UE1
Réseaux informatiques

Louis SWINNEN

SWILABUS

Références

- [1] James F. Kurose and Keith W. Ross, *Computer Networking : A Top-Down approach*, 6th edition, Pearson education, 2012

- [2] A. S. Tanenbaum and D. Wetherall, *Réseaux*, 5ème édition, Pearson, 2011

- [3] O. Bonaventure, *Computer networking : Principles, Protocols and Practice,* 2nd edition, UCL, https://www.computer-networking.info/

Informations pratiques (1)

- A propos de cette UE
  - Comprends une seule AA, répartie :
    - **30 h** de théorie
    - **18 h** de laboratoire
  - Organisée au 1er quadrimestre: de septembre à janvier
  - Examen de 1ère session en janvier
    - Rattrapage possible en Juin
    - Examen de 2ème session en septembre

Informations pratiques (2)

- Le laboratoire
  - Illustre les concepts vus au cours
  - La partie programmation est réalisé en Python
- Répartition en 1ère session
  - Note Année = **30 %**
    - Laboratoires
    - Interrogations durant l'année
  - Note Examen = **70 %**
    - Examen de 1ère session en janvier

Informations pratiques (5)

- Les professeurs impliqués
  - Cours théorique : SWILA
  - Laboratoires : MANCH, XXXX
- Les diapositives
  - Sont disponibles sur la plateforme e-learning
- Prérequis
  - Aucun
- Questions ?
  - Durant le cours
  - Via la plateforme e-learning

Informations pratiques (6)

- Travailler chez soi
  - Les concepts théoriques doivent être revus très régulièrement
  - Passez un peu de temps sur les exemples et exercices proposés
    - Pour bien comprendre les notions sous-jacentes
    - Pour vous amuser !
  - Tous les outils utilisés existent sous Windows, Linux et/ou Mac OS X

INTRODUCTION

Introduction (1)

- Le cours
  - Présente les mécanismes de base en réseau
    - Les modèles
    - Les différentes couches
    - Le pourquoi de cette découpe
  - Donne une vue d'ensemble du fonctionnement d'un réseau
    - Au travers de nombreux exemples

Introduction (2)

- Fil conducteur
  - Partir d'éléments que vous connaissez
    - Les applications qui utilisent le réseau (web, …)
  - Pour arriver à détailler le fonctionnement d'un réseau de manière précise
    - Explication des mécanismes mis en place et de leur pertinence
      - » Pourquoi cette solution ?
      - » Comment a-t-elle été élaborée ?

Introduction (3)

- L'intérêt du cours
  - Pourquoi étudier les réseaux ?
    - Les réseaux sont partout aujourd'hui
    - Porte d'entrée d'un système informatique
    - Indispensable pour tout spécialiste en sécurité des systèmes
    - Parce que c'est passionnant !
  - Liens avec d'autres matières
    - Programmation (Python)
    - Systèmes d'exploitation

Introduction (4)

- Objectifs
  - Ce cours a pour but de :
    - Vous permettre d'appréhender et de **comprendre les mécanismes** mis en place pour envoyer et recevoir de l'information
      - *Comment l'information est-elle transmise ?*
    - Vous permettre de **programmer** une application légère en Python pour interagir avec le réseau

Notions d'architecture (1)

- Dans un système informatique
  - Les informations sont enregistrées pour pouvoir être traitées par un ordinateur
  - L'ordinateur ne peut mémoriser l'information que sous la forme d'états
    - La donnée est présente – absente
    - La tension électrique est présente – absente
    - Il y a 2 états: 0 ou 1 ➔ c'est un bit (binary digit)
    - On peut regrouper plusieurs états ensembles: 8 états = 8 bits = 1 octet (ou 1 byte)

Notions d'architecture (2)

- Ex: 1100 1101 est un octet (= 8 bits)

Compter en binaire (base-2):

0000 0000 = 0	0000 1000 = 8
0000 0001 = 1	0000 1001 = 9
0000 0010 = 2	0000 1010 = 10
0000 0011 = 3	0000 1011 = 11
0000 0100 = 4	0000 1100 = 12
0000 0101 = 5	0000 1101 = 13
0000 0110 = 6	0000 1110 = 14
0000 0111 = 7	0000 1111 = 15

Notions d'architecture (3)

- Ex: 1100 1101 est un octet (= 8 bits)

Rappel (en base 10 [donc 10 chiffres utilisés]):

$205 = 2 \times 10^2 + 0 \times 10^1 + 5 \times 10^0$ → Position du chiffre dans le nombre

En base 2 [donc 2 chiffres utilisés]) :

$1100\ 1101 = 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 205$

SWILABUS

Notions d'architecture (4)

- Ex: 1100 1101 est un octet (= 8 bits)

Compter en hexadécimal (base-16):

0_h = 0000 0000 = 0_d	8_h = 0000 1000 = 8_d
1_h = 0000 0001 = 1_d	9_h = 0000 1001 = 9_d
2_h = 0000 0010 = 2_d	A_h = 0000 1010 = 10_d
3_h = 0000 0011 = 3_d	B_h = 0000 1011 = 11_d
4_h = 0000 0100 = 4_d	C_h = 0000 1100 = 12_d
5_h = 0000 0101 = 5_d	D_h = 0000 1101 = 13_d
6_h = 0000 0110 = 6_d	E_h = 0000 1110 = 14_d
7_h = 0000 0111 = 7_d	F_h = 0000 1111 = 15_d

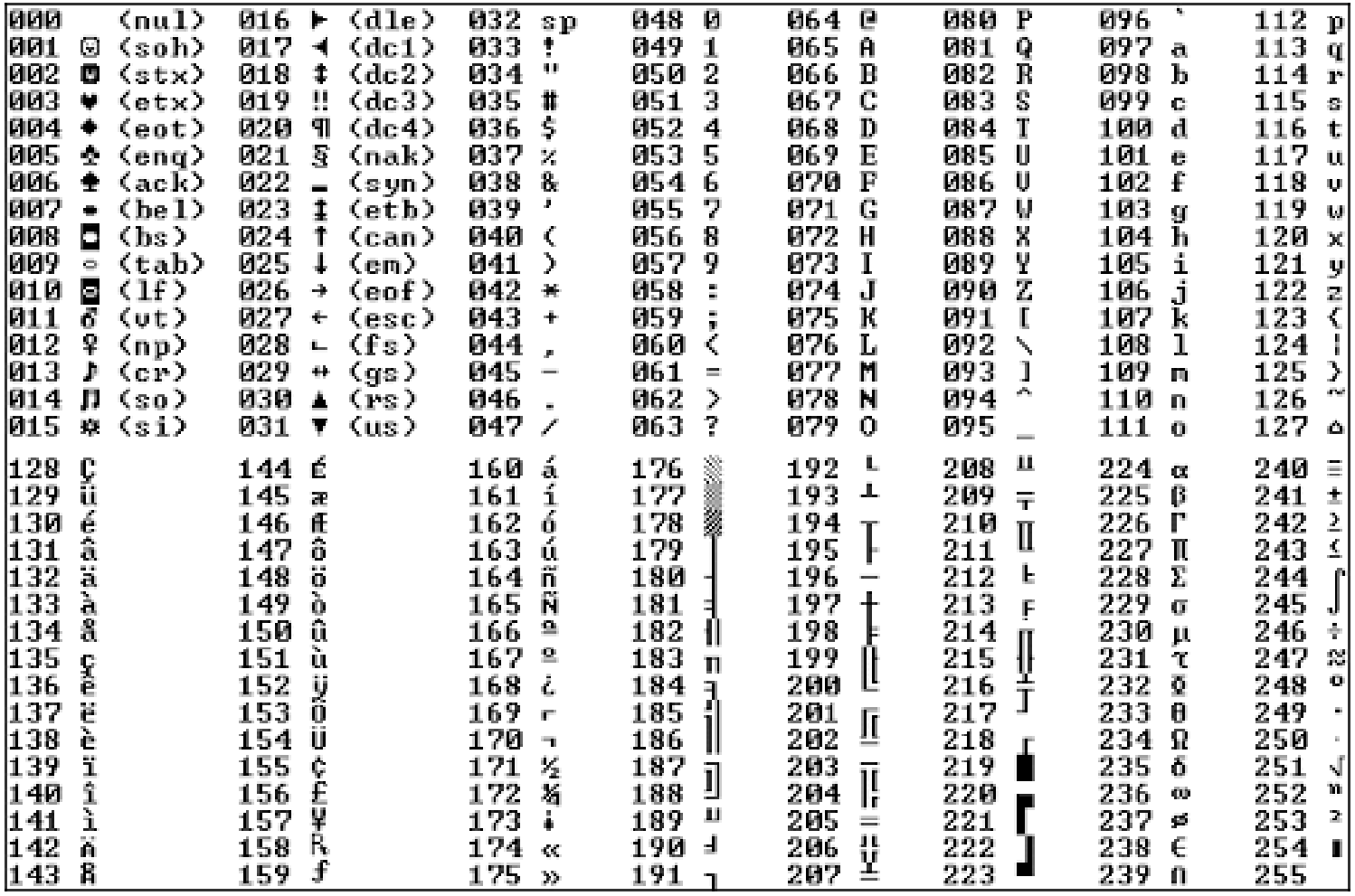
Utile car un chiffre hexadécimal représente 4 bits

