

Applications réseaux

adresses IP,
interfaces réseaux
et résolution de nom

Etienne Duris
Arnaud Carayol

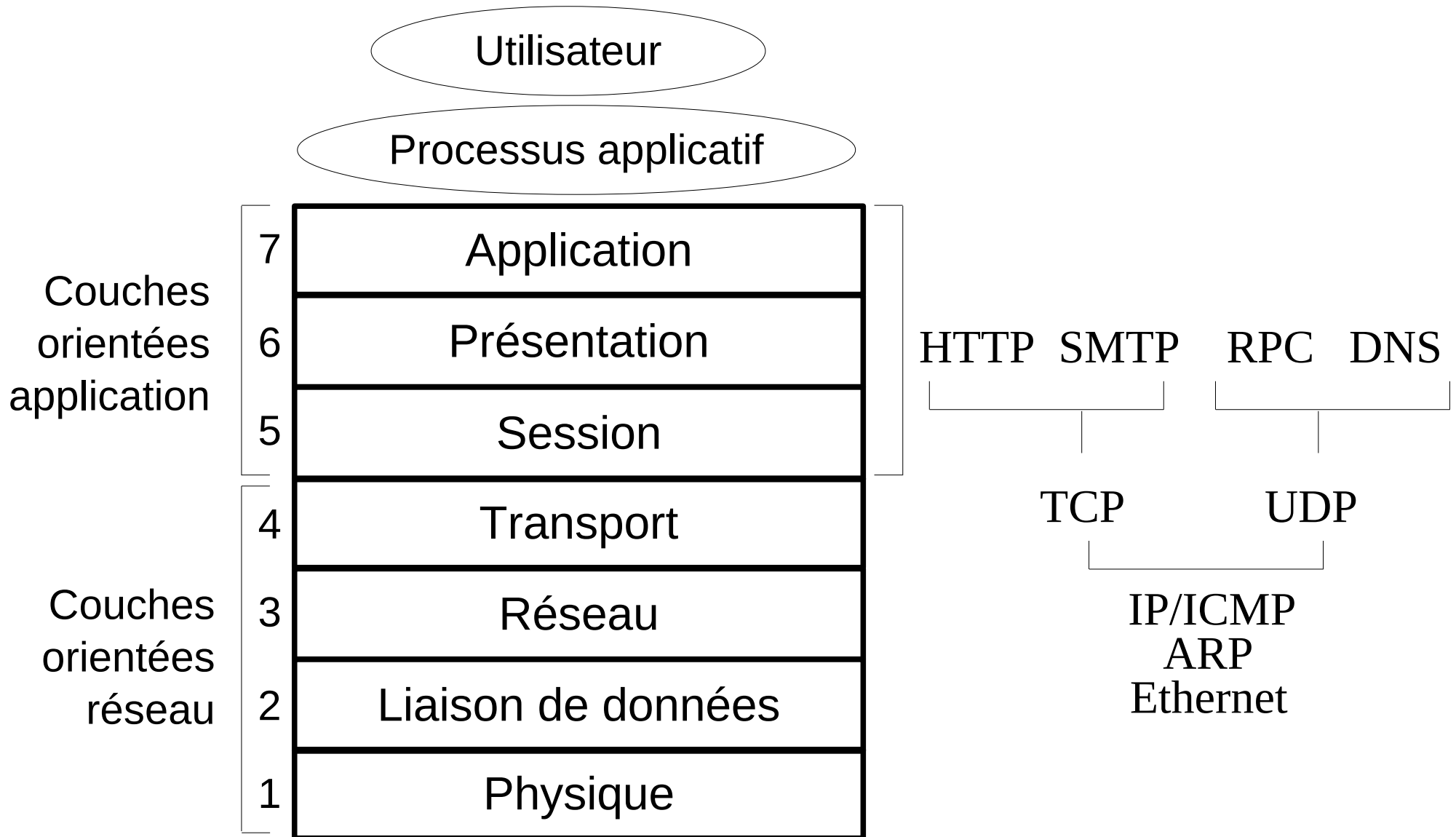
Bibliographie et sources

- *Java et Internet*
G. Roussel, E. Duris, N. Bedon et R. Forax. Vuibert 2002.
- Les cours de Rémi Forax
<http://igm.univ-mlv.fr/~forax/>
- Documentations Java Oracle
<http://docs.oracle.com/javase/>
- Java Network Programming, Third Edition, Elliotte Rusty Harold, O'Reilly

