American Standard Code for Information Interchange
- caractères ASCII codés sur 7 bits
- Permet de coder : *les lettres de l'alphabet latin* (minuscules et majuscules), *les chiffres arabes*, *les caractères de ponctuation* et d'autres *caractères spéciaux* (ex : +, -, ESC (la touche Echap du clavier)).
- La table des caractères ASCII :

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	¨	#	$	%	&	`	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	´	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- En ligne : les 3 bits de poids fort. En colonne : les 4 bits de poids faible.
- Exemple : le caractère E est codé $\underbrace{100}_{4}\ \underbrace{0101}_{5}$

Le codage UTF-8

- UTF-8 = Universal character set Transformation Format - 8 bits
- codage le plus utilisé dans les systèmes Linux
- caractères UTF-8 codés sur 1, 2, 3 ou 4 octets
- permet de coder des textes d'à peu près n'importe quel alphabet (arabe, chinois, grec, indien, latin, ...)
- compatible avec l'ASCII : Un texte codé en ASCII est codé de la même manière en UTF-8.
  - (Les caractères ASCII étant codés sur 7 bits on rajoute un bit de poids fort à 0 pour obtenir un caractère UTF-8 sur 1 octet.)
- Exemples de codage :

E	$\rightarrow$	01000101
é	$\rightarrow$	11000011 10101001
€	$\rightarrow$	11100010 10000010 10101100
α	$\rightarrow$	11001110 10110001

Plan

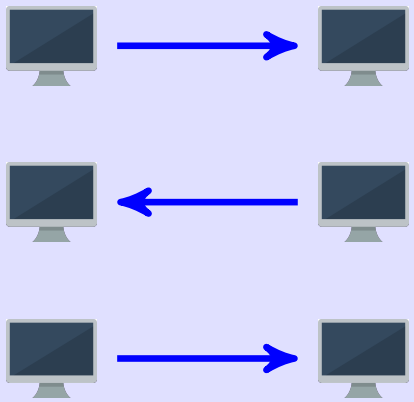
La notion de protocole

- un protocole = ensemble de règles régissant les échanges entre 2–N parties (ou **entités protocolaires**)
  - entité protocolaire = carte Wi-Fi, système d'exploitation, logiciel de messagerie, ...
- Les protocoles réseau sont basés sur l'**échange de messages**.
- Pour que l'échange se déroule correctement, il faut que toutes les entités suivent le même protocole.
- Un protocole indique les actions à réaliser dans certains contextes.
- Exemple de règle appliquée par un logiciel de messagerie :
  - Contexte : je reçois un mail.
  - Actions à réaliser :
    1. Si le mail m'est bien destiné, j'envoie un *accusé de réception*.
    2. Sinon j'envoie un message *erreur, destinataire incorrect*.

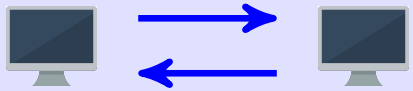
Liaisons half duplex et full duplex

2 modes de transmission pour les liaisons point-à-point (entre 2 machines) :

Half duplex
l'un après l'autre



Full duplex
les deux en même temps



- En half-duplex, une machine ne sait pas forcément si l'autre est en train de transmettre des données.

$\Rightarrow$ Elle peut commencer sa transmission alors que des données lui arrivent.

- Les deux messages vont se rencontrer sur le câble : on parle de **collision**.
- Les deux machines devront alors retenter leurs transmissions plus tard.

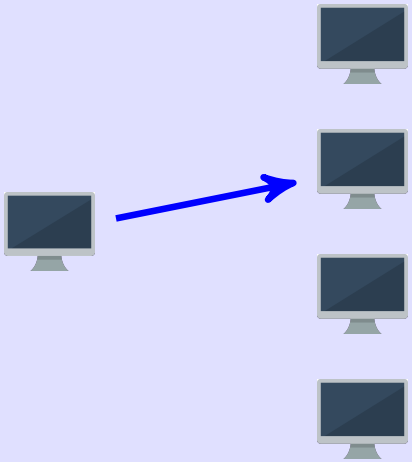
La notion d'adresse

- Pour les besoins de la communication, il est nécessaire de pouvoir désigner les machines sur le réseau.
- On utilise pour cela les **adresses**.
- L'adresse identifie une machine sur le réseau.
- Pour que l'échange se fasse correctement il faut que ces adresses soient uniques : deux machines ne peuvent pas partager la même adresse.
- Nous allons voir l'adressage IP utilisé sur Internet mais il y en a d'autres.

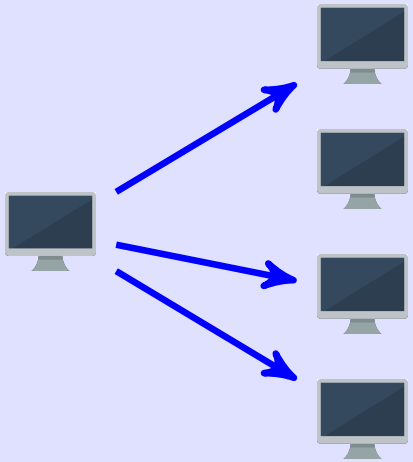
Destinataires d'un message

Unicast, multicast, broadcast (ou diffusion) permettent de caractériser l'envoi d'un message par rapport au nombre de destinataires sur le réseau.

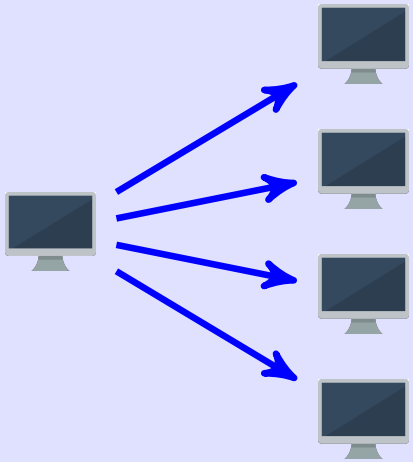
unicast
1 vers 1



multicast
1 vers plusieurs



broadcast
1 vers tous

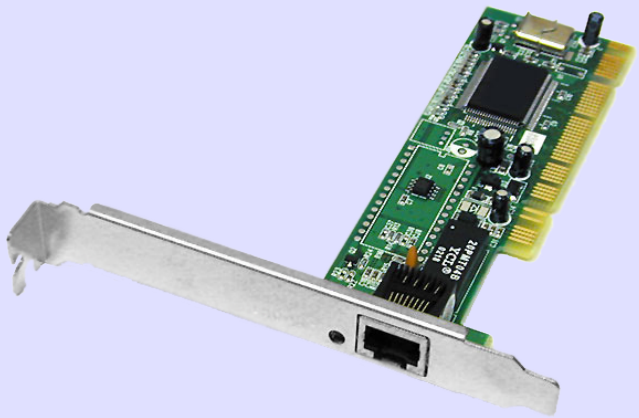


Dans le cas du multicast et du broadcast, une même adresse désigne plusieurs machines.

Interfaces réseau

- interface réseau = moyen utilisé par l'ordinateur pour envoyer/recevoir des données sur le réseau
- interface réseau = carte réseau (Ethernet, Wi-FI, ...)

  Ethernet = réseaux filaires les plus utilisés
- Photo d'une carte Ethernet :



- Sous Linux les interfaces réseau sont désignées par un nom de la forme `typeN` avec :
  - `type` = type de réseau (eth = Ethernet, wlan = Wi-Fi, ...)
  - `N` = numéro de la carte de ce type (la numérotation démarre à 0)

  Exemple : `eth1` = $2^{\text{ème}}$ carte Ethernet de l'ordinateur.

Internet : quelques dates importantes

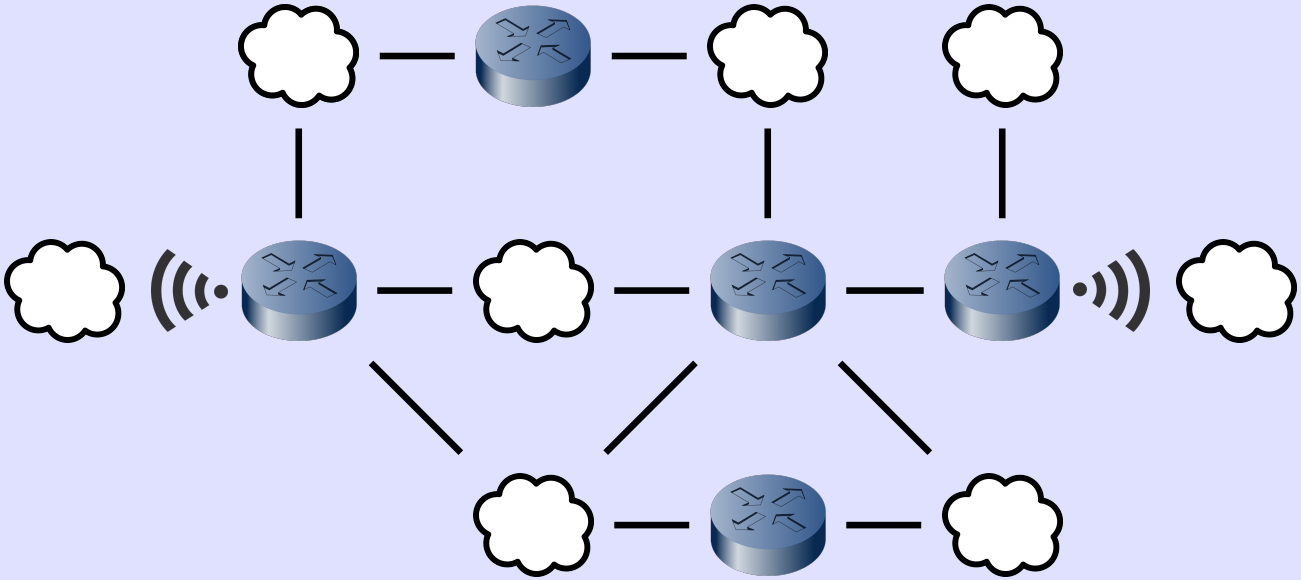
- **1961** — premières recherches sur l'interconnexion de réseaux à Boston au MIT
- **1962** — début de la recherche par ARPA, une agence du ministère de la Défense américain, en collaboration avec des universités états-uniennes
- **1969** — lancement du projet ARPANET, un réseau interconnectant 4 universités états-uniennes
- **1973** — définition du protocole IP
- **1983** — adoption du protocole IP et du mot "Internet"
- **1989** — 100 000 ordinateurs connectés
- **1989–1992** — apparition des premiers fournisseurs d'accès à Internet
- **1996** — 36 millions d'ordinateurs connectés
- **2000** — 368 millions d'ordinateurs connectés
- **2011** — pénurie d'adresses IP : L'IANA annonce qu'elle ne dispose plus d'adresses IP à distribuer.

Différentes organisations/sociétés internationales sont liées au développement et à la gestion de l'Internet.

- **ISOC** — Internet SOCiety
  promouvoir le développement de l'Internet
- **IETF** — Internet Engineering Task Force
  définir et expérimenter les protocoles de l'Internet
- **IANA** — Internet Assigned Number Authority
  gérer les numéros utilisés sur Internet (les adresses IP, par exemple)

Le réseau Internet

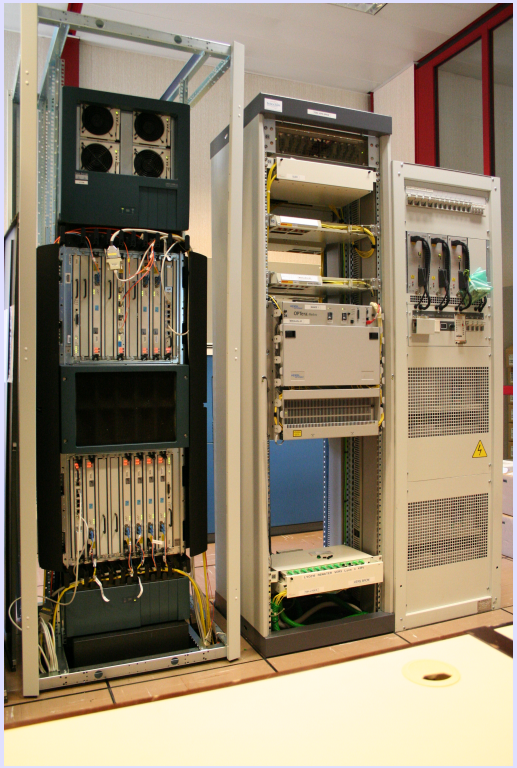
- Internet est une interconnexion de millions de réseaux
  aux caractèristiques différentes (Wifi, Ethernet, satelitte, ...)
  interconnectés par des routeurs
  qui se chargent de faire circuler l'information entre les réseaux.
- routeur () = n'importe quel équipement qui a plusieurs interfaces réseau et qui est à la frontière entre plusieurs réseaux
- réseau () = un ensemble de machines qui peuvent communiquer entre elles sans intermédiaire
  - Ex : des machines sur un même réseau Wifi ou Ethernet



Les routeurs

Plusieurs types d'équipement peuvent jouer le rôle de routeur :

- des équipements dédiés. Ex : un routeur CISCO



- un ordinateur de bureau ou portable
- une box Internet
- un téléphone portable

...