1100 1101 = CD => $Cx16^1 + Dx16^0 = 12x16 + 13x1 = 192+13 = 205$

Notions d'architecture (5)

- Combien d'états différents avec 8 bits ?
  - Il est possible de mémoriser 2^8 états différents (valant chacun soit 0, soit 1) avec un octet
  - Ainsi:
    - La plus petite valeur est: 0000 0000 = 0_d
    - La plus grande valeur est: 1111 1111 = 255_d = FF_h
  - Avec un octet (= 8 bits), je peux représenter une information dont la valeur est comprise entre 0 et 2^8-1
  - Comment mémoriser un information comme un texte ?
    - La table ASCII

Notions d'architecture (6)



000	(nul)	016	(dle)	032	sp	048	0	064	@	080	P	096	`	112	p
001	(soh)	017	(dc1)	033	!	049	1	065	A	081	Q	097	a	113	q
002	(stx)	018	(dc2)	034	"	050	2	066	B	082	R	098	b	114	r
003	(etx)	019	(dc3)	035	#	051	3	067	C	083	S	099	c	115	s
004	(eot)	020	(dc4)	036	$	052	4	068	D	084	T	100	d	116	t
005	(enq)	021	(nak)	037	%	053	5	069	E	085	U	101	e	117	u
006	(ack)	022	(syn)	038	&	054	6	070	F	086	V	102	f	118	v
007	(bel)	023	(etb)	039	'	055	7	071	G	087	W	103	g	119	w
008	(bs)	024	(can)	040	(	056	8	072	H	088	X	104	h	120	x
009	(tab)	025	(em)	041	)	057	9	073	I	089	Y	105	i	121	y
010	(lf)	026	(eof)	042	*	058	:	074	J	090	Z	106	j	122	z
011	(vt)	027	(esc)	043	+	059	;	075	K	091	[	107	k	123	{
012	(np)	028	(fs)	044	,	060	<	076	L	092	\	108	l	124	|
013	(cr)	029	(gs)	045	-	061	=	077	M	093	]	109	m	125	}
014	(so)	030	(rs)	046	.	062	>	078	N	094	^	110	n	126	~
015	(si)	031	(us)	047	/	063	?	079	O	095	_	111	o	127	⌂
128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▒	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	∈	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: http://www.thealmightyguru.com/Pointless/ASCII.html

SWILABUS

Notions d'architecture (7)

– Quelques puissances de 2^n courantes:

Puissances de 2^n :

$2^0 = 1$	$2^8 = 256$	$2^{16} = 65\,536$
$2^1 = 2$	$2^9 = 512$	
$2^2 = 4$	$2^{10} = 1\,024$	$2^{24} = 16\,777\,216$
$2^3 = 8$	$2^{11} = 2\,048$	
$2^4 = 16$	$2^{12} = 4\,096$	$2^{32} = 4\,294\,967\,296$
$2^5 = 32$	$2^{13} = 8\,192$	
$2^6 = 64$	$2^{14} = 16\,384$	
$2^7 = 128$	$2^{15} = 32\,768$	

Grâce à ces valeurs, on peut déterminer le nombre de bits nécessaires pour sauvegarder un nombre donné. Ainsi, quand on parle d'un entier *codé sur 32 bits*, il peut mémoriser une valeur décimale entre 0 et 4 294 967 295 (2^{32} -1)

SWILABUS

Notions d'architecture (8)

– Conversion décimal → binaire

- Ex: **205**, la valeur est comprise entre 2^7 - 1 et 2^8 -1) il faut donc **<u>au moins 8 bits</u>** pour enregistrer cette valeur.

Valeur: 205 >= 2^7, donc le $8^{ème}$ bit vaut 1, reste=205-128 = 77
Valeur: 77 >= 2^6, donc le $7^{ème}$ bit vaut 1, reste=77-64 = 13
Valeur: 13 < 2^5, donc le $6^{ème}$ bit vaut 0
Valeur: 13 < 2^4, donc le $5^{ème}$ bit vaut 0
Valeur: 13 >= 2^3, donc le $4^{ème}$ bit vaut 1, reste 13-8 = 5
Valeur: 5 >= 2^2, donc le $3^{ème}$ bit vaut 1, reste 5-4 = 1
Valeur: 1 < 2^1, donc le $2^{ème}$ bit vaut 0
Valeur: 1 = 2^0, donc le 1^{ier} bit vaut 1

On obtient ainsi: 1100 1101

SWILABUS

Notions d'architecture (9)

- Quelques exercices de conversion

Nombre à convertir	Nombre de bits nécessaires	Valeur en binaire	Valeur en hexadécimal
16	5	0001 0000	10
159	8	1001 1111	9F
254	8	1111 1110	FE
630	10	0010 0111 0110	276
2301	12	1000 1111 1101	8FD
4789	13	0001 0010 1011 0101	12B5
65200	18	1111 1110 1011 0000	FEB0

Notions d'architecture (10)

- Quelques exercices de conversion

Nombre à convertir	Nombre de bits	Valeur décimale	Valeur en hexa/binaire
FF_h	8	255	1111 1111
99_h	8	153	1001 1001
$ABCD_h$	16	43981	1010 1011 1100 1101
$1001\ 1100_2$	8	156	9C
$1101\ 0111_2$	8	215	D7
$1010\ 1000\ 0111_2$	12	2695	A87
$1001\ 0000\ 1001\ 0011\ 0110\ 0101\ 1001\ 1100_2$	32	2425578908	9093 659C

Notions d'architecture (11)

- Parmi les opérations courantes sur les nombres binaires, il y a les opérations logiques AND, OR, NOT, et XOR
  - Ce sont des instructions de base de l'unité centrale (le processeur ou CPU)
  - Permet de construire des opérations plus complexes comme les opérations arithmétiques classiques (+, -, *, /)
    - Voir cours d'architecture

Notions d'architecture (12)

– En réseau, on utilise principalement les opérateurs logiques, raison pour laquelle ces opérations sont détaillées ici:

AND	0	1
0	0	0
1	0	1

1101 AND 0011

```
 1101
 0011
------
 0001
```

OR	0	1
0	0	1
1	1	1

1101 OR 0011

```
 1101
 0011
------
 1111
```

XOR	0	1
0	0	1
1	1	0

1101 XOR 0011

```
 1101
 0011
------
 1110
```

NOT	
0	1
1	0

NOT(1101)

0010

Notions d'architecture (13)

– Exercices avec opérateurs logiques:

	1100 0000	1010 1000	1000 0101	1110 0011
AND	1111 1111	1111 1111	1111 0000	0000 0000

	1100 0000	1010 1000	1000 0101	1110 0011
XOR	1111 1111	1111 1111	1111 0000	0000 0000

	1100 0000	1010 1000	1000 0101	1110 0011
OR	1111 1111	1111 1111	1111 0000	0000 0000

SURVOL INTRODUCTIF

Introduction

Plan :

Qu'évoque, pour vous, le terme *réseau* ?

SWILABUS

Les standards (1)

Plan :

- La nécessité de communiquer a fait émerger **des standards**.
  - Indispensable pour construire des applications interopérables
    - Ex: Le mail, l'accès à des dossiers partagés, …
  - Permet de s'appuyer sur l'existant
    - Nombreux développements réalisés
  - Ils sont décrits *très précisément* dans des documents présentant tous les aspects techniques

Les standards (2)

- Organismes de standardisation:
  - ANSI (American National Standard Institute)
    - Organisme de standardisation américain
      - » Ex: ANSI C
  - IEEE (Institute of Electrical and Electronics Engineers)
    - Organisme de standardisation des ingénieurs
      - » Ex: IEEE 802.11n (=Wifi)
  - ISO (International Standard Organisation)
    - Organisme de standardisation international
    - Standardisation de nombreuses choses (vis, clou, langages, moyens de communication, …)

Plan :