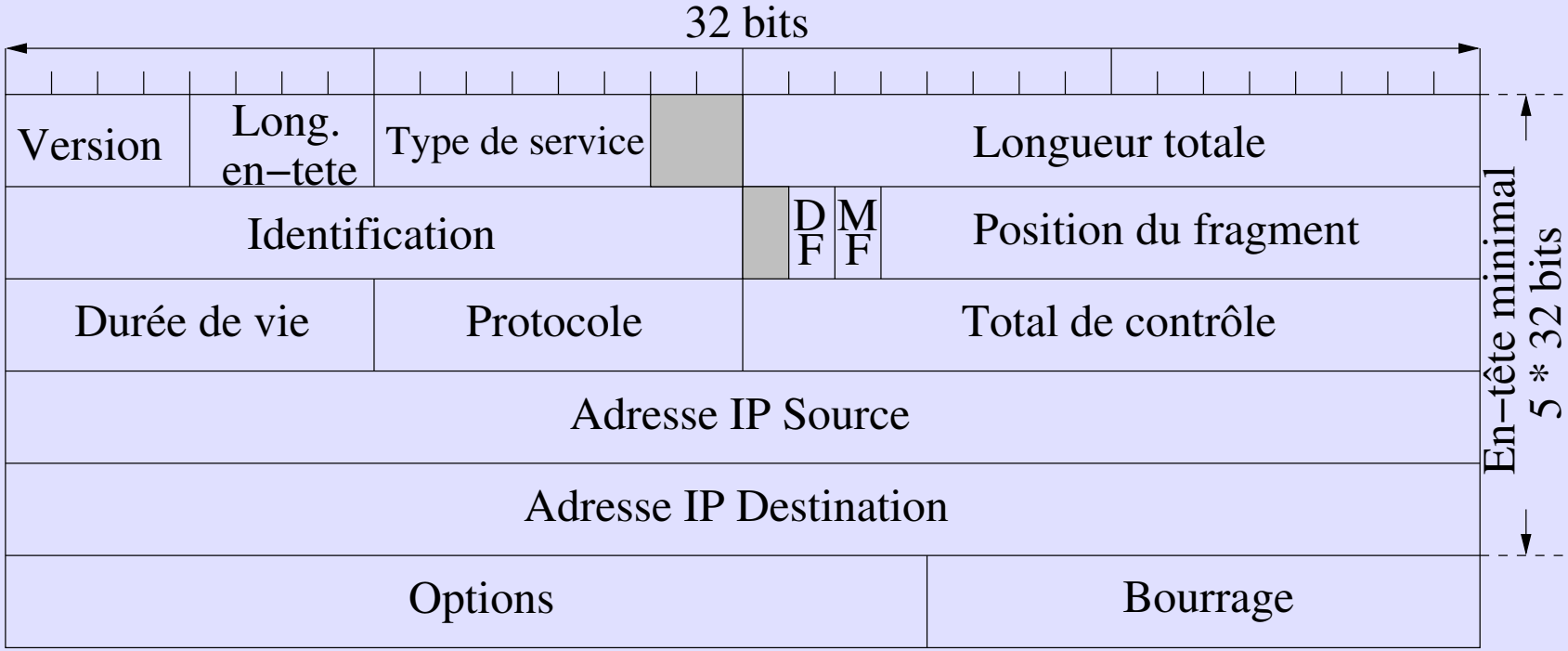
Le protocole IP

- Pour s'échanger des données, les machines connectées à Internet utilise le protocole IP : Internet Protocol.
- IP définit principalement :
  - le mode d'adressage des machines sur le réseau Internet : l'adresse IP ;
  - la structure des messages échangés ;
  - et la façon d'acheminer des messages d'une machine A à une machine B du réseau Internet : le routage.
- Remarque : il existe plusieurs versions du protocole.
  - actuellement : IPv4 (étudiée dans tous les modules de $1^{\text{ère}}$ année)
  - la prochaine : IPv6 (étudiée en $2^{\text{ème}}$ année)

Les paquets IP

- Les messages échangés dans le cadre du protocole IP s'appellent des **paquets**.
- La structure de ces paquets est normalisée : tout paquet IP a une forme bien précise et est constitué
  - d'un en-tête : principalement, les informations nécessaires au routage du paquet ($\Leftrightarrow$ emballage du paquet avec l'adresse)
  - et d'un corps : les informations envoyées par la machine émettrice ($\Leftrightarrow$ contenu du paquet).
- L'en-tête contient, entre autres :
  - l'adresse IP de la machine destinataire (nécessaire pour que le paquet arrive à destination).
  - l'adresse IP de la machine émettrice (nécessaire en cas de réponse du destinataire).

L'en tête-IP



- 20 octets (au moins) apparaissant au début de chaque paquet IP divisés en **champs** donnant des informations sur le paquet
- Exemples :
  - Version : version d'IP utilisée pour le paquet (4 ou 6)
  - Durée de vie : entier diminué de 1 par chaque routeur traversé. Arrivé à 0 le paquet est détruit ($\Rightarrow$ évite qu'un paquet circule indéfiniment).
- Plus de détails dans le module M2103 (Technologies de l'Internet).

Le découpage en paquets

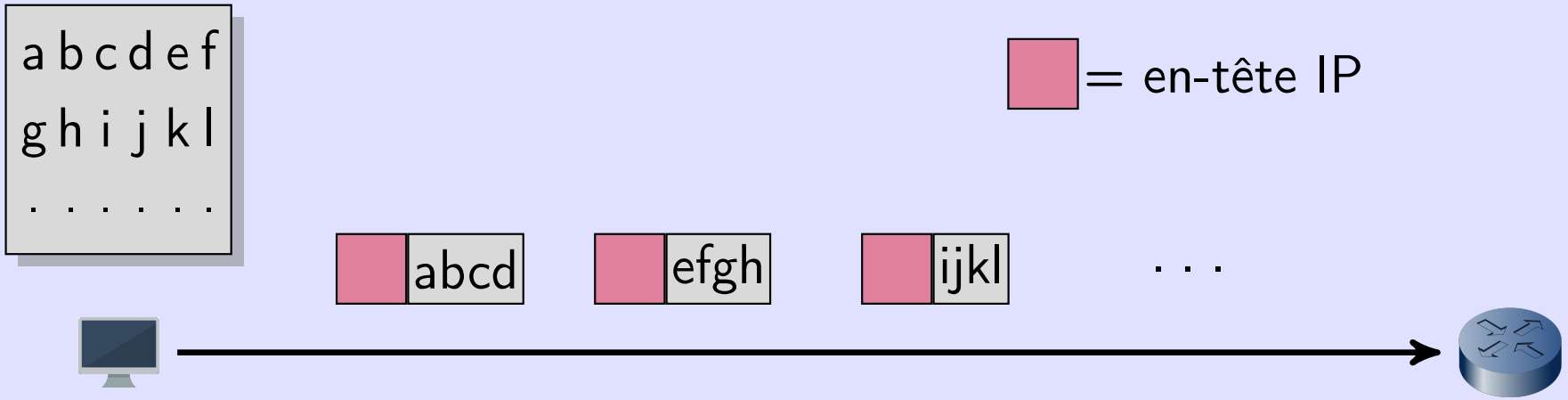
- Pour des raisons techniques la taille des paquets est limitée.

  Ex : taille max≈2 300 octets sur un réseau Wi-Fi.

⇒ Si la taille des données à envoyer > taille max, on doit découper les données en plusieurs paquets.

- C'est ce qu'on appelle la **fragmentation**.
- L'en-tête IP se retrouve dans chaque paquet.

  (Parce que chaque paquet doit arriver à destination.)
- Exemple : envoi d'une suite d'octets `abcdef...`



- *N.B.* En réalité, on peut avoir d'autres en-têtes (d'autres protocoles) dans chaque paquet. Voir le module M1104 (Principes des réseaux).

Structure d'une adresse IP

- Une adresse IP est composée de **32 bits**.
  - On a donc (à peu près) $2^{32} = 4\ 294\ 967\ 296$ d'adresses IP disponibles.
    - "à peu près" car certaines adresses sont nécessaires au fonctionnement du protocole et ne peuvent pas être attribuées à des machines. On dit qu'elles sont **réservées**.
    - $\Rightarrow$ En réalité on a donc un peu moins de 2^{32} adresses IP disponibles.
- On la note sous la forme de 4 octets (en notation décimale) séparés par des points.
  - On parle de notation **décimale pointée**.
  - Rappel : 1 octet = 8 bits
    - $\Rightarrow$ On retrouve bien $4 \times 8 = 32$ bits.
    - $\Rightarrow$ Les nombres qui composent une adresse IP sont dans l'intervalle $[0, 255]$.
- Exemple d'adresse IP :

  10.20.30.40

  dont la représentation binaire est :

  00001010 00010100 00011110 00101000

Anatomie d'une adresse IP

- Toute adresse IP se décompose en deux parties :
  - un identifiant de réseau (ou **net-id**) qui identifie le réseau sur lequel se trouve la machine ;
  - suivi d'un identifiant de machine (ou **host-id**) qui identifie la machine (ou l'hôte) au sein du réseau.
- Le net-id est déterminé par les bits de poids fort de l'adresse IP.
- Le host-id est déterminé par les bits de poids faible de l'adresse IP.
- Mais le nombre de bits utilisés pour représenter le net-id (et donc le host-id) est variable et dépent du réseau sur lequel on se trouve.
- On alors besoin d'une autre information pour pouvoir différencier les bits du net-id de ceux du host-id : le **masque**.

Le masque

- Le masque indique le nombre de bits dans l'adresse IP qui forment le net-id.
- Il est associé à un réseau et donc à toutes les machines de ce réseau.
- On le note comme une adresse IP :
  - dont tous les bits de poids fort correspondant au net-id valent 1 ;
  - et tous les bits de poids faible correspondant au host-id valent 0.
- Un masque est donc une série de bits à 1 suivie de bits à 0.
- Le nombre de bits à 1 permet de déterminer le nombre de bits dans l'adresse IP qui représentent le net-id.
- Exemple : soit l'adresse 10.20.30.40 de masque 255.255.255.0

  En binaire le masque s'écrit 11111111 11111111 11111111 00000000.

  Conséquences
  - Les 24 premiers bits à 1 dans le masque indiquent que les 24 premiers bits (ou 3 premiers octets) de l'adresse IP forment son net-id.
  - L'octet restant est le host-id de la machine.

  On a donc :

$$\underbrace{10.20.30}_{\text{net-id}}.\underbrace{40}_{\text{host-id}}$$

Notation du masque

- Plutôt que de noter le masque complet (avec la notation décimale pointée), on préfère généralement utiliser la notation "/".
- Principe : on note l'adresse IP de la machine suivie de "/N" où N est le nombre de bits à 1 dans le masque de réseau.
- On appelle N la **longueur du masque**.
- Exemple : soit l'adresse 10.20.30.40/24.

  $\Rightarrow$ Dans le masque, il y a 24 bits à 1 suivis de 32 - 24 = 8 bits à 0.

  On a donc :

  10.20.30.40/24

  $\Leftrightarrow$

  10.20.30.40 avec le masque 255.255.255.0

L'adresse de réseau

- Une adresse de réseau désigne un réseau sur Internet.
- Les adresses de réseau sont nécessaires au fonctionnement du routage.
- C'est une adresse réservée : elle ne peut pas être attribuée à une machine du réseau.
- À partir d'une adresse IP et de son masque, on obtient l'adresse de son réseau en mettant les **bits du host-id à 0**.
- Exemple : soit l'adresse 10.20.30.40/24.

  On a vu que l'adresse se décompose ainsi : $\underbrace{10.20.30}_{\text{net-id}}.\underbrace{40}_{\text{host-id}}$

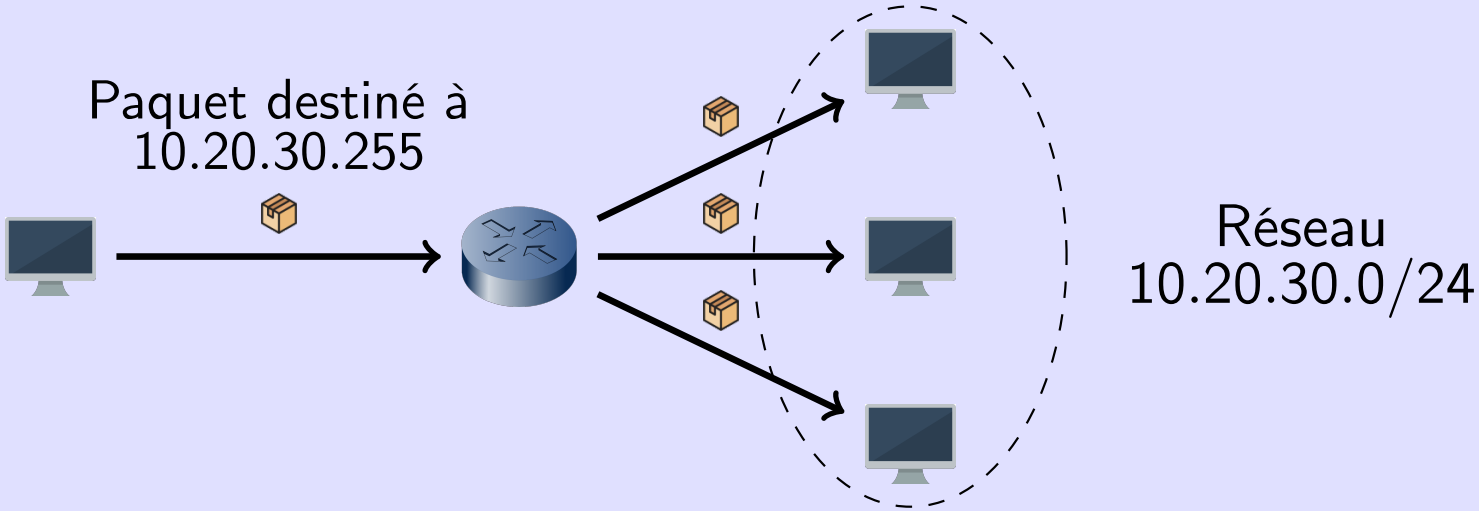
  L'adresse de son réseau est donc : 10.20.30.0
- Remarque : deux machines sur un même réseau doivent forcément avoir la même adresse de réseau.

L'adresse de diffusion de réseau

- L'adresse de diffusion d'un réseau désigne toutes ses machines.
- C'est une adresse réservée.
- Un paquet envoyé à cette adresse est reçu par toutes les machines du réseau.
- À partir d'une adresse IP et de son masque, on obtient l'adresse de diffusion de son réseau en mettant les **bits du host-id à 1**.
- Exemple : soit l'adresse 10.20.30.40/24.

  On a vu que l'adresse se décompose ainsi : $\underbrace{10.20.30}_{\text{net-id}}.\ \underbrace{40}_{\text{host-id}}$

  L'adresse de diffusion de son réseau est donc : 10.20.30.255



L'adresse de diffusion locale

- Dans certaines situations, une machine a besoin d'envoyer un paquet sur son réseau alors qu'elle n'a pas encore d'adresse IP.
- Exemple : au démarrage.
- Elle utilise alors l'**adresse de diffusion locale** pour envoyer un paquet à toutes les machines de son réseau.
- adresse de diffusion locale = 255.255.255.255 (adresse réservée)
- Les paquets envoyés à cette adresse ne sont pas relayés par les routeurs : ils restent sur le réseau.
- adresse utilisée dans le protocole DHCP (voir cours 3) lorsqu'une machine n'a pas encore d'adresse IP

Utilisation du masque

- L'opération effectuée pour connaître l'adresse de réseau à partir d'une adresse IP et d'un masque est le ET logique.
- Cette opération permet de
  - mettre à 0 les bits du host-id ;
  - afin de ne conserver que les bits du net-id.

  autrement dit : de masquer les bits du host-id pour ne garder que ceux du net-id.
- Exemple :
  - Soit l'adresse 10.20.30.40/24.