SWILABUS

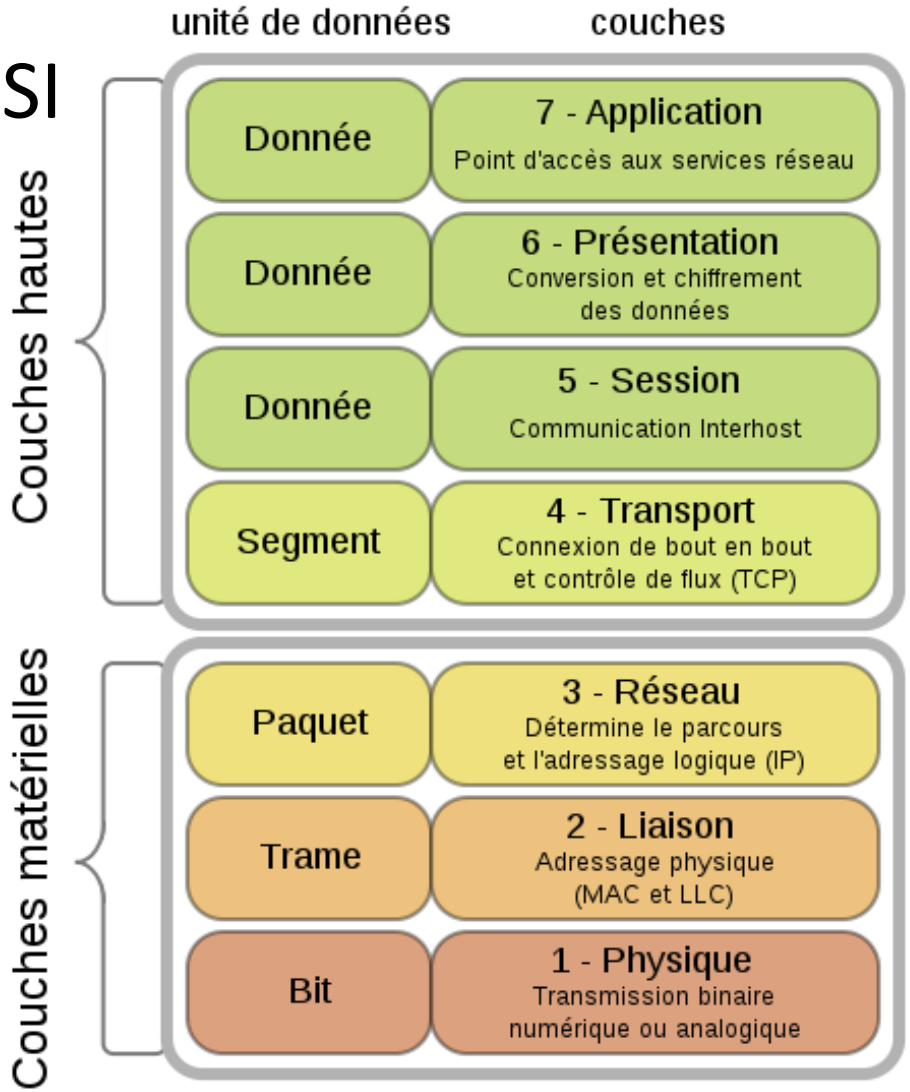
Les standards (3)

Plan :

- IETF (Internet Engineering Task Force)
  - Organisme de standardisation très important en ce qui concerne les réseaux
  - Standardise les applications courantes :
    - Ex: mail (SMTP, POP3, IMAP), web (HTTP), …
  - Standardise les protocoles réseaux
    - Ex: IP, UDP, TCP, …
  - Les documents décrivant ces standards sont appelés des RFC (**R**equest **F**or **C**omment) et peuvent être obtenus sur le site web : http://www.ietf.org/rfc

Les modèles (1)

- Modèle OSI



Source: Wikipedia, modèle OSI

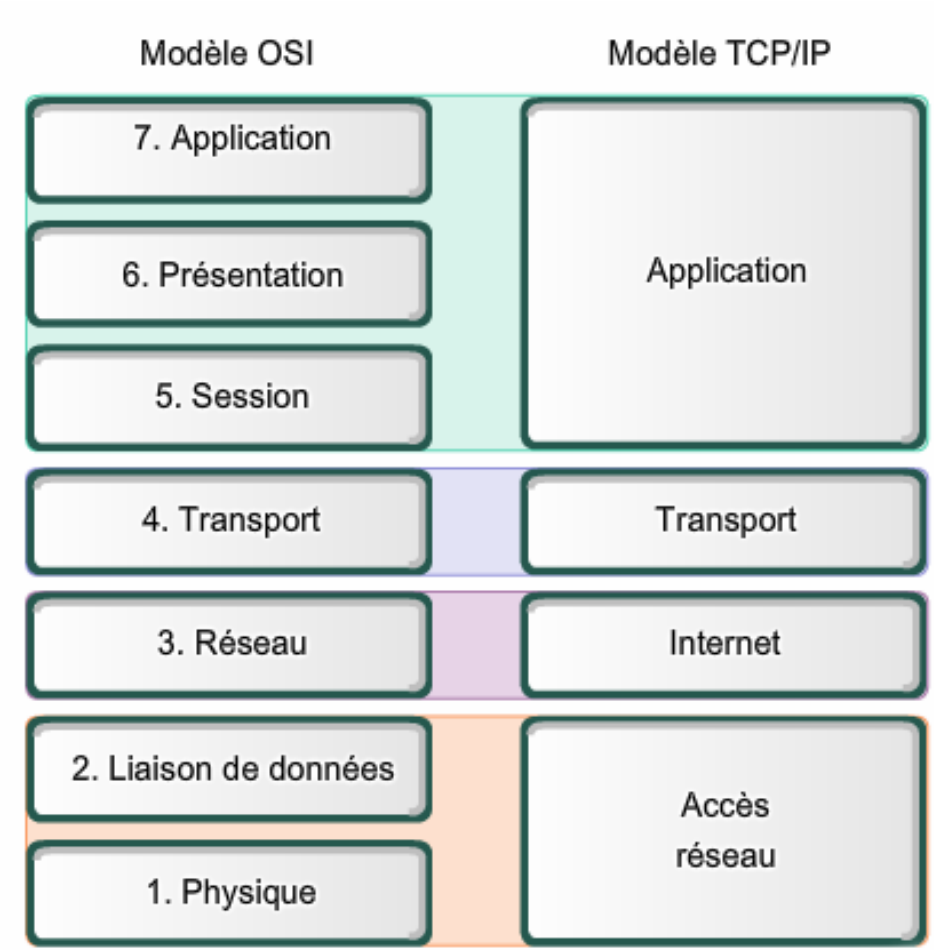
Plan :

SWILABUS

Les modèles (2)

- Modèle de référence TCP/IP

Plan :



Source: Wikipedia, modèle TCP/IP

SWILABUS

Plan du cours (1)

Plan :

- Le cours suivra la découpe en couche:
  - **Chapitre 2: La couche application**
    - Etude des applications courantes en réseau
    - Présentation des éléments de programmation
  - **Chapitre 3: La couche transport**
    - Besoins des applications
    - Support proposé par la couche réseau
    - Construction d'un protocole de transport fiable

Plan du cours (2)

Plan :

- **Chapitre 4: La couche internet / réseau**
  - Besoins
  - Routage des informations
  - Fonctionnement de IPv4 et IPv6
- **Chapitre 5: Accès réseau / couche liaison de données**
  - Services proposés
  - Délimitation de l'information
  - Accès au média
  - Topologie réseau
  - PPP

SWILABUS

CHAPITRE 2

LA COUCHE APPLICATION

Introduction (1)

- Emetteur
  - Application / programme souhaitant envoyer des informations via le réseau vers un destinataire
  - Il peut transmettre *une requête* qui sera analysée par le destinataire *qui y répondra*
    - Dans ce cas, on parle de **mode client/serveur**
    - Le **client** soumet une demande à un **serveur** et ce dernier *peut* y répondre.

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

SWILABUS

Introduction (2)

- Destinataire
  - Application ou programme qui recevra des informations par le réseau informatique
  - Le destinataire (ou serveur) peut *répondre* à une requête en provenance de l'émetteur (ou client).
- Exemple d'échange client / serveur:
  - La consultation d'une page web :
    - Client: Google Chrome, Mozilla Firefox
    - Serveur: serveur web Apache, nGinX, Tomcat ou IIS