  $\Rightarrow$ Son adresse de réseau peut être calculée ainsi :

	10	.	20	.	30	.	40
ET	255	.	255	.	255	.	0
	10	.	20	.	30	.	0

= adresse de réseau

ou en binaire :

	00001010	00010100	00011110	00101000
ET	11111111	11111111	11111111	00000000
	00001010	00010100	00011110	00000000

= adresse de réseau

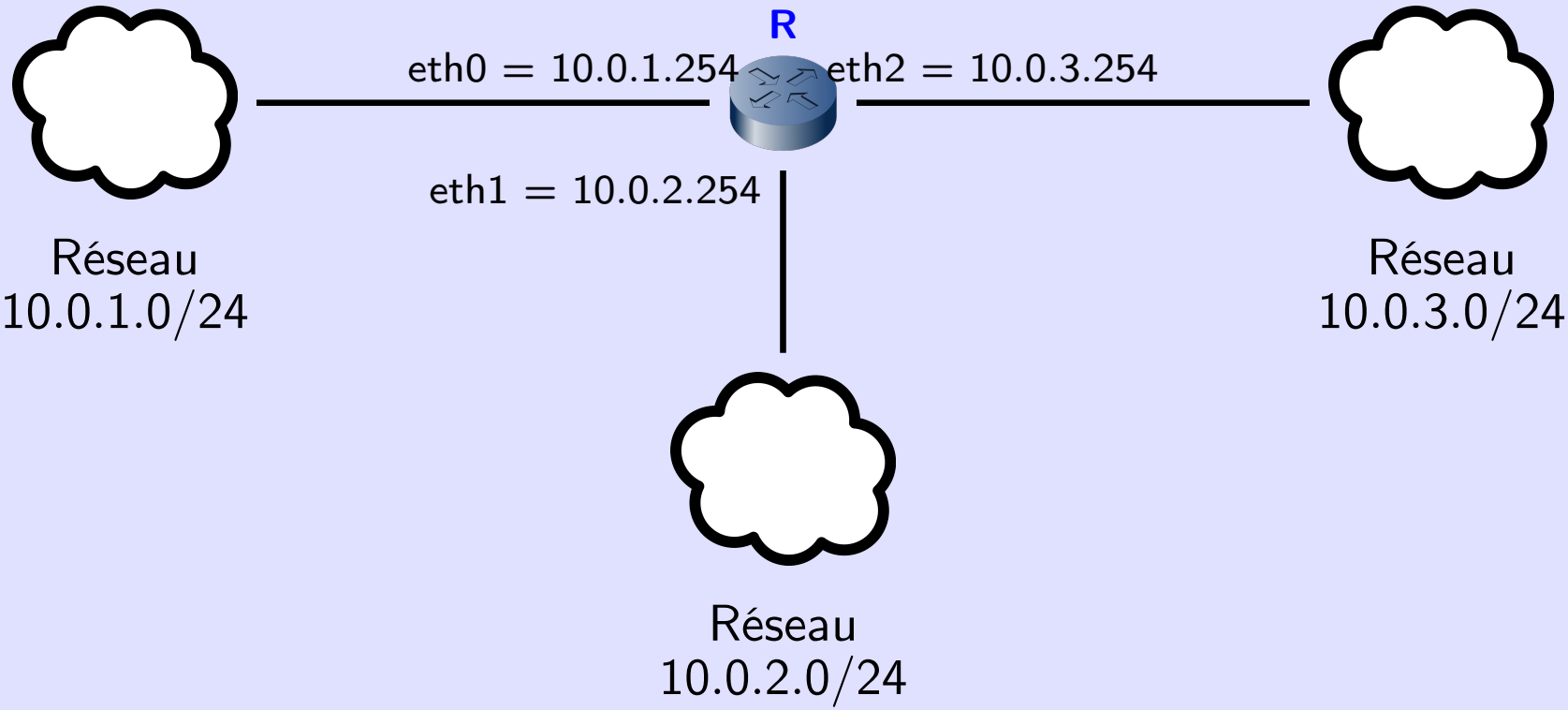
Attribution d'adresses IP

- Règles à respecter lors de l'attribution une adresse IP :
  - net-id cohérent avec l'adresse de réseau
  - ne pas choisir l'adresse de réseau (réservée)
  - ne pas choisir l'adresse de diffusion (réservée)
- Conséquences :
  - La première adresse disponible est (adresse de réseau + 1).
  - La dernière adresse disponible est (adresse de diffusion - 1).
  - Pour un réseau en $/N$, le nombre d'adresses IP disponibles est $2^{32-N} - 2$.
    - $32 - N$ = nombre de bits dans le host-id
    - 2^{32-N} = nombre de host-ids que l'on peut former avec $32 - N$ bits
    - et on retranche les 2 adresses réservées du réseau
- Exemples :

Réseau	$1^{\text{ère}}$ adresse disponible	dernière adresse disponible	Nb. d'adresses disponibles.
10.20.30.0/24	10.20.30.1	10.20.30.254	$2^{32-24} - 2 = 254$
20.30.0.0/16	20.30.0.1	20.30.255.254	$2^{32-16} - 2 = 65\ 534$

Qui possède une adresse IP ?

- Toute machine connectée à Internet doit posséder une adresse IP.
- Mais elle peut en avoir plusieurs.
  C'est le cas, par exemple, des routeurs.
- Un routeur étant connecté à plusieurs réseaux, il est nécessaire qu'il ait une adresse IP par réseau auquel il est connecté.
- On a donc une adresse IP par interface utilisée par le routeur.
- Exemple : un routeur R relié à 3 réseaux par 3 interfaces Ethernet.

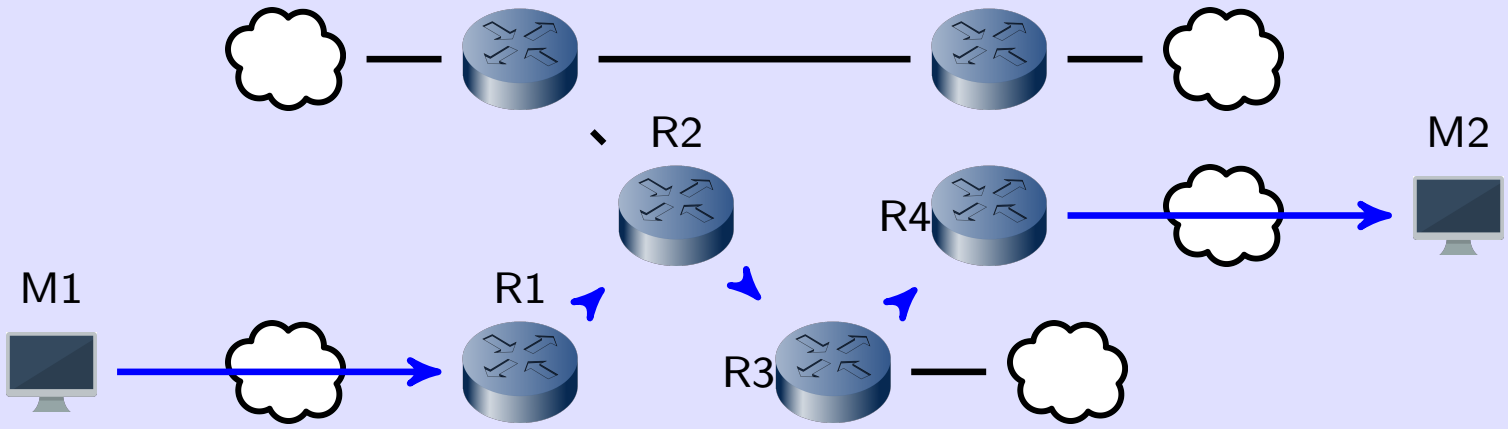


Remarque sur l'utilisation du masque

- Dans les exemples que l'on a vus le masque avait toujours une longueur multiple de 8.
- Comme 8 bit = 1 octet on pouvait directement raisonner sur la notation décimale pointée qui utilise des octets.
- Par contre, si la longueur du masque n'est pas un multiple de 8 il faut **systématiquement** raisonner sur la représentation binaire (et donc traduire en binaire).
- Exemple : quelle est l'adresse du réseau de la machine 10.141.0.1/10 ?
- Voir les exercices de TD.

Définition du routage

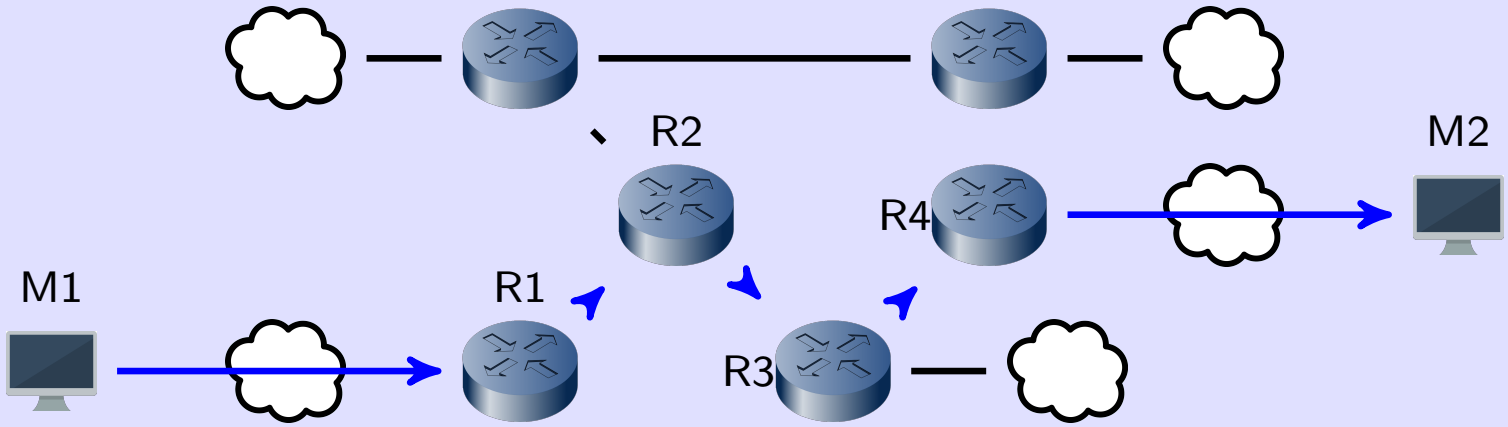
routage = processus de sélection d'une route dans l'acheminement d'un paquet à son destinataire final



Sur Internet :

- Le routage est effectué par tous les routeurs sur le chemin emprunté.
- Les routeurs n'ont pas de carte globale d'Internet. Ils connaissent :
  - les autres routeurs auxquels ils sont connectés ;
  - et les réseaux se trouvant derrière ces routeurs.
- Pour déterminer la route à emprunter pour joindre le destinataire, le routeur consulte une **table de routage**.

Exemple d'acheminement d'un paquet



Situation :

- M1 envoie un paquet à M2
- Le paquet va suivre la route M1→R1→R2→R3→R4→M2

Comment ?

- Chaque routeur, à la réception du paquet, de même que M1 au tout début, va consulter sa table de routage.
- La table de routage va lui permettre de répondre à la question *À qui envoyer un paquet destiné à M2 ?*.
- Dans chaque cas, la table indiquera le routeur suivant sur le chemin.
  - pour M1 ⇒ R1, pour R1 ⇒ R2, ...
- Mais un routeur ne connaît pas le chemin complet suivi par le paquet.
- Par exemple, R1 ne sait pas ce que R2 fera du paquet.

La table de routage

- Elle est consultée par une machine A dès qu'elle doit acheminer un paquet à une machine B et donc déterminer à qui envoyer ce paquet.

  2 situations possibles de consultation :
  - si le paquet est routé par A (cas pour R1, R2, R3 et R4 dans la diapo. précédente)
  - si le paquet est directement envoyé par A (cas pour M1 dans la diapo. précédente)

$\Rightarrow$ Toute machine (qu'elle soit routeur ou non) a une table de routage.

- En consultant sa table de routage, A peut obtenir 3 types de réponse :
  - **remise directe**
  - **remise indirecte**
  - **hôte inaccessible**

Types de réponse

- A = machine devant transmettre le paquet et qui consulte sa table
- B = destinataire du paquet

Remise directe ⇔ A et B sont sur un même réseau

⇒ A n'a pas besoin de passer par un routeur pour envoyer le paquet à B

- La consultation de la table indique :
  - **l'interface** à utiliser pour envoyer le paquet à B

Remise indirecte ⇔ A et B sont sur des réseaux différents

⇒ Il y a (au moins) un routeur entre A et B.

- La consultation de la table indique :
  - **l'adresse IP du routeur R** sur le chemin qui mène à B
  - et **l'interface** à utiliser pour envoyer le paquet à R.

Hôte inaccessible ⇔ la table n'indique pas comment router le paquet

⇒ A ignore le paquet.

Les lignes de la table de routage

- Chaque ligne de la table de routage est un quadruplet **(D,M,R,I)** avec :
  - **D** = Destination
  - **M** = Masque
  - **R** = Routeur
  - **I** = Interface
- Une ligne indique comment envoyer un paquet à une machine du réseau de destination **D** dont le masque est **M**.
- 2 possibilités pour R et I :
  - remise directe ⇒
    - **R** = vide
    - **I** = interface connectée au réseau D
  - remise indirecte ⇒
    - **R** = adresse du routeur sur la route vers le réseau D
    - **I** = interface connectée au réseau sur lequel se trouve R

La route par défaut

- La route par défaut est une ligne de la table de routage utilisée lors de la consultation quand aucune des autres lignes ne permet de déterminer la route vers le destinataire.
  - Les autres lignes indiquent comment joindre certains réseaux.
  - La route par défaut indique comment joindre tous les autres réseaux (c'est-à-dire le reste du réseau Internet).
- Remarques :
  - Pour la route par défaut, on a toujours **D** = **M** = 0.0.0.0.
  - Il ne peut pas y avoir plusieurs routes par défaut dans une table de routage.

Exemple de table de routage

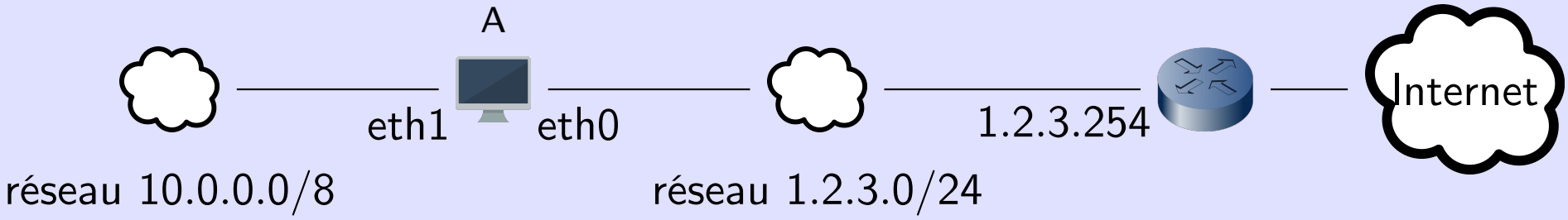
Soit la table de routage d'une machine (routeur) A :

	Destination	Masque	Routeur	Interface
1	1.2.3.0	255.255.255.0	—	eth0
2	10.0.0.0	255.0.0.0	—	eth1
3	0.0.0.0	0.0.0.0	1.2.3.254	eth0

Cette table peut se lire ainsi :

- ligne 1 : pour envoyer un paquet à une machine du réseau 1.2.3.0/24 : **remise directe** sur l'interface **eth0**
- ligne 2 : pour envoyer un paquet à une machine du réseau 10.0.0.0/8 : **remise directe** sur l'interface **eth1**
- ligne 3 (route par défaut) : pour envoyer un paquet à une machine de n'importe quel autre réseau : **remise indirecte** au **routeur** 1.2.3.254 en utilisant l'interface **eth0**

La topologie du réseau est donc la suivante :



Décision de routage

- Soit la situation suivante :
  - A = machine devant transmettre un paquet
  - B = destinataire du paquet
- A va chercher dans sa table de routage la ligne (D, M, R, I) qui indique comment joindre B.
- Cette ligne est telle que B **ET** $M = D$

  autrement dit : on teste si B est sur le réseau D.
- Si une telle ligne existe alors la réponse est :
  - **remise directe** si R est vide
  - **remise indirecte** si R n'est pas vide
- Si aucune ligne ne vérifie cette condition, alors la réponse est :
  - **hôte inaccessible** ($\Rightarrow$ aucune ligne n'indique comment envoyer un paquet à B)

- On a vu que la route par défaut est telle que $D = M = 0.0.0.0$.

$\Rightarrow$ La route par défaut vérifie toujours la condition B **ET** $M = D$.

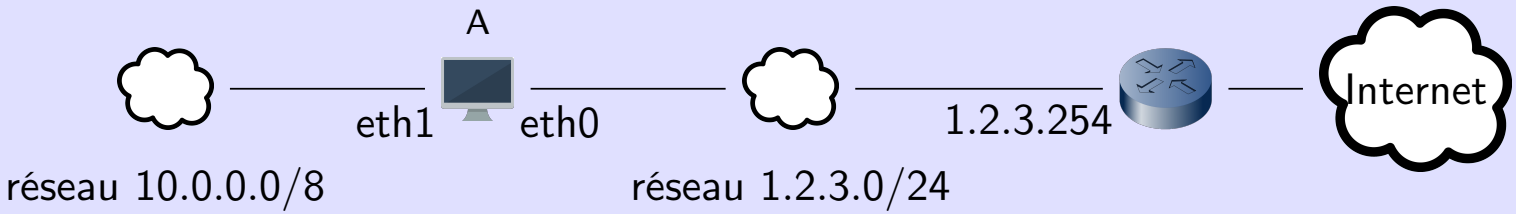
(puisque, quel que soit B, on a toujours, B **ET** $0.0.0.0 = 0.0.0.0$).

- Conséquence : si une route par défaut est présente dans la table, alors la réponse ne sera jamais **hôte inaccessible**.
- Mais la route par défaut est toujours la dernière ligne lue dans la table :

  On cherche d'abord une ligne contenant le réseau de B puis, seulement si on ne trouve pas cette ligne, alors on suit la route par défaut.

Exemples de décisions de routage

	Destination	Masque	Routeur	Interface
1	1.2.3.0	255.255.255.0	—	eth0
2	10.0.0.0	255.0.0.0	—	eth1
3	0.0.0.0	0.0.0.0	1.2.3.254	eth0



Exemple 1 : A doit envoyer un paquet à 10.1.1.2

- ligne 1 : non (car 10.1.1.2 **ET** 255.255.255.0 = 10.1.1.0 ≠ 1.2.3.0)
- ligne 2 : **ok** (car 10.1.1.2 **ET** 255.0.0.0 = 10.0.0.0)

⇒ réponse = **remise directe** du paquet sur **eth1**

Exemple 2 : A doit envoyer un paquet à 24.3.1.7

- ligne 1 : non (car 24.3.1.7 **ET** 255.255.255.0 = 24.3.1.0 ≠ 1.2.3.0)
- ligne 2 : non (car 24.3.1.7 **ET** 255.0.0.0 = 24.0.0.0 ≠ 10.0.0.0)

⇒ réponse = on suit la route par défaut : **remise indirecte** du paquet au **routeur** 1.2.3.254 sur **eth0**

Routage statique et routage dynamique

- Comment les tables de routage des routeurs sont-elles remplies ?
- 2 possibilités : statiquement ou dynamiquement

Routage statique

- table remplie manuellement par l'administrateur du réseau (voir TP)
- OK pour les petits routeurs (interconnectés à peu de réseaux) ou si la topologie (connexions entre les réseaux) évolue peu.

Routage dynamique

- table remplie automatiquement par le routeur lui-même

$\Rightarrow$ aucune configuration manuelle par l'administrateur

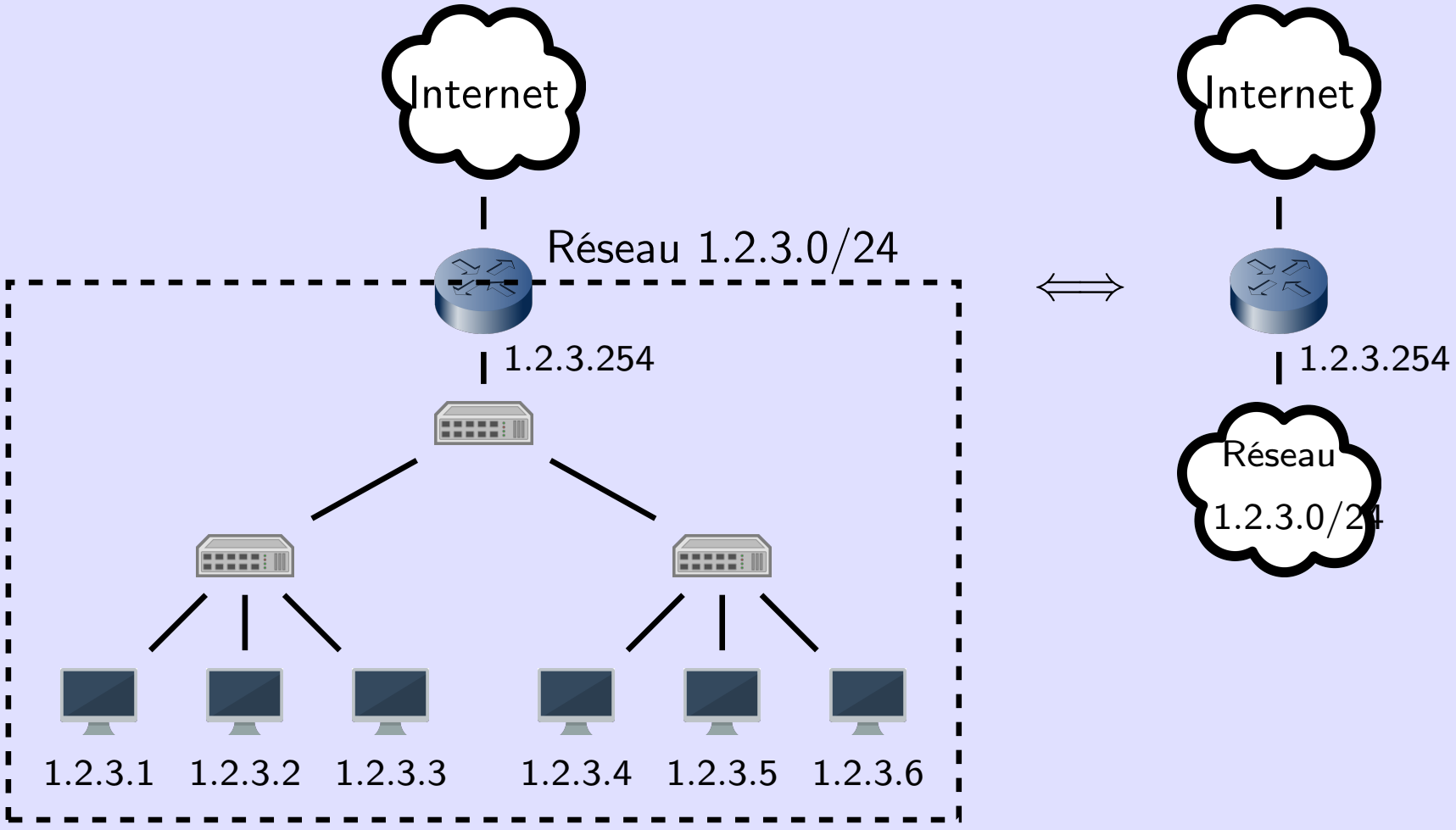
- Comment ? Par l'échange entre routeurs d'informations de routage afin de découvrir de nouvelles routes.
- utilisation de protocoles comme OSPF (voir module M2103)
- plus adapté pour les gros routeurs

Les hubs et switchs

- Ils permettent d'interconnecter des machines.
- Ils ont des **ports** en entrée qui permettent de brancher des machines avec un câble Ethernet.
- Différence entre un hub et un switch :
  - Quand il reçoit des données sur un port, le hub les retransmet sur tous les autres ports
  - alors que le switch les retransmet uniquement sur le port qui permet de joindre la machine destinataire.
- Différence entre un switch/hub et un routeur :
  - Le switch interconnecte des machines d'un **même réseau**.
  - Le routeur interconnecte des machines de **réseaux différents**.
  - Le switch n'a pas d'adresse IP (il n'intervient pas dans le protocole IP).
- Fonctionnement des switchs étudié au $2^{\text{ème}}$ semestre (module M2101).

Exemple de réseau utilisant des switchs

- Un réseau 1.2.3.0/24 composé de switchs () en cascade.
- Architecture classique dans les structures de taille moyenne (ex : à l'IUT).



Câbles RJ45

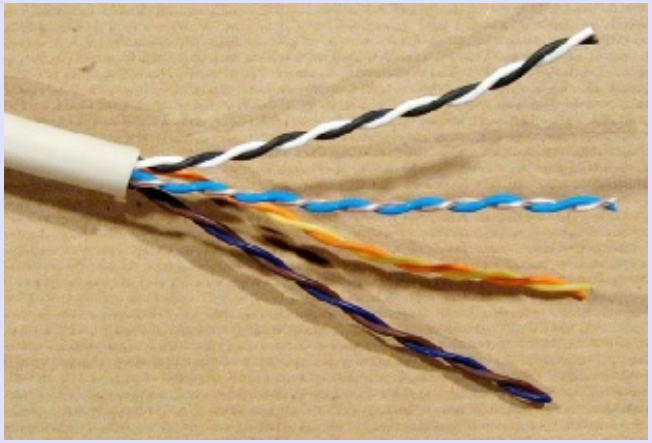
- En Ethernet, les câbles les plus utilisés sont des câbles RJ45.
  - RJ45 = Registered Jack 45 (ou prise enregistrée n°45)
- Dans un câble RJ45 : 8 fils électriques (4 paires torsadées).
- une paire torsadée = 2 fils identifiés par leurs couleurs
- Les cables RJ45 ont d'autres applications que les réseaux informatiques (p.ex., la téléphonie aux USA).
- En ethernet, 4 fils seulement sont utilisés :
  - 2 fils pour la réception de signaux (R^+ et R^-)
  - 2 fils pour la transmission de signaux (T^+ et T^-)
- Pourquoi utiliser 2 fils pour la réception (ou pour l'émission) ?
  - bits = signaux électriques mesurés par des tensions
  - tension = différence de potentiels entre 2 points, donc 2 fils
  - signal de réception = différence de potentiel entre R^+ et R^-
  - signal de transmission = différence de potentiel entre T^+ et T^-

Types d'équipements pour le câblage

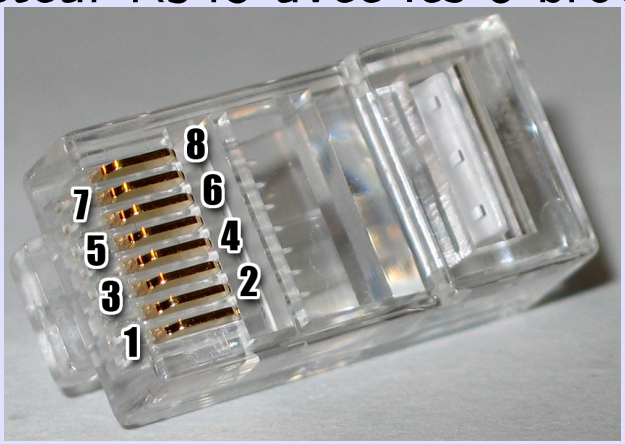
- Lors d'un câblage avec du RJ45 on distingue 2 types d'équipements :
  - DCE = Data Communication Equipment = équipements d'interconnexion dans un réseau (hubs et switchs)
  - DTE = Data Terminal Equipment = équipements terminaux d'un réseau (routeurs et ordinateurs)
- Selon le type, les broches utilisées ne sont pas les mêmes :

Broche	1	2	3	4	5	6	7	8
DCE	R^+	R^-	T^+	-	-	T^-	-	-
DTE	T^+	T^-	R^+	-	-	R^-	-	-

Un câble RJ45 dénudé :

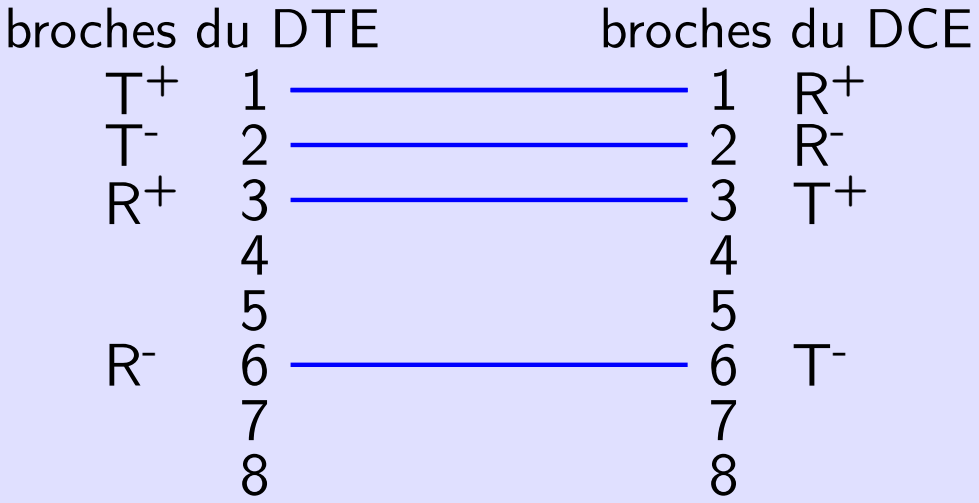


Connecteur RJ45 avec les 8 broches :



Câbles droits et câbles croisés 1/2

- Comment câbler deux équipements ?
- Pour que la transmission fonctionne, il faut que les broches affectées à la transmission d'un côté soit reliées aux broches affectées à la réception de l'autre côté.
  - T^+ de l'un relié au R^+ de l'autre
  - T^- de l'un relié au R^- de l'autre
  - et inversement
- Pour relier un DTE à un DCE, on utilise donc un **câble droit** (qui relie chaque broche d'un côté à la broche de même numéro de l'autre côté).
- On obtient alors le schéma suivant :



Câbles droits et câbles croisés 2/2

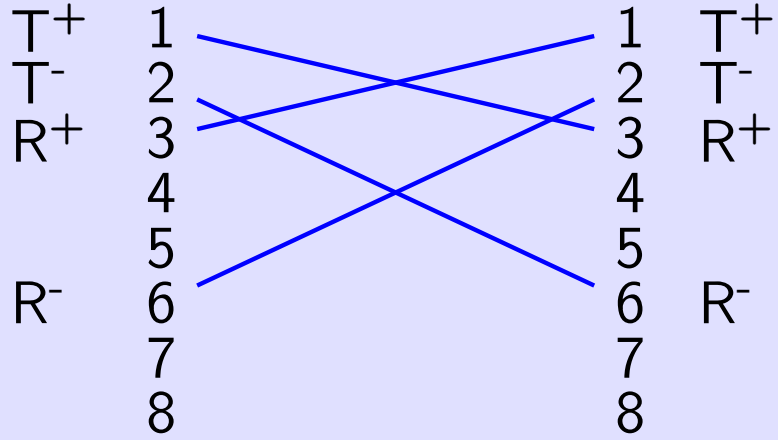
- Par contre, pour relier deux DTE ou deux DCE, le câble droit n'est pas adapté.

  (On obtiendrait alors le T^+ de l'un relié au T^+ de l'autre, le T^- de l'un relié au T^- de l'autre et ainsi de suite.)

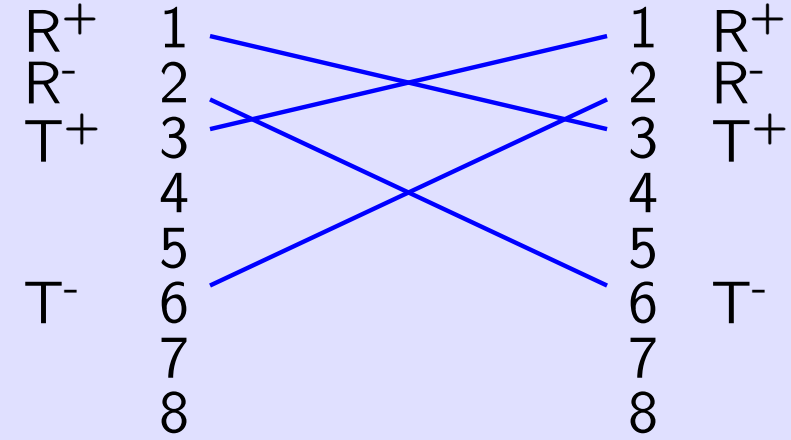
- On utilise alors un **câble croisé** qui relie :
  - la broche 1 de l'un à la broche 3 de l'autre
  - la broche 2 de l'un à la broche 6 de l'autre
  - et inversement

- On obtient alors l'un des deux schémas suivants :

Pour relier deux DTE :



Pour relier deux DCE :



- On a bien alors le T^+ de l'un relié au R^+ de l'autre, le T^- de l'un relié au R^- de l'autre et ainsi de suite.