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

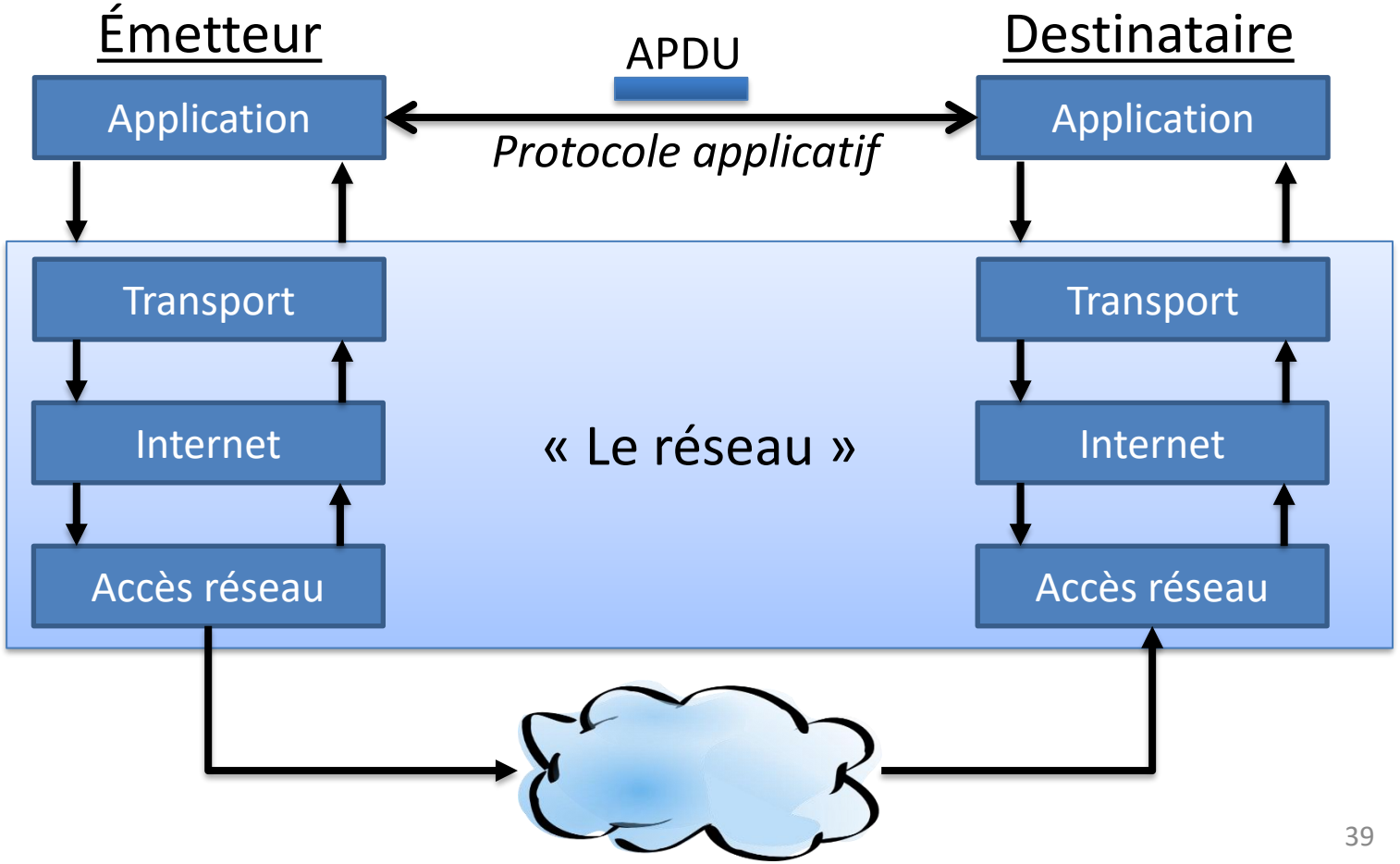
SWILABUS

Introduction (3)

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

- Modélisation



SWILABUS

Introduction (4)

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

– Interprétation

- L'émetteur utilise **sa couche transport** pour communiquer avec le réseau
  - Il lui envoie les informations et cette dernière est chargée de les faire parvenir à destination *en utilisant les couches inférieures.*
- Le destinataire reçoit les informations **au travers de sa couche transport.**
  - Cette couche est responsable de transmettre les informations émises au destinataire de la façon souhaitée (en fournissant le service voulu).
  - Deux types de couche transport :
    - » Envoi fiable d'information : le réseau assure que l'information arrive à destination

Introduction (5)

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

  - » Envoi *simple* d'information : aucun mécanisme assurant la fiabilité n'est de mise
- Deux protocoles de transports standards:
  - » TCP (Transmission Control Protocol)
    - Permet l'envoi d'information de manière fiable, avec une connexion, l'envoi des informations et la fermeture de la connexion
    - Utilisé dans la plupart des applications réseaux
  - » UDP (User Datagram Protocol)
    - Permet l'envoi *simple* et non fiable d'information
    - Ce protocole utilisé lorsque *des retransmissions* sont jugées trop coûteuses

Introduction (6)

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

- Le protocole applicatif est le langage utilisé par l'application pour communiquer. Il s'agit:
  - D'une définition **syntaxique** (forme des messages) des mots du langage.
  - D'une définition **sémantique** (sens de l'échange) du dialogue
- Ce protocole applicatif est partagé entre l'émetteur et le récepteur qui doivent parler le même langage **pour se comprendre**.
  - Ex: le navigateur internet parle le langage HTTP version 1.1 ou 2.0, compris par le serveur web et standardisé

Introduction (7)

- Besoin d'identification
  - Pour que deux applications puissent dialoguer, il faut *identifier* chacune d'entre-elle.
    - On associe à chaque application un numéro, appelé *port* permettant de l'identifier
    - Dans le réseau internet (utilisant IP) :
      - L'adresse IP (couche internet) permet d'identifier **la machine** émettrice ou destinataire de l'information.
      - Le numéro de port (couche transport) permet d'identifier **l'application** émettrice ou destinataire de l'information.

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

SWILABUS

Introduction (8)

Plan :

- **Intro.**
- DNS
- Mail
- Web
- Eléments de progr.

- Exemple:
  - **Adresses IP**: 193.190.64.124 (helmo.be) ou 2001:bc8:38eb:fe10::11 (swila.be)
  - Quelques **numéros de ports** courants:
    - 21 – FTP
    - 22 – Connexion terminale chiffrée (SSH)
    - 23 – Connexion terminale en clair (telnet)
    - 25 – Envoi de mail
    - 53 – résolution de noms
    - 80 – HTTP (web)
    - 443 – HTTPS (web sécurisé)

Le DNS (1)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

- Le problème
  - Chaque machine est identifiée au moyen d'un numéro unique: **l'adresse IP**. Elle est codée sur 32 bits (IPv4) ou 128 bits (IPv6)
  - Comment les retenir facilement et rendre les sites accessibles ?
    - **En associant un nom (plus parlant) à une (ou plusieurs) adresse(s) IP**

193.190.64.124
www.helmo.be

[2001:bc8:38eb:fe10::11]
195.154.39.227
www.swila.be

SWILABUS

Le DNS (2)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

– Qui parmi vous possède déjà un nom de domaine ?

– Comment peut-on « réserver » un nom de domaine ?

SWILABUS

Le DNS (3)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

– 1ère solution

- Utiliser un fichier texte qui permet la conversion entre un ensemble de noms et les adresses IP correspondantes
- Solution implémentée dans tous les systèmes d'exploitation
- Exemple

```
195.154.29.227 swila swila.be
    www.swila.be www.facebook.com
```

- Emplacement du fichier
  - Linux: `/etc/hosts`
  - Windows: `C:\Windows\System32\Drivers\Etc\Hosts`

Le DNS (3)

- 2ème solution
  - Utiliser une base de données
    - Centralisée ?
      - » Performance ?
      - » Fiabilité ?
    - Distribuée
      - » Organisation de l'information
    - Comment garantir l'unicité des noms ?
      - » Choix par l'utilisateur ? Unicité ?
      - » Construit aléatoirement ?
        - Plus facile à retenir qu'une adresse IP ?

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

SWILABUS

Le DNS (5)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

– Unicité des noms
  - Le DNS utilise un système hiérarchique
  - Ce système est déjà très éprouvé
    – Utilisé par les services postaux
  - Les noms possibles sont répartis en catégories
    – Géographique, le suffixe identifie un pays
      » Ex: .be, .fr, .uk, .nl, .lu, .de, .jp, .eu, …
    – Générique:
      » Ex: .com, .org, .net, .guru, .paris, .info, .xxx, …
  - On voit apparaître une **hiérarchie**
    – Les machines HELMo: *.helmo.be
    – Les machines du campus: *.cg.helmo.be

Le DNS (6)

- Comment l'unicité des noms est-elle assurée ?
  - Chaque domaine est responsable des noms situés sous-lui
    - » Ex: HELMo est responsable des noms situés sous-lui (qui se termine par .helmo.be) et donc: cg.helmo.be, salto.helmo.be, …
  - Unicité est garantie par le gestionnaire du domaine qui gère **ses** noms (ex: *.helmo.be)
  - Système intelligent
    - » Un domaine ne gère que ses sous-domaine
    - » Niveau de sécurité satisfaisant
    - » Performance adéquate

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

Le DNS (7)

- Les serveurs DNS ou serveur de noms
  - Application serveur, UDP, port 53
  - Application qui gère un (sous-) domaine
  - Permet la conversion d'un nom de domaine en adresse IP et inversement
    - Ex: DNS de helmo: *ns* dont l'IP est 193.190.64.113
  - Des serveurs secondaires peuvent être configurés

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

Le DNS (8)

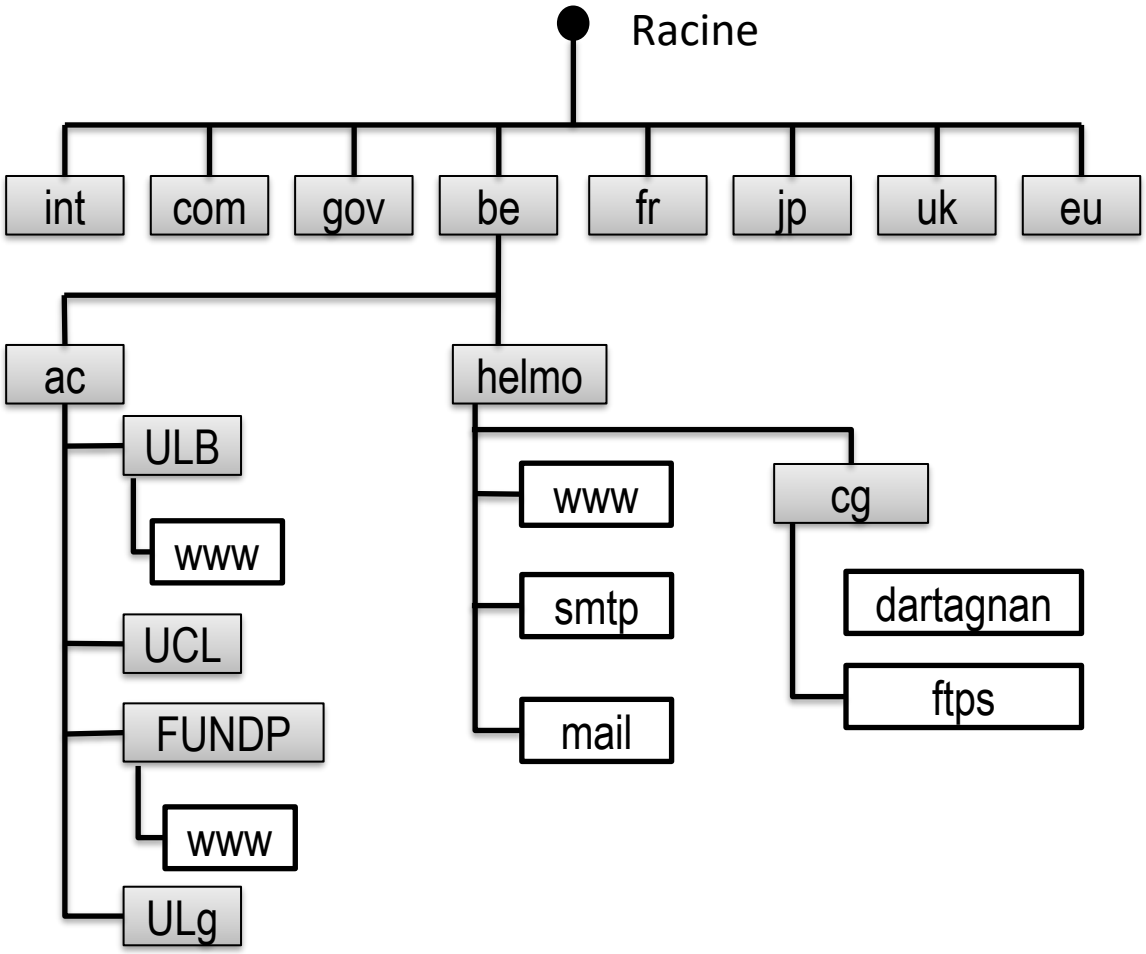
Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

– Il y a **des serveurs DNS mondiaux** ou **racines** :
  - Qui sont connus de tous les systèmes
  - Qui permettent de donner les adresses des serveurs DNS de 1^{ier} niveau:
    – Serveurs gérant les suffixes: « .be », « .fr », « .com », « .org », « .net », « .guru », …
  - Ils sont listés par une lettre (de A à M):
    – A.ROOT-SERVERS.NET, … M.ROOT-SERVERS.NET
  - Ils sont répartis géographiquement

Le DNS (9)

– Exemple sur HELMo



Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

SWILABUS

Le DNS (10)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

– Exemple de résolution de nom

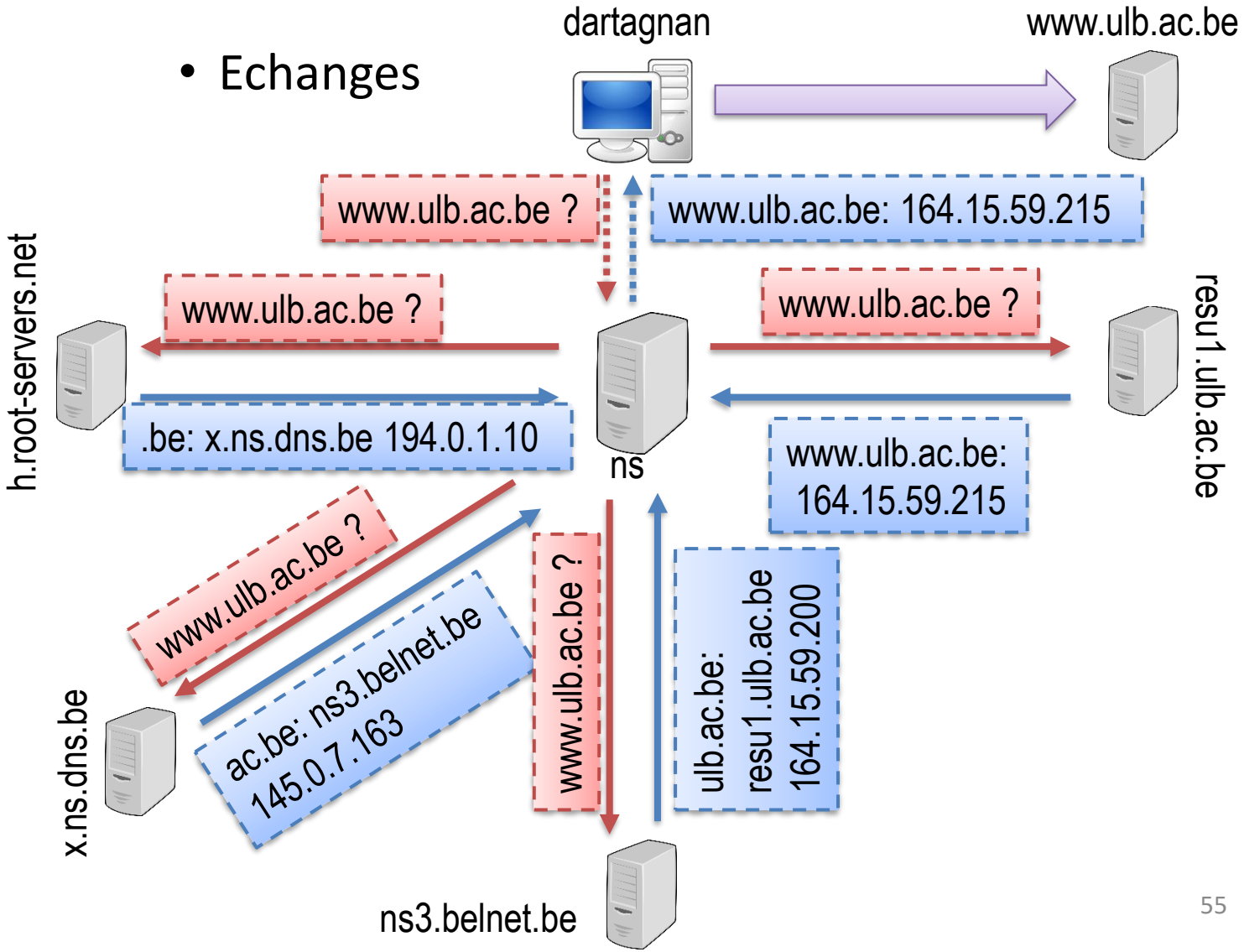
- La machine *dartagnan.cg.helmo.be* souhaite contacter www.ulb.ac.be
  - Il faut connaître l'adresse IP de cette machine pour pouvoir lui envoyer de l'information.
  - *Dartagnan* regarde dans son fichier *hosts* pour voir si l'information s'y trouve
  - Si elle ne s'y trouve pas, il contacte le serveur de nom pour son domaine (il contacte **son** serveur DNS) dans notre cas *ns*
  - *ns* vérifie si l'information se trouve dans sa cache
    - » Si oui, l'information est retournée à la machine
    - » Sinon, *ns* parcourra l'arbre **de manière adaptée** pour fournir l'adresse à *dartagnan*.

Le DNS (11)

Plan :

- Intro.
- **DNS**
- Mail
- Web
- Eléments de progr.