Quel type de câble utiliser ?

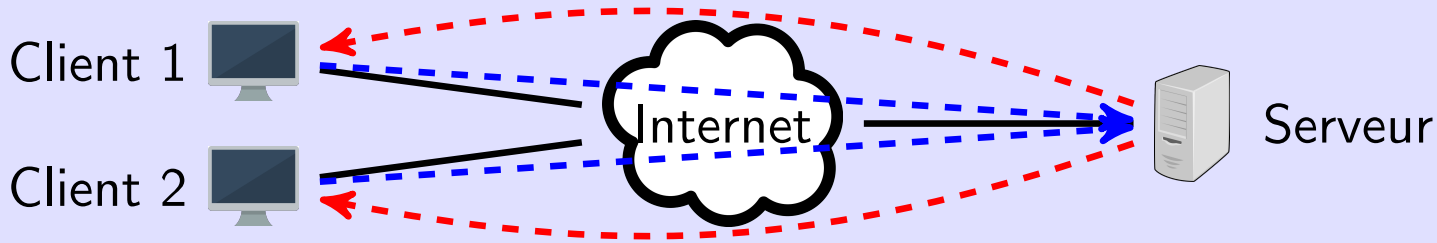
En résumé :

		DCE		DTE	
		Hub	Switch	Routeur	Ordinateur
DCE	Hub	croisé	croisé	droit	droit
	Switch	croisé	croisé	droit	droit
DTE	Routeur	droit	droit	croisé	croisé
	Ordinateur	droit	droit	croisé	croisé

Plan

Le modèle client-serveur

- Famille de protocoles qui font intervenir
  - des processus qui proposent des services (les **serveurs**)
  - à des processus qui peuvent en bénéficier (les **clients**)
  - et qui s'exécutent sur différentes machines d'un réseau.
- Rappel : 1 processus = 1 programme en cours d'exécution
- Exemples de services : télécharger un fichier, envoyer un mail, ...
- Échanges sous forme de
  - questions (ou requêtes) du client
  - suvies de réponses du serveur



- Remarques :
  - Le terme "client" désignera à la fois le processus client et la machine sur laquelle ce processus s'exécute. idem pour le serveur
  - C'est toujours le client qui est à l'initiative de l'échange : un serveur n'enverra jamais de messages à un client si celui-ci ne l'a pas sollicité.

Définition

Un processus qui attend du réseau des demandes ou requêtes en provenance de processus clients et traite ces requêtes puis, généralement, envoie au client une réponse.

Remarques

- Quand le service n'a rien à faire (il ne reçoit aucune requête du réseau) on dit qu'il est en **sommeil**.
  - Il est réveillé par le système d'exploitation à la réception d'une requête.
- Les services doivent être lancés par l'utilisateur root.
- Le client et le serveur utilisent le même protocole de service pour communiquer.
- Exemples de protocoles de service :
  - HTTP, FTP : transfert de fichiers
  - SMTP : envoi de mails
  - POP, IMAP : réception de mails

- Plusieurs commandes selon la version (distribution) de linux.
- Pour la distribution mageia (utilisée en TP) :

```
systemctl <action> <nom-du-service>.service
```

- Actions courantes :
  - start = démarrer le service
  - stop = arrêter le service
  - status = voir l'état du service (arrêté/démarré, temps d'exécution, ...)
- Souvent, on a `<nom-du-service>` = nom du protocole suivie de 'd' (pour démon).
- Exemple d'utilisation :

```
systemctl start sshd.service
```

⇒ lance le serveur SSH qui permettra à des clients de se connecter à distance sur la machine

Interaction avec le protocole IP

- Dans le cours 2 on a vu que sur Internet l'information circule dans des paquets IP.
- un paquet = un en-tête IP (adresses IP source et dest., ...) + un corps (l'information transportée)
- Dans le cas des protocoles de service, le corps du paquet = données du protocole de service.
- On dit que les données du protocole de service sont **encapsulées** dans le paquet IP.

Problématique

- Quand deux machines communiquent sur Internet, elles utilisent leurs adresses IP uniquement.
- L'adressage IP est utilisé par toutes les protocoles sur l'Internet (web, messagerie, transfert de fichiers, ...).
- Or, une adresse IP est difficile à mémoriser pour un utilisateur humain.

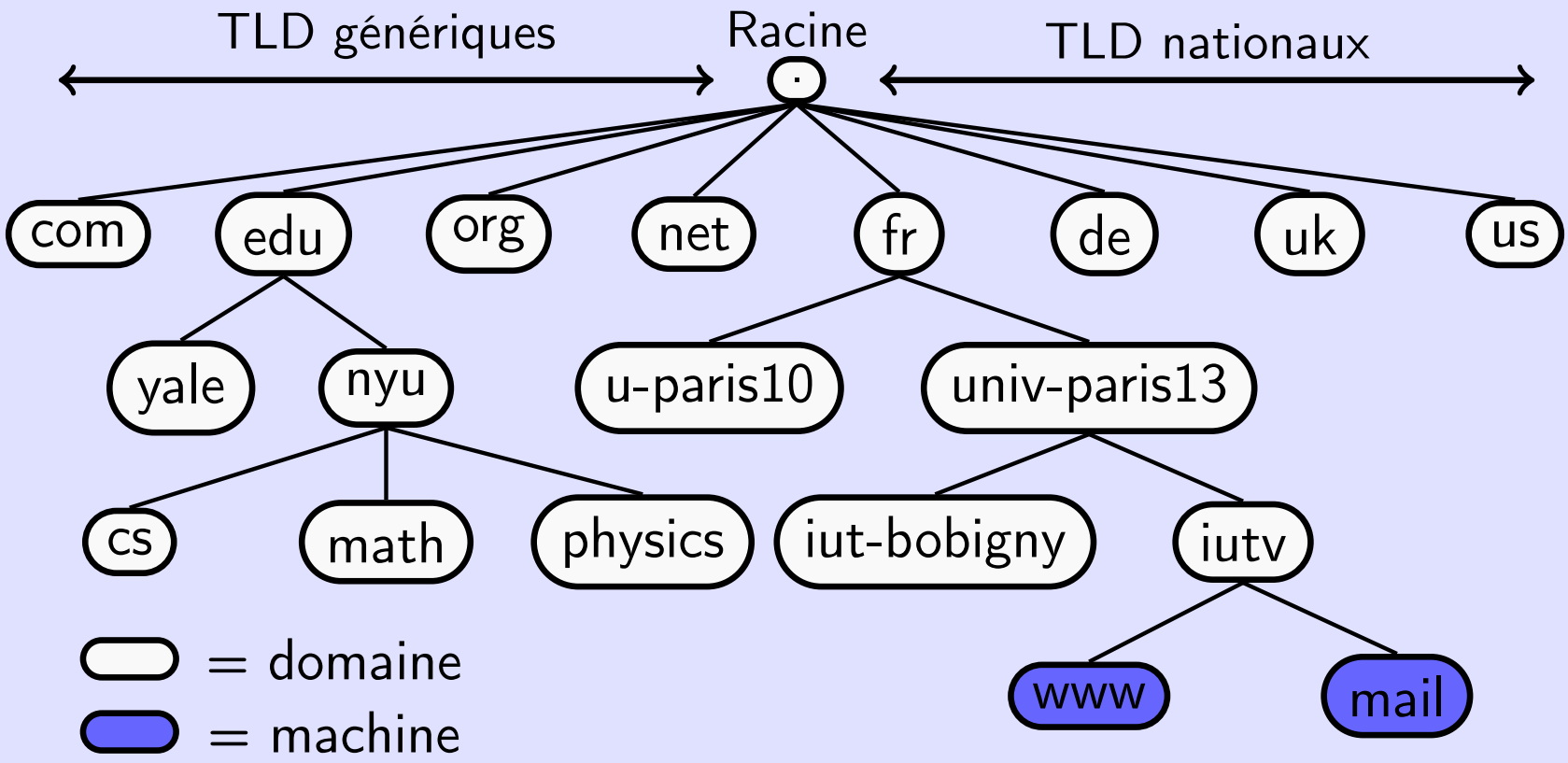
Conséquences

- besoin d'un second mode d'adressage plus pratique pour les utilisateurs
  $\Rightarrow$ Le **nom de domaine** ou **nom symbolique**.
- besoin d'un mécanisme de correspondance automatique entre adresse IP et nom symbolique
  $\Rightarrow$ Le protocole DNS.

Le nom de domaine

- noms de domaines gérés par l'IANA
- système de nommage **hiérarchique** comparable à celui utilisé pour les fichiers sous Linux
- au premier niveau : les TLD (**T**op **L**evel **D**omain) répartis en
  - TLD génériques (edu, com, net, ...)
  - et TLD nationaux (fr, de, jp, ...)
- exemple : pour la machine `www.exemple.fr` :
  - `fr` = le TLD
  - `exemple.fr` = le domaine de la machine
  - `www` = le nom de la machine sur ce domaine
- noms insensibles à la casse (`exemple.fr` = `ExEMpLe.Fr`)

Représentation arborescente de l'espace des noms



- à la racine de l'arbre : le domaine .
- Un nom se lit en allant du bas vers le haut de l'arbre.
- les nœuds de l'arbre = domaines divisés en sous-domaines
  - `nyu.edu` est un domaine avec 3 sous-domaines.
- Les machines (ou hôtes) sont toujours des feuilles de l'arbre.
  - `mail.iutv.univ-paris13.fr` = nom du serveur de mail de l'IUT

Caractéristiques du protocole DNS

- DNS = **D**omain **N**ame **S**ystem
- protocole défini en Novembre 1978
- Un serveur DNS fonctionne comme un annuaire :
  - Le client fournit le nom symbolique d'une machine avec laquelle il souhaite communiquer.
  - Le serveur lui donne l'adresse IP correspondant à ce nom.
- On parle de **résolution d'adresse**.
- Exemple d'utilisation de DNS :
  - Un utilisateur ouvre un navigateur à la page `http://www.mon-site.fr`.
  - Avant de contacter `www.mon-site.fr` le navigateur doit interroger un serveur DNS pour obtenir son adresse IP.
- Dès qu'un utilisateur (ou un programme) souhaite communiquer avec une machine dont il a uniquement le nom il y aura forcément une résolution DNS pour obtenir son adresse IP.

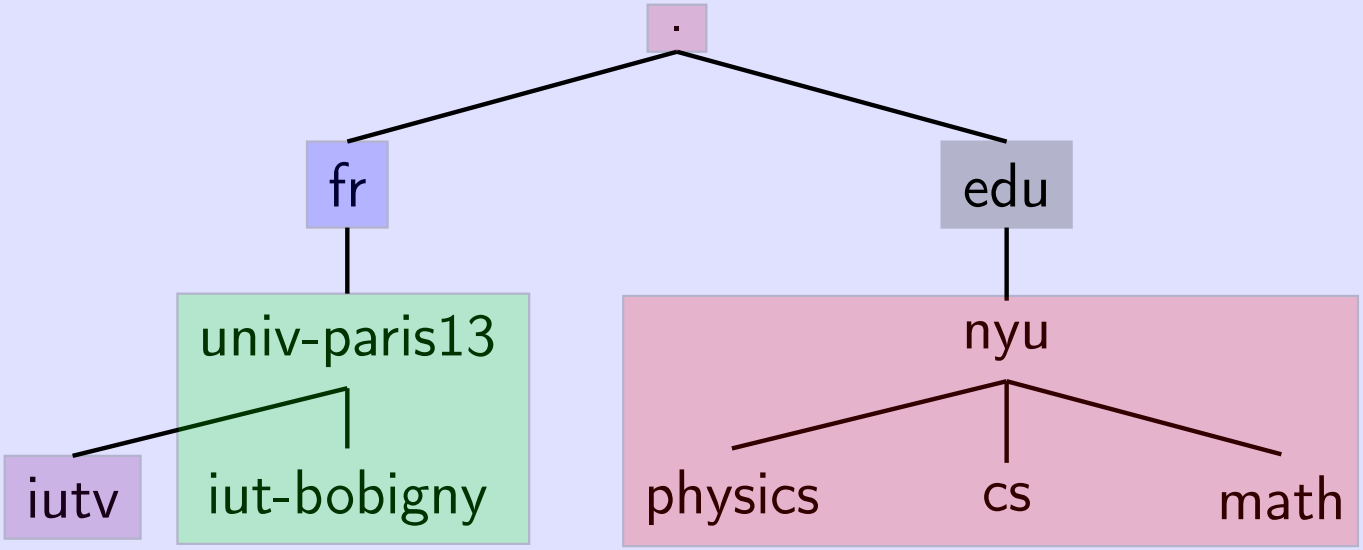
Difficulté de la résolution

- En théorie, un serveur DNS pourrait mémoriser les adresses IP de toutes les machines de l'Internet.
- Impossible en pratique : trop d'adresses IP à mémoriser.
- Pour résoudre ces problèmes :
  - La base de données des adresses IP est répartie sur des millions de serveurs DNS.
    - $\Rightarrow$ Chaque serveur DNS possède un petit fragment de la base de données complète.
  - La répartition des adresses se fait selon une division en zones.