- Echanges



SWILABUS

Le mail (1)

- Deux services différents:
  - L'envoi du courrier électronique
    - Via le protocole applicatif SMTP (Simple Mail Transfer Protocol)
      - Utilisation de TCP, port 25
  - La réception du courrier électronique
    - Via le protocole applicatif POP3 (Post Office Protocol v3) ou IMAP (Internet Message Access Protocol).
      - Utilisation de TCP, port 110 (POP3) ou 143 (IMAP)

Plan :

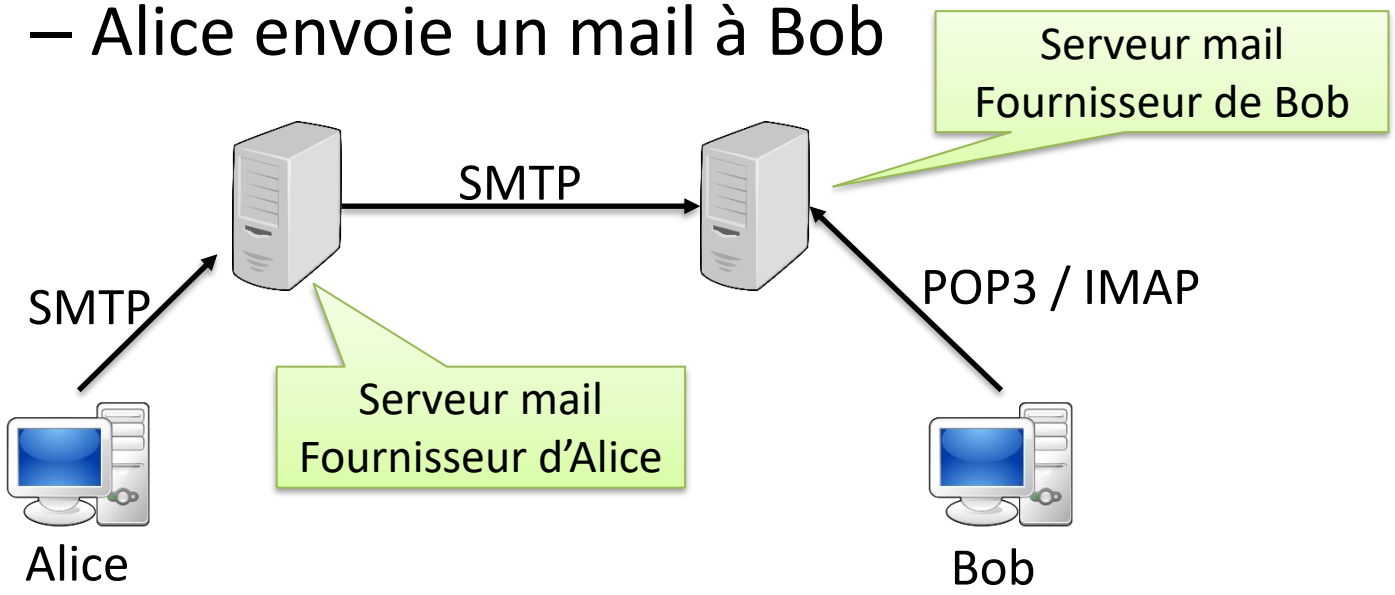
- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

Le mail (2)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

- Chemin suivi par le courrier
  - Alice envoie un mail à Bob



- Le courrier arrive sur le serveur mail destination (celui du fournisseur de Bob) et est stocké dans sa boîte. Bob doit venir consulter sa boîte pour recevoir les nouveaux courriers

SWILABUS

Le mail (3)

- Le contenu d'un courrier électronique
  - Entête:
    - Adresse source et destination (sous la forme `utilisateur@domain`)
    - Sujet du message (**Ne pas l'oublier !**)
    - Texte du courrier
  - Le message:
    - Doit respecter la Netiquette (RFC 1855)
    - Un mail doit toujours être considéré comme public (il peut facilement être propagé à d'autres destinataires – **pensez-y!**).

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

SWILABUS

Le mail (4)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

- Extension du courrier électronique
  - Initialement, uniquement des messages textes (codés en ASCII 7 bits)
  - Désormais, tous les serveurs mails prennent en charge l'ASCII 8 bits
  - Comment transmettre une pièce jointe binaire (photo, vidéo, …) ?
    - En transformant de fichier binaire en texte
      - Principe de l'encodage **base 64**
      - Encode le fichier en texte contenant uniquement les caractères A-Z, 0-9, +, /, =
      - Lisible par tous les serveurs mails.

SWILABUS

Le mail (5)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

- Envoi du message
  - Application serveur, TCP, port 25
  - Protocole applicatif SMTP (extrait):
    - HELO <domaine utilisateur> — Début du dialogue
    - MAIL FROM: <user@domain>
    - RCPT TO: <user@domain> — Emetteur et destinataire du mail
    - DATA — Transfert du message complet
    - QUIT — Fermeture du dialogue
  - ! Il s'agit des messages SMTP sans extension. Le protocole a évolué.

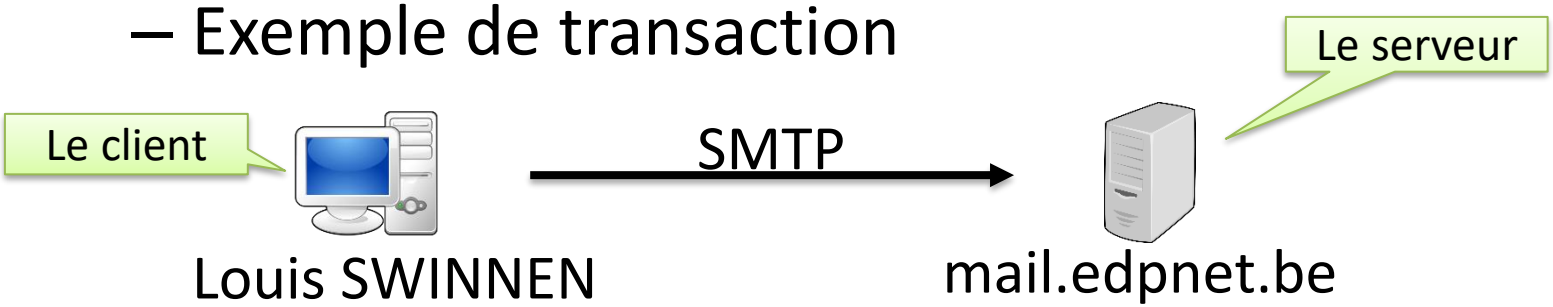
SWILABUS

Le mail (6)

– Exemple de transaction

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.




```
$ telnet mail.edpnet.be 25

220 EDPnet edpnet.org Mail Server ready and serving,
  ESMTP Postfix (LXmail03).
HELO test@localhost
250 LXmail03.edpnet.net
MAIL FROM: <admin@gmail.com>
250 2.1.0 Ok
RCPT TO: <l.swinnen@helmo.be>
250 2.1.5 Ok
```

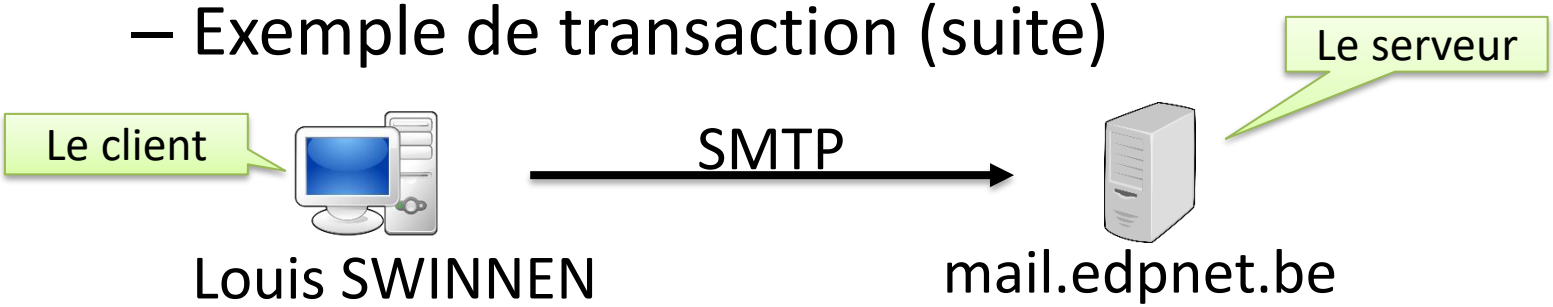

SWILABUS

Le mail (7)

– Exemple de transaction (suite)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.




```
DATA
354 End data with <CR><LF>.<CR><LF>
FROM: Admin GMAIL <admin@gmail.com>
TO: Louis SWINNEN <l.swinnen@helmo.be>

Hello,
Your account is now locked. Please go to the
following URLto unlock it !
.
250 2.0.0 Ok: queued as 3kc2Zd3cvPz5v7w
QUIT
221 2.0.0 Bye
```


SWILABUS

Le mail (8)

– Le contenu du mail

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

```
Return-Path: <admin@gmail.com>
Received: from relaygateway01.edpnet.net
 (relaygateway01.edpnet.net [212.71.1.210])
 by smtp.helmo.be (8.14.4/8.14.4) with ESMTP id t13AebMB007029
 for <l.swinnen@helmo.be>; Tue, 3 Feb 2015 11:40:43 +0100
Message-Id: <c58064$95r8rg@relaygateway01.edpnet.net>
Date: 03 Feb 2015 11:00:47 +0100
X-IronPort-Anti-Spam-Filtered: true
X-IronPort-Anti-Spam-Result: AkUIANik0FTURwGJ/2dsb2JhbABSCIMG
 gRSCNYFNg2gBrhIBS5RuBXFDAQEBAQF9hFsFD0UgOAQoBgICCBGIGgEYr3qR
 M4NTijIKGWAolGpngVWBFoE0BYEijWuKOIMDgjmITIM9IoFFAQsBgl6BOIEu
 CQMCAQ
X-IPAS-Result: AkUIANik0FTURwGJ/2dsb2JhbABSCIMGgRSCNYFNg2gBrh
 IBS5RuBXFDAQEBAQF9hFsFD0UgOAQoBgICCBGIGgEYr3qRM4NTijIKGWAolG
 pngVWBFoE0BYEijWuKOIMDgjmITIM9IoFFAQsBgl6BOIEuCQMCAQ
X-IronPort-AV: E=Sophos;i="5.09,512,1418079600";
   d="scan'";a="308126576"
Received: from unknown (HELO LXmail03.edpnet.net)
  ([212.71.1.137])
  by relaygateway01.edpnet.net with ESMTP/TLS/ADH-AES256-SHA;
  03 Feb 2015 11:00:47 +0100
```


SWILABUS

Le mail (9)

– Le contenu du mail (suite)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

```
Received: from test?localhost (213.219.155.197.adsl.dyn.
 edpnet.net [213.219.155.197])
 by LXmail03.edpnet.net (Postfix) with SMTP id 3kc2Zd3cvPz5v7w
 for <l.swinnen@helmo.be>; Tue,  3 Feb 2015 11:39:00
 +0100 (CET)

FROM: Admin GMAIL <admin@gmail.com>
TO: Louis SWINNEN <l.swinnen@helmo.be>
X-Greylist: ACL 177 matched, not delayed by
  milter-greylist-4.2.7 (smtp.helmo.be [192.168.3.208]);
  Tue, 03 Feb 2015 11:40:43 +0100 (CET)
X-HELMo-MailScanner-Information: Please contact the
  administrator for more information
X-HELMo-MailScanner-ID: t13AebMB007029
X-HELMo-MailScanner: Found to be clean
X-HELMo-MailScanner-From: admin@gmail.com
X-HELMo-MailScanner-Watermark:
  1423565086.70853@Knp6Q6k2v+qvIdtvbL1dqA
X-Spam-Status: No
```


SWILABUS

Le mail (10)

– Le contenu du mail (fin)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

```
Hello,


Your account is now locked. Please go to the following URL
to unlock it !
```


SWILABUS

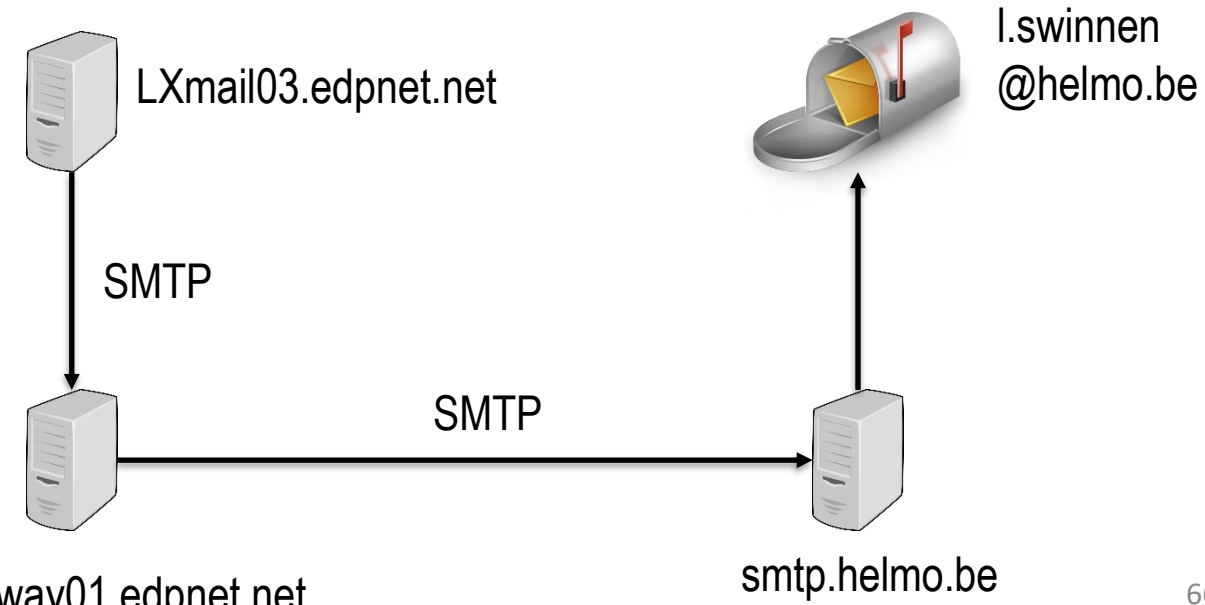
Le mail (11)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

– Chemin suivi par le message

- Grâce aux informations contenues dans l'entête du mail, il est possible de tracer le chemin suivi par le courrier, de serveur en serveur.



Le mail (12)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

- Réception du courrier
  - Exemple: POP3, TCP, port 110
  - Protocole applicatif:
    - USER <login>
    - PASS <password>
    - STAT
    - RETR <num>
    - DELE <num>
    - QUIT

Phase d'identification: login et mot de passe

Statistiques

Téléchargement du message identifié

Suppression du message identifié

Fin de l'échange

SWILABUS

Le mail (13)

– Exemple de transaction

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.


```
 $ telnet mail.helmo.be 110
+OK Dovecot ready.
USER p010544
+OK
PASS xxxxxx
+OK Logged in.
STAT
+OK 1574 160698415
RETR 1563
+OK 1789 octets
Return-Path: <admin@gmail.com>
Received: from relaygateway01.edpnet.net (relaygateway01.edp
 net.net [212.71.1.210])
 by smtp.helmo.be (8.14.4/8.14.4) with ESMTP id t13AebMB007029
 for <l.swinnen@helmo.be>; Tue, 3 Feb 2015 11:40:43 +0100
(...)
```


SWILABUS

Le mail (14)

– Exemple de transaction (suite)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.


```
 FROM: Admin GMAIL <admin@gmail.com>
TO: Louis SWINNEN <l.swinnen@helmo.be>
X-Greylist: ACL 177 matched, not delayed by
  milter-greylist-4.2.7 (smtp.helmo.be [192.168.3.208]);
  Tue, 03 Feb 2015 11:40:43 +0100 (CET)
(...)
X-Spam-Status: No

Hello,


Your account is now locked. Please go to the following URL
to unlock it !
.
QUIT
+OK Logging out.
```


SWILABUS

Le mail (15)

Plan :

- Intro.
- DNS
- **Mail**
- Web
- Eléments de progr.

– Quelques observations
  - Le serveur répond toujours pas +OK ou –ERR
  - L'authentification offerte est assez nulle dans le sens où **le login et le mot de passe ne sont pas chiffrés** (*et peuvent être interceptés durant l'échange*).
    – Solution: utiliser l'extension SSL

– Différence entre IMAP et POP3
  - Avec IMAP
    – Les messages sont conservés sur le serveur (possibilité d'y accéder depuis n'importe où)
    – Sécurisation avec SSL

Le Web (1)

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

- Le Web a été développé au CERN à Genève (http://www.cern.ch)
- Techniques derrières le web:
  - Langage de description de documents (HTML)
  - Serveur web pour délivrer ces pages
  - Développement du protocole HTTP (Hypertext Transfer Protocol)
    - Initialement, version 1.0 (RFC 1945)
    - Aujourd'hui, version 1.1 (RFC 2616/2817)
    - Demain, version 2.0 (RFC 7540)

Le Web (2)

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

– Définition l'URL
  - Uniform Ressource Locator
    – <protocol>://<machine>:<port>/<chemin>
      » Protocol peut être http, https, ftp, …
      » Machine: nom DNS, adresse IP. Si IPv6, doit être entre crochets
      » Port: application à contacter (80, 443 ou 21)

– Le programme serveur
  - Appelé serveur web, comme Apache ou IIS
  - Utilise TCP, port 80 (http) ou 443 (https)

Le Web (3)

- Fonctionnement (Requête)

Requête

Commande:

GET – Obtenir une ressource
HEAD – Info. sur une ressource
POST – Envoyer des infos
PUT – Stockage / mise à jour d'une ressource
DELETE – Suppression d'une ressource

Entête:

Authorization – Contrôle d'accès
From – mail utilisateur
Referer – Indique d'où l'on vient
User-Agent – Identification du navigateur

Corps:

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

SWILABUS

Le Web (4)