La division en zones

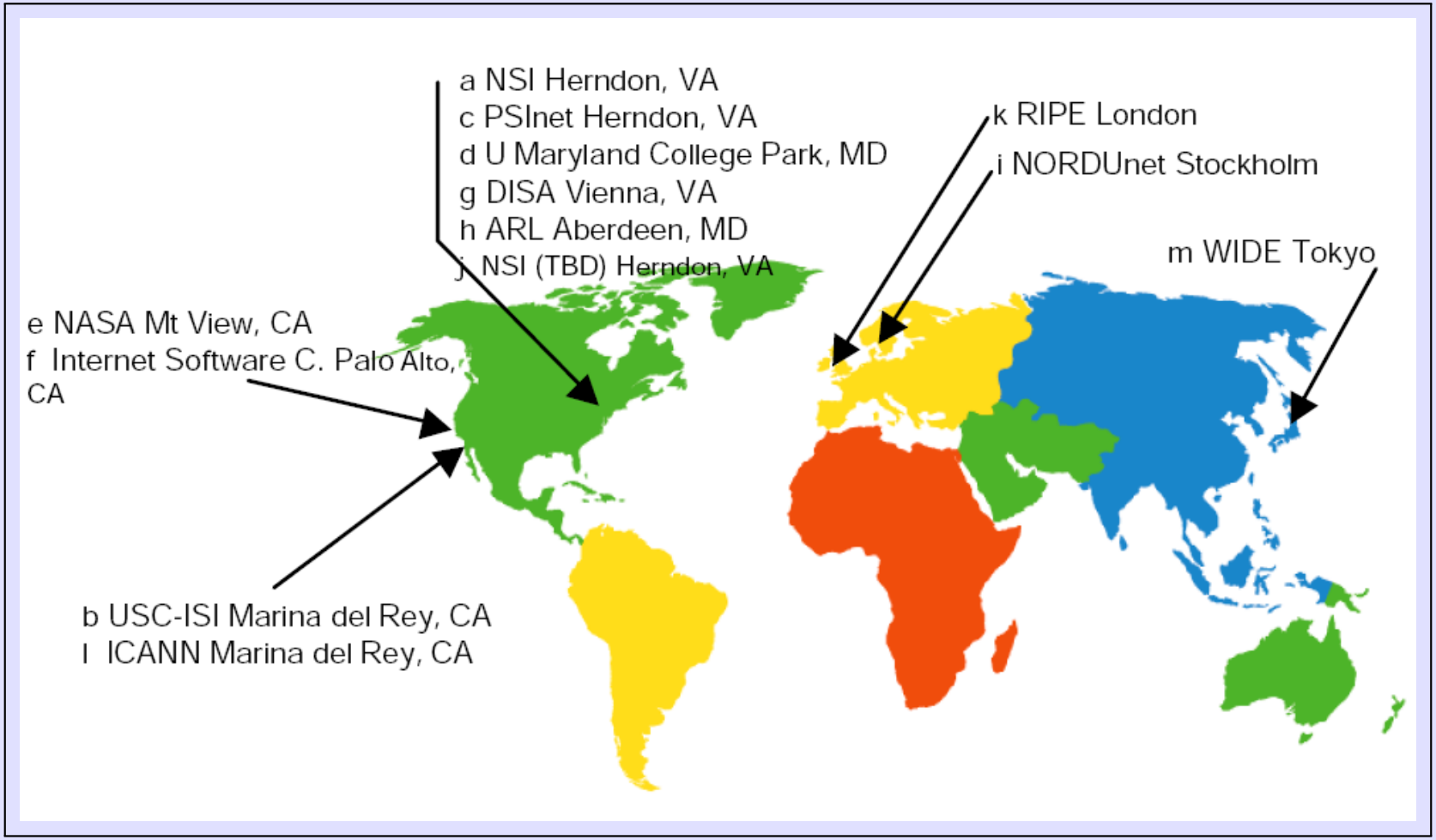
- Pour simplifier la résolution, l'espace de noms est divisé en **zones**.
  - 1 zone = 1 domaine + 0 à N sous-domaines
  - 1 zone = portion contigüe de l'arbre
- Une zone a un serveur DNS **primaire**.
- On dit que c'est le serveur qui fait **autorité** dans cette zone.
- Il a les informations exactes sur les machines et domaines de cette zone.
- C'est le serveur auquel on doit s'adresser pour trouver une IP
  - dans la zone ;
  - ou dans une des sous-zones.
- Il peut trouver l'IP d'une machine :
  - directement : il l'a dans sa base (si la machine est dans sa zone) ;
  - ou indirectement : il connaît l'IP d'un serveur qui pourra la trouver.
- les adresses IP des autres serveurs qu'un serveur doit connaître :
  1. les primaires des 13 serveurs racines (voir la suite)
  2. les primaires des zones qui se trouvent juste en-dessous de la sienne

La division en zones — Exemple



- Le DNS faisant autorité sur la zone **univ-paris13.fr** connaît
  - les IP des 13 serveurs DNS racine ;
  - les IP des machines du domaine **univ-paris13.fr** ;
  - les IP des machines du sous-domaine **iut-bobigny.univ-paris13.fr** ;
  - l'IP du serveur faisant autorité sur **iutv.univ-paris13.fr**.
- Le DNS faisant autorité sur la zone **fr** connaît
  - les IP des 13 serveurs DNS racine ;
  - l'IP du serveur faisant autorité sur **univ-paris13.fr** ;
  - mais pas celle du serveur faisant autorité sur **iutv.univ-paris13.fr**.

Les 13 serveurs DNS racine



- 13 serveurs qui font autorité sur la zone racine (.).
- Ils connaissent les adresses IP des serveurs qui font autorité sur les TLD (les serveurs DNS primaires des zones `fr`, `de`, `com`, ...).

Comment fait un serveur DNS pour répondre à une demande de résolution :

Quelle est l'adresse IP de la machine machine.dom1.dom2.fr ?

2 réponses possibles du serveur :

- réponse positive contenant l'IP demandée
- réponse négative contenant l'IP d'un autre serveur DNS à interroger

Cas 1 : je fais autorité sur `dom1.dom2.fr`

réponse : positive + IP de machine.dom1.dom2.fr

Cas 2 : je ne fais pas autorité sur `dom1.dom2.fr` mais à un niveau supérieur (sur `dom2.fr`, ou `fr`)

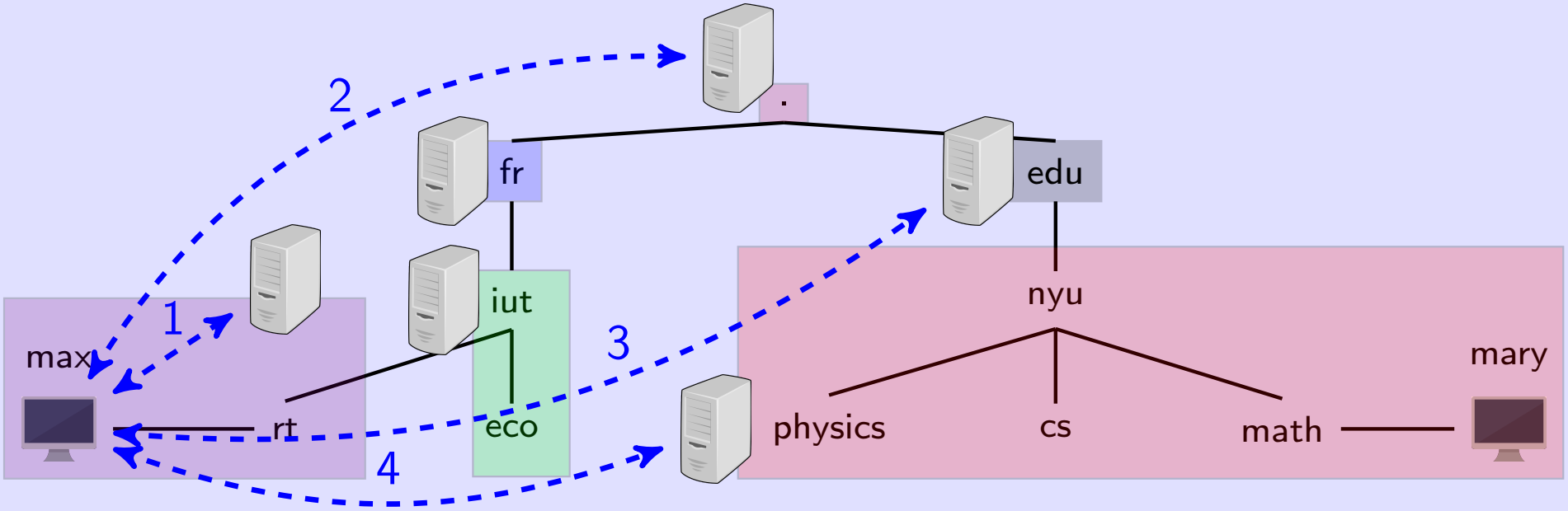
réponse : négative + IP du serveur DNS qui fait autorité au niveau inférieur

Cas 3 : dans tous les autres situations

réponse : négative + IP d'un serveur DNS racine

Protocole de résolution de nom — Exemple

- **max.rt.iut.fr** veux connaître l'adresse IP de **mary.math.nyu.edu**.
- Il demande d'abord à son serveur DNS.



1. Cas 3 : le DNS de **rt.iut.fr** renvoie l'adresse IP d'un serveur DNS racine
2. Cas 2 : le DNS racine renvoie l'adresse IP du serveur DNS de **edu**
3. Cas 2 : le DNS de **edu** renvoie l'adresse IP du serveur DNS de **nyu.edu**
4. Cas 1 : le DNS de **nyu.edu** renvoie l'adresse IP de **mary.math.nyu.edu**

Côté client

- `/etc/hosts` : correspondances nom↔adr. IP déjà connues.
- `/etc/resolv.conf` : adr. IP du serveur DNS de la machine (serveur interrogé dès qu'un programme veut faire une résolution)
- `/etc/nsswitch.conf` : méthode à suivre pour obtenir une adr. IP

Commandes d'interrogation d'un serveur DNS : nslookup, host, dig.

Côté serveur : les fichiers de zone

- Fichier contenant les informations sur la zone d'autorité du serveur.
- Exemple de fichier pour la zone `iutv.fr` :

```
; adresses IP de www.iutv.fr et de toto.iutv.fr
www                     A       1.2.3.1
toto                    A       1.2.3.2
; définition de la zone rt.iutv.fr dont le serveur DNS
; est nom.rt.iutv.fr d'adresse IP 2.3.4.5
rt.iutv.fr.             NS      nom.rt.iutv.fr.
nom.rt.iutv.fr.         A       2.3.4.5
```

- SSH = Secure SHell (terminal sécurisé)
- SSH permet d'ouvrir à distance une connexion sécurisée sur une autre machine.
- sécurisée = SSH chiffre toutes les données envoyées sur le réseau.
- On peut ensuite exécuter des commandes sur la machine distante.

$\Rightarrow$ très utilisé par les administrateurs pour intervenir à distance

- SSH est à la fois un protocole de communication et un programme de connexion.

- 2 machines sont utilisées :
  - la machine **serveur** = celle sur laquelle on veut ouvrir une connexion
  - la machine **cliente** = celle depuis laquelle on se connecte
- Conditions :
  - l'utilisateur doit avoir un compte sur le serveur
  - le service sshd doit être démarré sur le serveur

    sshd gère les demandes de connexion distantes faites par les clients.
- Étapes :
  1. sur le serveur : démarrer le service sshd
  2. sur le client : utiliser la commande ssh pour ouvrir une connexion

La commande SSH

- Commande à utiliser sur la machine cliente pour ouvrir la connexion.
- Syntaxe :

```
ssh <login>@<machine>
```

- login = compte (sur la machine serveur) utilisé pour ouvrir la connexion
- machine = nom DNS ou adresse IP de la machine serveur

Exemple :

```
sami@q20514 $ pwd
/home/sami
sami@q20514 $ ssh evangelista@p20301
Password:
evangelista@p20301 $ pwd
/home/evangelista
```

Configuration réseau d'une machine

- Pour pouvoir communiquer sur Internet une machine a besoin de :
  - une adresse IP + le masque de son réseau
  - connaître l'adresse IP du routeur qui le relie au monde extérieur

    Sinon elle peut uniquement communiquer avec les machines de son réseau.
  - connaître l'adresse IP d'un serveur DNS

    Sinon impossible de faire de la résolution de noms.
- Ces informations forment une **configuration IP**.
- 2 modes de configuration des machines :
  1. **statique** = c'est l'administrateur du réseau qui
     - choisit une adresse IP et l'écrit dans le fichier de configuration de l'interface réseau
     - renseigne les tables de routage des machines (commande route)
     - indique l'adresse IP d'un serveur DNS dans `/etc/resolv.conf`
  2. **dynamique** = c'est la machine qui obtient elle-même cette configuration en interrogeant un serveur **DHCP**
     - DHCP = Dynamic Host Configuration Protocol
- Avantages de la solution dynamique :
  - moins d'erreurs (p.ex. : donner la même adresse IP à 2 machines)
  - facilité de maintenance (un seul fichier de configuration DHCP à modifier)

Client et serveur DHCP

- client = une machine qui veut obtenir une configuration IP
- serveur = la machine qui va lui fournir cette configuration
- Les deux machines doivent se trouver sur le même réseau.
- Quand a lieu l'échange entre le client et le serveur ?
  - à la connexion du client à un nouveau réseau (ex : pour les ordinateurs portables)
  - au démarrage du client
- Le serveur possède
  - une base de données d'adresses IP qu'il peut attribuer aux clients
  - et toutes les informations sur le réseau nécessaires pour pouvoir communiquer sur Internet.
- Ces informations sont stockées sur le serveur dans un fichier de configuration DHCP (`/etc/dhcp/dhcpd.conf` généralement)

Fonctionnement du protocole DHCP (1/2)

1 — La demande (Client → 255.255.255.255)

- Initialement, le client ne connaît pas l'adresse IP de son serveur DHCP.

⇒ Il envoie un message **Discover** à toutes les machines du réseau (en utilisant l'adresse de diffusion locale 255.255.255.255).

- Ce message est ignoré par toute machine qui n'est pas serveur DHCP.
- but du Discover : demander une configuration IP à un serveur DHCP

2 — La réponse (Serveur → Client)

- Après réception d'un message Discover, le serveur détermine la configuration du client et lui attribue une adresse IP.
- Il envoie ces informations dans un message **Offer**.

Fonctionnement du protocole DHCP (2/2)

3 — L'acceptation/refus (Client → 255.255.255.255)

- Le client qui reçoit une configuration peut l'accepter ou la refuser.
- Exemple de refus : le client a déjà accepté une autre configuration (dans le cas où il y a plusieurs serveurs DHCP sur le réseau).
- Dans tous les cas, il envoie un message **Request** à 255.255.255.255 pour que tous les serveurs DHCP connaissent sa décision.

4 — La confirmation (Serveur → Client)

- Le serveur dont la proposition a été acceptée envoie un **Ack** pour confirmer au client qu'il peut utiliser la configuration proposée.