– Exemple d'échange


```
GET / HTTP/1.1
Host: www.unamur.be
User-Agent: Mozilla/5.0 (Macintosh; Intel
  Mac OS X 10.9 rv 35) Gecko /20100101 Firefox/35.0
Accept: text/xml,application/xml,application/xhtml+xml,
  text/html;q=0.9,text/plain;q=0.8, image/png,*/*; q=0.5
Accept-Language: fr,fr-fr;q=0.8,en-us;q=0.5,en
Accept-Encoding: gzip,deflate
Connection: keep-alive
```

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

SWILABUS

Le Web (5)

- Fonctionnement (réponse)

Réponse

Etat:

1XX – Information
2XX – Requête satisfaite
3XX – Redirection. Ressource déplacée. URL dans le corps
4XX – Erreur coté client
404 ➔ Not found
5XX – Erreur coté serveur
500 ➔ Internal error

Entête:

Date – Date & heure de la réponse
Content-type
Content-encoding
Content-Length
Information/format du corps
Server – Identification du serveur web

Corps:

Réponse du serveur avec la ressource demandée

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

SWILABUS

Le Web (6)

– Exemple d'échange (réponse)


```
HTTP/1.1 200 OK
Date: Tue, 03 Feb 2015 12:06:19 GMT
Server: Zope/(2.13.21, python 2.7.5, linux2) Zserver/1.1
Content-Length: 32189
Content-Language: fr
Expires: Sat, 01 Jan 2000 00:00:00 GMT
Content-Type: text/html;charset=utf-8
Cache-control: private
Set-Cookie: haproxy_server=client-6602; path=/
Via: 1.1 www.unamur.be

<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" lang="fr">

(...fichier index.html ...)
```

Plan :

- Intro.
- DNS
- Mail
- **Web**
- Eléments de progr.

SWILABUS

Eléments de programmation (1)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

- Les sockets
  - Un socket est le moyen par lequel une application peut communiquer avec une autre au moyen du réseau
  - Les sockets permettent d'utiliser les protocoles de transport TCP ou UDP
  - Les sockets sont présents dans tous les langages de programmation:
    - Python, C, C# et Java

Eléments de programmation (2)

- En Python, comme en C# et Java
  - Ces plateformes proposent de nombreux composants permettant d'interagir avec le réseau
  - Ainsi, la **notion évoluée de flux** permet d'abstraire le canal de communication
    - Un flux peut être utilisé pour lire des données depuis un fichier, mais aussi depuis le réseau.
    - On trouve **le flux d'entrée** pour lire les informations depuis l'émetteur
    - Le **flux de sortie** permet d'envoyer des informations au destinataire.

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

SWILABUS

Eléments de programmation (3)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

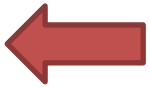
- Rôle du client (transport fiable TCP)
  - Dans un échange client / serveur, le client:
    - Se connecte au serveur (uniquement dans le cas d'un transport fiable)
    - Envoie sa requête
    - Lit la réponse depuis le serveur
    - Ferme la connexion (uniquement dans le cas d'un transport fiable)

Eléments de programmation (4)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

```
mkdir tcp_client
cd tcp_client
python3 -m venv .venv
.venv\Scripts\activate.bat
```



Création de l'environnement virtuel. A faire pour tous vos projets. PyCharm le propose automatiquement.

```
import asyncio

async def tcp_client(hote, port, message):
  reader, writer = await asyncio.open_connection(hote, port)
  print(f'Envoi de {message}')
  writer.write(message.encode())
  await writer.drain()

  data = await reader.read(5000)
  print(f'Réception de {data.decode()!r}')

  print('Fermeture de la connexion')
  writer.close()
  await writer.wait_closed()

asyncio.run(tcp_client('www.perdu.com', 80,
 'GET / HTTP/1.1\r\nHost: www.perdu.com\r\n\r\n'))
```

Source du code : https://docs.python.org/3/library/asyncio-stream.html

Eléments de programmation (5)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

– En Python, le framework *asynchrone* est utilisé pour la gestion des flux.

- **async** est placé devant la définition de la fonction
- **await** est placé devant tous les appels asynchrones *pour attendre les données*
  - Ce sont des points de synchronisation, notre programme attend que les informations arrivent
  - Pendant ce temps, le système peut être occupé à autre chose.
- Ainsi des *co-routines* sont définies (dépendantes l'une de l'autre)
  - Info: https://realpython.com/async-io-python/

Eléments de programmation (6)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

– Explication des méthodes / fonctions:

- asyncio.open_connection()
  - Permet d'ouvrir une connexion fiable (TCP) vers une destination. Cette méthode est asynchrone et il faut utiliser **await** (on continue quand la connexion est établie)
  - Paramètres:
    - » Hote: machine vers laquelle la connexion est établie (nom / IP)
    - » Port: numéro de l'application sur laquelle il faut se connecter (80 = serveur web, …)
    - » … (d'autres paramètres sont possibles)
  - Retour: un flux lecteur et écrivain

Plus d'info: https://docs.python.org/3/library/asyncio-stream.html

Eléments de programmation (7)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

- writer.write()
  - Cette fonction permet d'envoyer des informations dans le flux *écrivain.* Utilisé pour envoyer des données.
  - L'écriture est asynchrone. C'est pourquoi la méthode *drain* est présente.
- writer.drain()
  - Cette fonction doit être préfixée de **await**
  - Elle assure que l'écriture ordonnée par *write* est finalisée.
- reader.read()
  - Cette fonction permet de lire sur le flux *lecteur*.
  - Utilisé pour recevoir des données.
  - La taille maximale des données est fournie en paramètre.

Plus d'info: https://docs.python.org/3/library/asyncio-stream.html

SWILABUS

Eléments de programmation (8)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

- writer.close()
  - Demande la fermeture de la connexion établie (mode fiable, TCP)
  - Doit être fermée à la fin de l'échange
  - Cette fonctionnalité est asynchrone et l'appel à la fonction *wait_closed* assure la fermeture (point de syncrhonisation)
- writer.wait_closed()
  - Cette fonction doit être préfixée de **await**
  - Elle assure que la fermeture ordonnée par *close* est bien finalisée.
- chaine.encode()
  chaine.decode()
  ➔ Assure un même codage (UTF-8)

Plus d'info: https://docs.python.org/3/library/asyncio-stream.html

Eléments de programmation (9)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

- Rôle du serveur (transport fiable TCP)
  - Le serveur doit effectuer les opérations suivantes :
    - Ecouter, sur un port donné, qu'une connexion arrive
    - Pour chaque connexion établie, répéter:
      - Communiquer avec le client connecté
      - Fermer la connexion avec ce client
      - Attendre la connexion suivante
  - Si le serveur doit pouvoir traiter plusieurs clients en même temps, il faut une forme de concurrence (pas vu ici)

Eléments de programmation (10)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

```
import asyncio

async def process_echo(reader, writer):
  data = await reader.read(5000)
  message = data.decode()
  addr = writer.get_extra_info('peername')
  print(f'Source: {addr!r}, message: {message!r}')
  print(f'Renvoi du message {message}')
  writer.write(data)
  await writer.drain()

  print('Fermeture de la connexion')
  writer.close()

async def main():
  server = await asyncio.start_server(process_echo,
   'localhost', 8888)
  addr = server.sockets[0].getsockname()
  print(f'Ecoute sur {addr}')
  async with server:
    await server.serve_forever()

asyncio.run(main())
```

Source du code : https://docs.python.org/3/library/asyncio-stream.html

SWILABUS

Eléments de programmation (11)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

– Explication des méthodes / fonctions :

- asyncio.start_server()
  - Permet de démarrer un serveur d'écoute. Cette méthode est asynchrone et il faut utiliser **await**.
  - Paramètres:
    - » *Routine:* Fonction à appeler lorsque la connexion est établie avec un client.
    - » *IP:* Adresse de l'interface pour l'écoute, doit correspondre à l'interface utilisée par le client
      - localhost désigne la machine locale (pas de connexion depuis une autre machine)
    - » *Port:* Numéro de l'application (> 1024), doit correspondre au port utilisé par le client
    - » Retour: un *server* à l'écoute

Plus d'info: https://docs.python.org/3/library/asyncio-stream.html

SWILABUS

Eléments de programmation (12)

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**

- getsockname()
  - Permet d'obtenir des informations sur un socket, notamment l'adresse d'écoute
  - server.sockets[0] est le socket utilisé pour l'écoute
- async with server:
  await server.serve_forever()
  - Assure la libération des ressources (async with…)
  - serve_forever() assure que le serveur continue de servir les clients qui se connectent, les uns après les autres, sans s'arrêter.

Plus d'info: https://docs.python.org/3/library/asyncio-stream.html

Exercice

- Ecrivez en **Python**
  - Un programme serveur
    - Permettant à un serveur de recevoir une chaîne composée comme suit: $chiffre_1$ opération $chiffre_2$
    - Retourne la valeur calculée
- Ecrivez en **Python**
  - Un programme client
    - Permettant d'interagir avec ce serveur, saisissant l'opération, transmettant celle-ci au serveur et affichant le résultat reçu.

Plan :

- Intro.
- DNS
- Mail
- Web
- **Eléments de progr.**