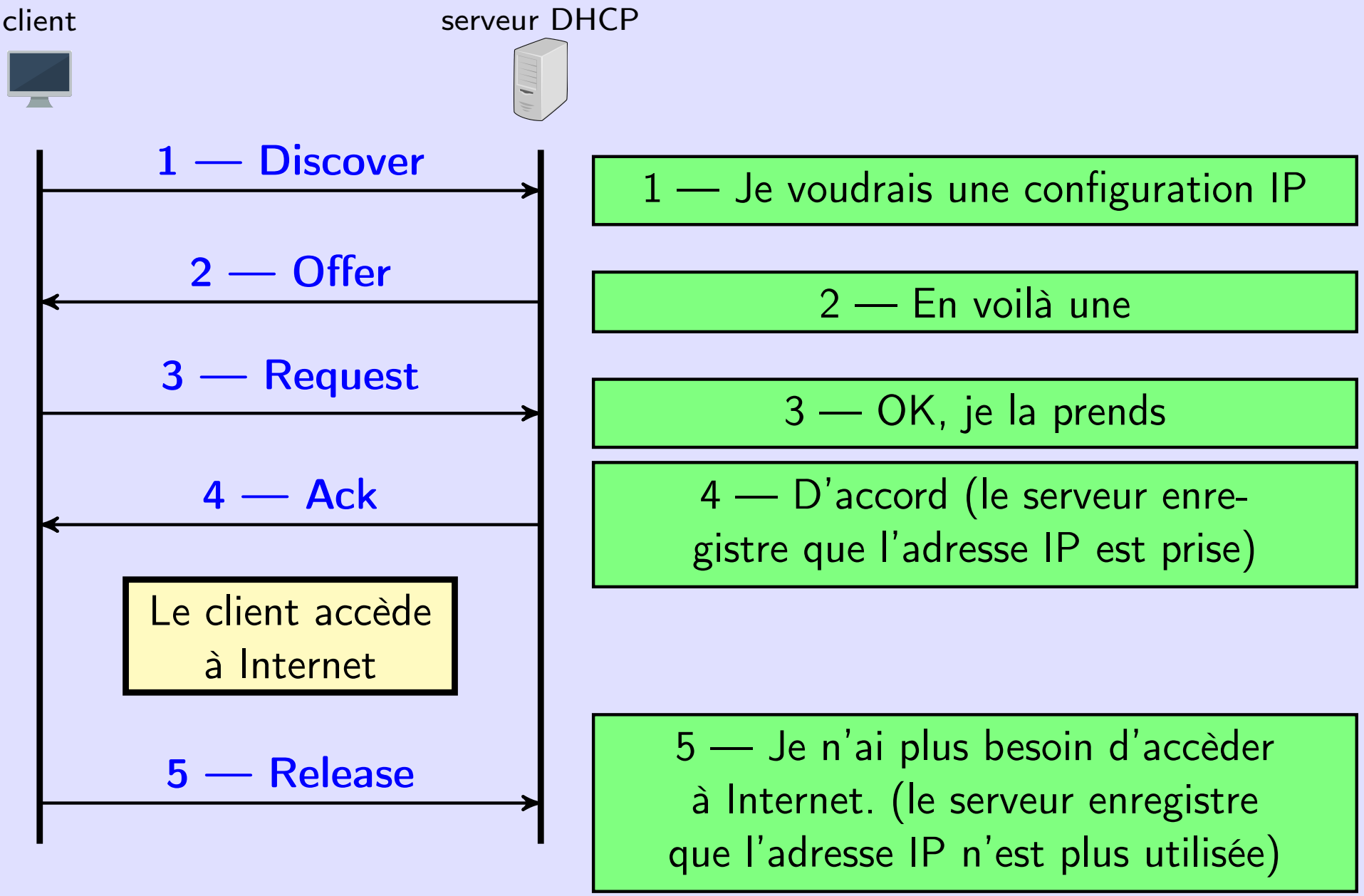
  (Ack = Acknowledgment, acquittement)

5 — La libération (Client → Serveur)

- Dès que le client n'a plus besoin d'accéder à Internet, il envoie au serveur un message **Release**.

⇒ L'adresse IP qui était utilisée par le client devient libre et peut être utilisée par une autre machine.

Exemple de scénario DHCP



- Toute attribution d'une configuration IP a une durée de validité.
- Cette durée est fixée par le serveur à l'attribution.
- C'est le principe du **bail DHCP**.
- Avant l'expiration du bail, le client peut demander un prolongement en retransmettant un message **Request**.
- Plusieurs cas possibles :
  1. le serveur accepte le prolongement ⇒ il répond avec un message **Ack** au client qui peut continuer à utiliser la configuration IP
  2. le serveur refuse ⇒ il répond avec un message **Nak** au client
     - 2.1 Le client peut interroger d'autres serveurs (en envoyant d'autres **Request**) jusqu'à ce qu'un serveur lui réponde positivement avec un **Ack** ;
     - 2.2 ou il abandonne et ne peut plus accéder à Internet.

Problèmes liés à la mobilité des utilisateurs

- Dans beaucoup d'organisations, les utilisateurs n'ont pas de machine attribuée : ils peuvent se connecter sur n'importe quelle machine.
  - Exemple : à l'IUT avec votre compte étudiant.
- Problèmes à résoudre :
  - Il faut que l'utilisateur puisse retrouver ses données quelle que soit machine sur laquelle il se connecte.
  - Il faut que son identifiant et son mot de passe soient reconnus sur toutes les machines.
- Une solution : centraliser les données.

La centralisation des données

- Les données (les fichiers ou mots de passe, par exemple) sont stockées sur un serveur (unique).
- Quand un utilisateur lit (ou modifie) ces données, elles ne sont pas lues (ou modifiées) en local (sur sa machine) mais directement sur le serveur.

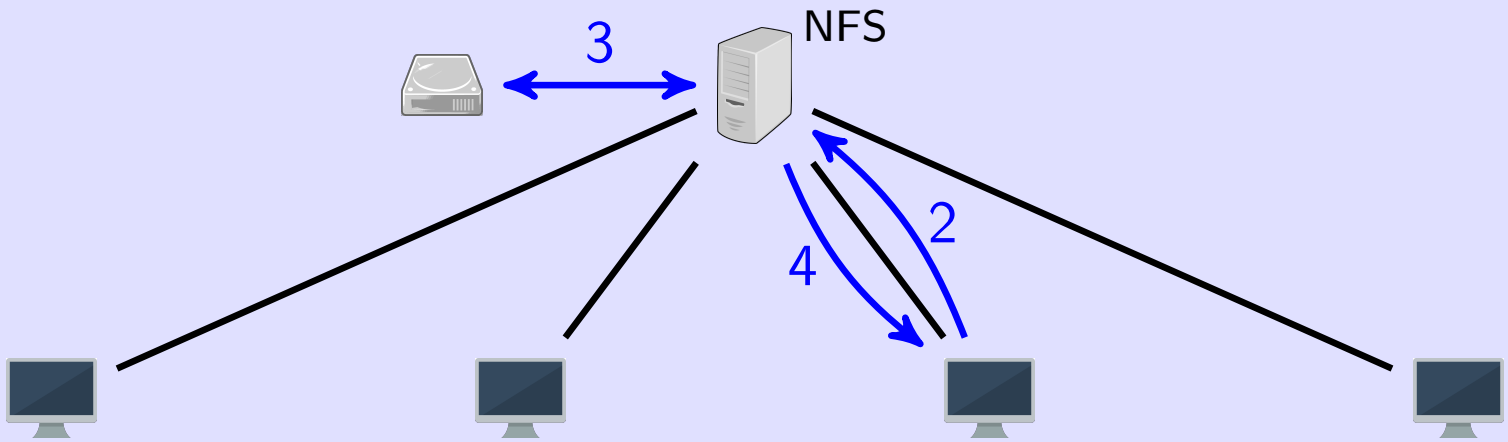
$\Rightarrow$ Des échanges sur le réseau sont nécessaires à chaque lecture/modification.

- Principe de transparence :
  - L'utilisateur ne doit pas se rendre compte que les données qu'il utilise sont stockées sur une autre machine.
  - Il utilise les données stockées sur le serveur comme toute autre donnée stockée en local.
- 2 protocoles basés sur ce principe :
  - **NFS** (Network File System)
    centralisation des fichiers
  - **NIS** (Network Information System)
    centralisation des données d'administration d'un réseau (identifiants et mots de passe des utilisateurs, noms des machines, ...)

- Sur la machine serveur, l'administrateur du réseau doit préciser les répertoires qui pourront être accédés à distance par les clients.
  - On dit que ces répertoires sont **exportés**.
- Sur la machine cliente, l'utilisateur doit, pour accéder à un répertoire exporté, associer un répertoire local au répertoire exporté.
  - On dit que le répertoire exporté a été **monté** sur le répertoire local qui est le **point de montage**.

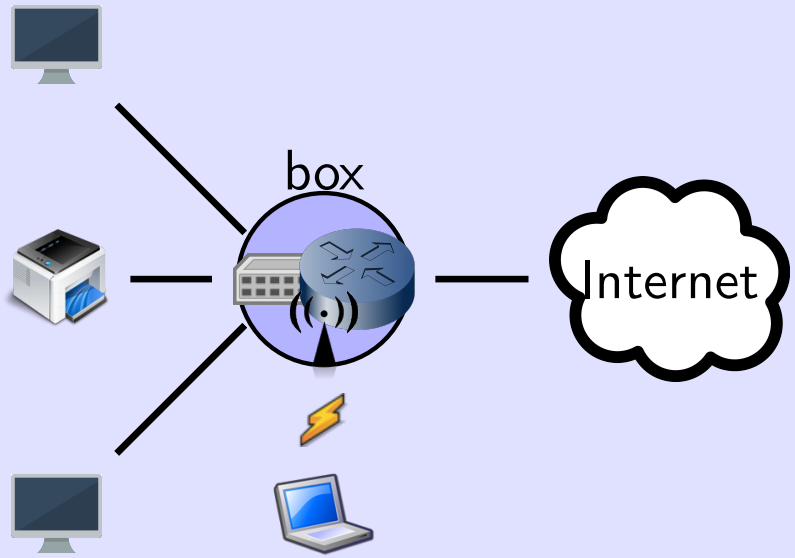
NFS — Exemple

- Le répertoire `/utilisateurs/max` est exporté par le serveur NFS.
- Le client a monté ce répertoire sur le point `/home/max`.



1. L'utilisateur exécute la commande `cat /home/max/fichier.txt`.
2. Le client envoie la requête "récupérer `/fichier.txt`" sur le répertoire exporté `/utilisateurs/max`.
3. Le serveur NFS consulte le fichier `/utilisateurs/max/fichier.txt`.
4. Le serveur NFS envoie le contenu de ce fichier au client.

Exemple 1 : Un réseau domestique



Quelques fonctions assurées par la box :

- switch (pour les appareils connectés en Ethernet)
- borne d'accès Wi-Fi
- routeur
- serveur DHCP pour attribuer des adresses IP aux appareils qui lui sont connectés (ordinateurs, portables, imprimantes, . . .)
- passerelle NAT

Principe

- En interne, les machines utilisent une **adresse privée**.
- Ils partagent tous la même **adresse publique**.
- L'adresse privée est utilisable uniquement sur le réseau interne.
- L'adresse publique est utilisable sur Internet.
- La passerelle NAT (la box dans notre exemple) se charge de modifier dans les paquets sortant et entrant les adresses.

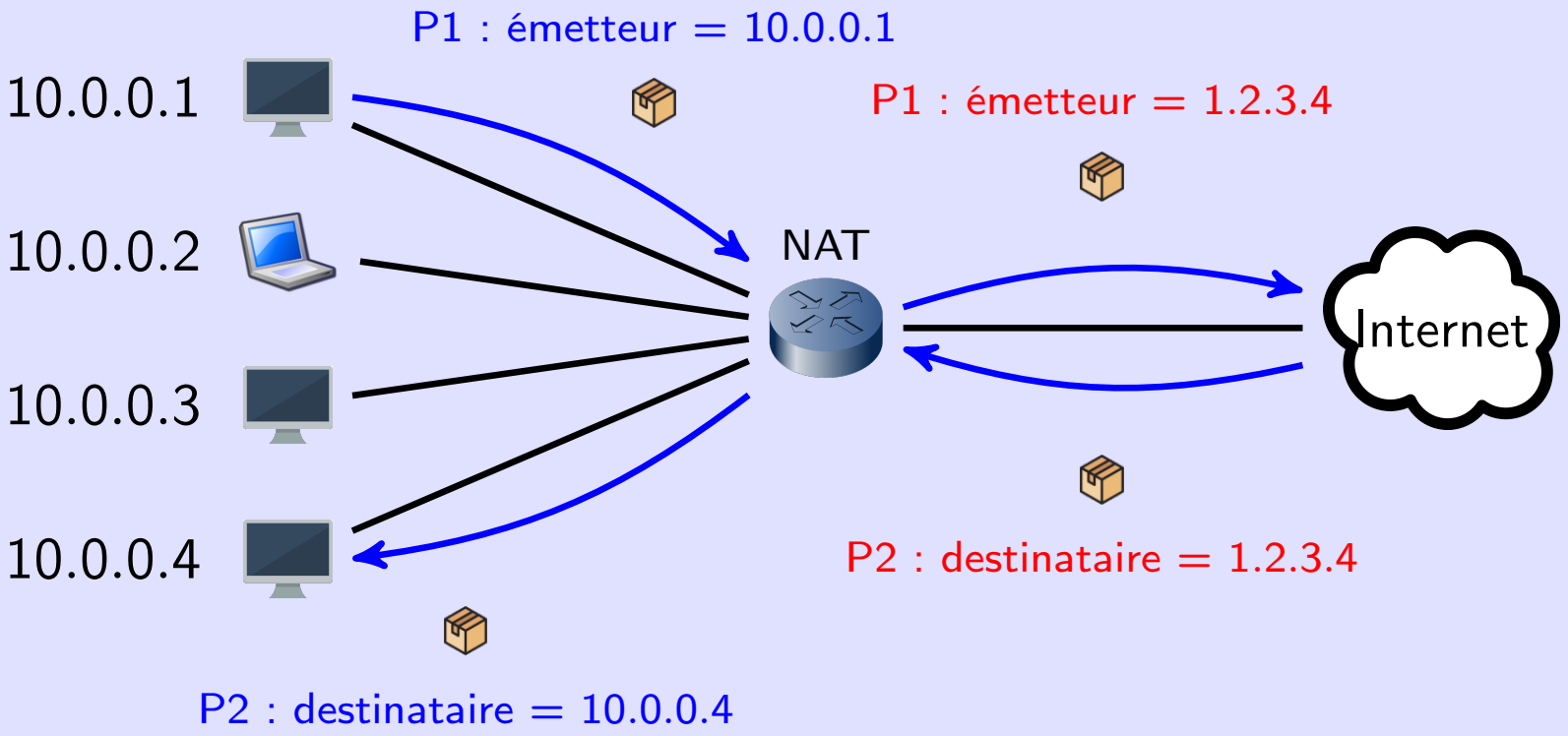
$\Rightarrow$ Vu de l'extérieur, toutes les machines du réseau interne ont la même adresse IP.

Avantage

On utilise **une seule adresse IP** (l'adresse publique) pour toutes les machines connectées à la box.

NAT — Exemple

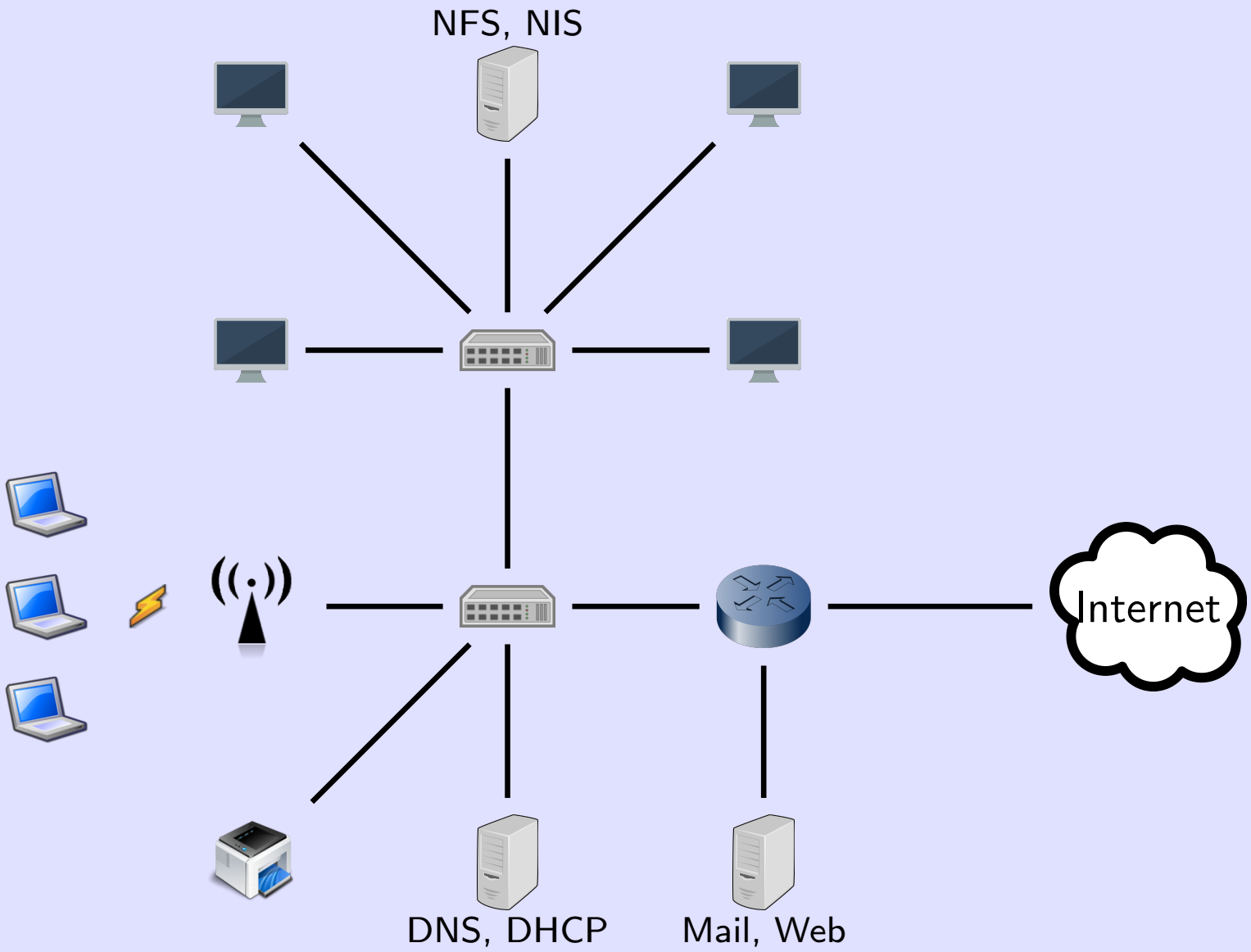
- Les machines du réseau privé ont des adresses de la forme 10.0.0.X.
- L'adresse publique du réseau est 1.2.3.4
- La machine 10.0.0.1 envoie un paquet P1 sur Internet.
- La machine 10.0.0.4 reçoit un paquet P2 depuis Internet.



- Comment la passerelle NAT sait-elle que P2 est destiné à 10.0.0.4 ?

$\Rightarrow$ détails dans le module M4210 (Infrastructures de sécurité)

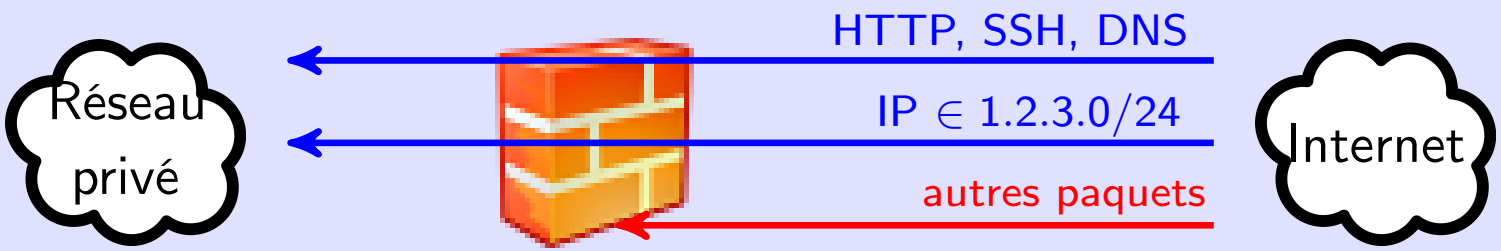
Exemple 2 : Un réseau d'entreprise



Fonctions assurées par le routeur : routage, pare-feu, NAT.

Le pare-feu (firewall)

- Son rôle :
  - Sécuriser le réseau (ou une machine)
    en laissant entrer certains types de paquets
    et en bloquant d'autres.
- Fonction généralement assurée par les routeurs.
- Comment ?
  - par l'ajout (par l'administrateur du réseau) de règles de filtrage sur le pare-feu
- Exemples de règles :
  - laisser entrer les paquets des protocoles HTTP, SSH ou DNS
  - laisser passer les paquets venant du réseau 1.2.3.0/24
  - bloquer tous les autres paquets



Plan

Son rôle : gérer un parc de machines (PC + matériel réseau)

- gestion des comptes des utilisateurs
- installation/mise à jour des logiciels
- sauvegarde des données
- support technique
- planification : mise en place de procédures automatiques
- mise en place de services pour les utilisateurs : NFS, DHCP, ...
- identification/correction des problèmes de sécurité
- identification/correction des problèmes de performance

...

- Unix est un système multi-utilisateurs.
- Un utilisateur a
  - un login/nom utilisé pour se connecter au système
  - un mot de passe
  - un identifiant (l'**UID**)
- Des groupes d'utilisateurs peuvent être créés par l'administrateur.
- Un groupe a :
  - un nom
  - un mot de passe (utilisé pour modifier les caractéristiques du groupe)
  - un identifiant (le **GID**)
- Un utilisateur a un groupe principal et peut appartenir à plusieurs groupes secondaires.
- Un utilisateur a tous les droits attribués au(x) groupe(s) auxquel(s) il appartient.
- Un fichier appartient à un utilisateur et à un groupe.

Afficher les droits et les propriétaires : `ls -l`

Exemple : afficher les droits sur le fichier `exec`

```
$ ls -l exec
-rwxr-x-x 1 max users 54678 24 sept. 09:08 exec
```

- `max` = utilisateur propriétaire du fichier `exec`
- `users` = groupe propriétaire du fichier exec
- `rwx` = droits de l'utilisateur propriétaire
  - lecture (r) + écriture (w) + exécution (x)
- `r-x` = droits des membres du groupe propriétaire
  - lecture (r) + exécution (x)
- `-x` = droits des autres utilisateurs
  - exécution (x)
- En représentation octale (base 8) les droits du fichier sont codés 751.
  - $7 = \underbrace{\text{r}}_{4} + \underbrace{\text{w}}_{2} + \underbrace{\text{x}}_{1}$
  - $5 = r + x$
  - $1 = x$

Le fichier des utilisateurs : /etc/passwd

- Il contient la liste des utilisateurs (une ligne / utilisateur).
- consultable par tous / modifiable uniquement par root
- Structure des lignes :

  Uname:x:UID:GID:commentaire:home:shell

  - Uname = login de l'utilisateur
  - x = mot de passe stocké dans /etc/shadow
  - UID = identifiant de l'utilisateur
  - GID = identifiant du groupe principal de l'utilisateur
  - commentaire = généralement, le nom complet de l'utilisateur
  - home = répertoire de connexion de l'utilisateur
  - shell = shell (interpréteur de commandes) de l'utilisateur

Exemple :

```
root:x:0:0:root:/root:/usr/bin/bash
max:x:1000:1000:Max le terrible:/home/max:/usr/bin/bash
```

L'utilisateur root

- Sur tous les systèmes Linux il y a toujours un utilisateur root.
- root = administrateur du système
- root a toujours le UID 0.
- Les règles de lecture/écriture ne s'appliquent pas à root :
  root a toujours les droits rw sur tous les fichiers/répertoires.

Le fichier des mots de passe : /etc/shadow

- Il contient les mots de passes cryptés des utilisateurs (une ligne / utilisateur) et leur validité dans le temps.
- consultable uniquement par `root`
- Structure des lignes :

`Uname:passwd:C3:C4:C5:C6:C7:C8:C9`

  - `Uname` = login de l'utilisateur
  - `passwd` = mot de passe. 2 possibilités :
    1. `*` ou `!` si le compte est désactivé (impossible de se connecter)
    2. chaîne sous la forme `$...` : mot de passe crypté
  - `C3`, ..., `C8` = sert à contrôler la validité du mot de passe dans le temps
  - `C9` = inutilisé, réservé à un usage futur

Exemple :

```
root:$6$16yG8oQIXDW3Dll0$XpCqlxm:15804:0:99999:7:::
sshd:!:15714::::::
max:$6$x6oGEIiv$28SrL2d4:15821:0:99999:7:::
```

Chiffrement des mots de passe

- Le système ne stocke jamais de mots de passe en clair.
- Quand un utilisateur modifie son mot de passe, le système :
  1. chiffre ce mot de passe en utilisant un algorithme de chiffrement (SHA, MD5, ...)
  2. stocke le mot de passe chiffré dans `/etc/shadow`
- Quand un utilisateur saisit son mot de passe pour se connecter, le système :
  1. chiffre ce mot de passe
  2. et compare le mot de passe chiffré à celui qui est dans `/etc/shadow`
- L'opération de chiffrement est irréversible : impossible de retrouver le mot de passe en clair à partir de la version chiffrée.

$\Rightarrow$ Même si un pirate a accès au fichier il ne peut pas retrouver le mot de passe.

Le fichier des groupes : /etc/group

- Il contient la liste des groupes (une ligne / groupe).
- consultable par tous, modifiable uniquement par root
- Structure des lignes :

Gname:x:GID:liste-Unames

  - Gname = nom du groupe
  - x = mot de passe stocké dans /etc/gshadow
  - GID = identifiant du groupe
  - liste-Unames = noms des utilisateurs du groupe (dont ce n'est pas le groupe principal) séparés par une virgule

Exemple :

```
root:x:0:
bin:x:1:root,toto
daemon:x:2:ivan
max:x:500:
```

root et toto sont dans le groupe bin. ivan est dans le groupe max

Commandes de gestion des groupes/utilisateurs

`groupadd`	créer un groupe
`groupmod`	modifier un groupe
`groupdel`	supprimer un groupe
`useradd`	créer un utilisateur
`usermod`	modifier un utilisateur
`userdel`	supprimer un utilisateur
`id`	afficher les informations sur un utilisateur
`passwd`	modifier le mot de passe d'un utilisateur
`chsh`	modifier le shell d'un utilisateur
`newgrp`	modifier le groupe principal d'un utilisateur
`su`	se connecter sous une autre identité
`pwck`	vérifier la cohérence des fichiers de configuration des utilisateurs (`/etc/passwd`, `etc/group`, ...)

La plupart de ces commandes nécessite d'être identifié en tant que `root`.

Utilisateurs système

- Un processus est associé à un utilisateur et a tous les droits de cet utilisateur.
- C'est aussi le cas des services.
- Les services sont exposés à des tentatives d'intrusion.
- En cas de faille de sécurité, un pirate peut par exemple prendre le contrôle du service.

$\Rightarrow$ Si le service est associé à root, le pirate peut faire ce qu'il veut sur la machine ...

- Pour éviter ce problème, on utilise un **utilisateur système**.
- Ce n'est pas un utilisateur réél (humain) :
  - impossible de se connecter sous cette identité
  - pas de répertoire personnel
- Cet utilisateur système sert uniquement à lancer le service.
- Il a des droits restreints sur le système.

Tâches périodiques avec cron

- Certaines tâches doivent être effectuées périodiquement mais ne nécessitent pas d'intervention humaine.

$\Rightarrow$ Un programme peut se charger du lancement de ces tâches.

- Avantages :
  1. Gain de temps.
  2. Pas de possibilité d'oubli/erreur humaine.
- Exemples typiques :
  - Supprimer les fichiers temporaires toutes les heures pour faire de l'espace sur le disque.
  - Sauvegarder des données tous les jours.

Principe de cron

- **cron** est un programme qui lance périodiquement des tâches.
- Le nom cron vient du dieu grec Chronos, dieu du temps.
- Ce programme est lancé par le service **crond**.
- Une fois une tâche planifiée, l'utilisateur n'a plus à intervenir.
  - Le service se charge de les lancer au bon moment.
- Les tâches à lancer sont décrites dans une table appelée **crontab**.
- Chaque utilisateur a sa propre table.

Fonctionnement du service crond

- Le système réveille le service crond toutes les minutes (la fréquence la plus élevée de lancement des tâches planifiées).
- Une fois réveillé, il
  1. consulte le répertoire `/var/spool/cron` qui contient, pour chaque utilisateur les dernières tâches qu'il a saisies
  2. mémorise ces tâches
  3. lance les tâches mémorisées et qui doivent être lancées à cet instant
  4. envoie un mail au propriétaire de la tâche concerné contenant le résultat
  5. puis il se rendort.

La table des tâches

- Chaque utilisateur a une table appelée crontab contenant des **règles** décrivant une tâche planifiée :
  1. la fréquence de lancement
  2. la commande lancée à la fréquence précisée
- Chaque règle contient 6 champs :

	Quand ?					Quoi ?
	minute	heure	jour du mois	mois	jour de la semaine	commande à exécuter
Valeurs possibles	0 à 59	0 à 23	1 à 31	1 à 12	1 à 7	

- Meta-caractères pour exprimer la fréquence :
  - ***** $\Rightarrow$ toutes les minutes (ou heures, ou jours, ...)
  - ***/N** $\Rightarrow$ tous les N minutes (ou heures, ou jours, ...)
  - **X-Y** $\Rightarrow$ X à Y
  - **X,Y** $\Rightarrow$ X et Y

Exemples d'enregistrements


```
#  supprimer les fichiers temporaires
#  toutes les 4 heures
0 */4 * * * rm -rf /tmp/*

#  afficher "Bonne année" tous les premiers de l'an
0 0 1 1 * echo "Bonne année"

#  afficher "Bon week-end" le vendredi à 17 h.
#  (seulement de Septembre à Juin)
0 17 * 1-6,9-12 5 echo "Bon week-end"
```

La commande crontab

- crontab permet de voir/modifier la table de l'utilisateur qui l'exécute.
- Syntaxe :

```
crontab [options] action
```

- actions possibles :
  - **-e** (édition) — ouvre l'éditeur de texte par défaut. L'utilisateur saisit ses règles qui sont ensuite sauvegardées dans `/var/spool/cron`.
  - **-l** (list) — affiche les règles de la table
  - **-r** (remove) — suppression de la table
- option courantes :
  - **-u util** — applique l'action pour l'utilisateur util.
    - Permet, par exemple, de modifier les règles d'un autre utilisateur.
    - Option utilisable par root uniquement.

Tâches à occurence unique avec at

- cron est utile pour définir des tâches exécutées périodiquement
- mais pas pour des tâches à occurence unique
  - (des tâches effectuées une seule fois)
- On utilise pour cela **at**.
- at permet d'exécuter
  - à un instant t
  - une tâche donnée sous la forme de commandes.

La commande at

Syntaxe :

```
at [options] [temps]
```

Déroulement :

1. at rend la main à l'utilisateur qui peut saisir les commandes à exécuter.
2. Une fois la saisie terminée l'utilisateur presse Ctrl+D (fin de fichier).
3. at attribue un numéro à la tâche créé.
4. Au temps spécifié, at exécute les commandes écrites par l'utilisateur.

Spécification du temps

- **absolu** — Exemples : "12 :00", "10 :00 2011-24-06"
- **relative** — Exemples : "now + 10 minutes", "midnight - 15 minutes"

$\Rightarrow$ syntaxe complexe, penser à consulter le manuel

Utilisation de at — Exemple


```
$ #  reboot de l'ordinateur dans 10 minutes
$ at "now + 10 minutes"
at> reboot<EOT>
job 18 at Wed Jun 22 15:13:00 2011

$ #  lancement du programme sauvegarde de mon
$ #  répertoire bin à minuit
$ at -f ~/bin/sauvegarde "midnight"
job 19 at Thu Jun 23 00:00:00 2011

$ #  annulation du reboot
$ at -d 18
```

Remarque. `EOT` est affiché lorsque l'utilisateur presse Ctrl+D.