Contexte

- On s'intéresse aux applications de type Internet
 - Interconnexion des réseaux utilisant le protocole Internet (IP)
 - La couche IP sert à **adresser**, **acheminer**, **router** les paquets IP
 - Pour échanger des informations (être utilisé), le protocole Internet de la **couche réseau (3)** requiert un protocole « compagnon » de la **couche transport (4)**
 - UDP (*User Datagram Protocol*) ou
 - TCP (*Transmission Control Protocol*)
 - La grande majorité des applications (DNS, SMTP, FTP, HTTP, etc.) reposent sur cette suite dite « TCP/IP »
- L'accès aux « couches basses » (1 et 2) est plus délicat
 - Dépendant du protocole, du système d'exploitation (libpcap)

Open System Interconnection (OSI)



Standards et normes

- Adoptés par l'IAB (*Internet Architecture Board*)
 - IRTF (*Internet Research Task Force*) : long terme
 - IETF (*Internet Engineering Task Force*) : court terme
- Distribués par l'INTERNIC (*INTERnet Network Information Center*)
 - sous la forme de documents appelés **RFC** (*Request For Comments*)
 - www.rfc-editor.org
 - www.irtf.org
 - www.ietf.org

Gestion des adresses Internet

- Les noms et numéros sont gérés par
 - IANA (*Internet Assigned Numbers Authority*)
 - www.iana.org
 - ICANN (*Internet Corporation for Assigned Names and Numbers*)
 - www.icann.org
- Délèguent leurs fonctions au niveau régional
 - RIPE NCC (Réseaux IP Européens, Network Coordination Center) pour une partie de l'Europe de l'Est et l'Afrique
 - Délègue, au niveau national, à des LocalIR (*Internet Registry*). GEANT (europe) RENATER (france)
 - Délègue à l'administrateur (ex. Université)

Adresses réservées

- L'adresse du **réseau** lui même
 - Tout à zéro pour ce qui concerne les bits « machine »
- L'adresse de **broadcast**
 - Broadcast réseau local (tout à 1): 255.255.255.255
 - Broadcast dirigé: tout à 1 pour ce qui concerne les bits « machine » (souvent bloqué par les routeurs)
- L'adresse « **non spécifiée** »: 0.0.0.0 (*wildcard* ou *anylocal*)
 - Elle représente la machine locale avant qu'elle ait une adresse
- L'adresse de **loopback**: 127.0.0.1
- Plages d'adresses réservées non affectées (RFC 1918)
 - 10.0.0.0/8 et 127.0.0.0/8 pour la classe A
 - 169.254.0.0/16 et 172.16.0.0/16 pour la classe B
 - 192.168.0.0/16 pour la classe C

Protocole Internet (rappels)

- RFC 791 (Jon Postel) et STD 0005
 - Acheminement de datagrammes d'un point à un autre du réseau, repérés par des adresses IP
- Principe de **roulage de paquet**
 - Chaque datagramme est « routé » indépendamment des autres (**plusieurs chemins possibles**, **non conservation de l'ordre**)
 - Lorsqu'un datagramme est transmis, rien n'assure qu'il arrivera un jour: **non fiable** ou **au mieux** (*best effort*)
- Traversée de plusieurs réseaux physiques différents
 - possibilité de **fragmentation** des paquets

Format des datagrammes (IP v4)

0	4	8	12	16	20	24	28
Version	Taille (mots de 32 bits)	Service (TOS)		Taille totale, en-tête compris (en octets)			
Identificateur				Marq.	Décalage du fragment		
Durée de vie		Protocole (6: TCP, 17: UDP, etc...)		Somme de contrôle			
Adresse IP émetteur							
Adresse IP destinataire							
[Options éventuelles...							
...options éventuelles...							
...options éventuelles]					Bourrage (pour compléter à un mot de 32)		

Type Of Service	0	1	2	3	4	5	6	7
	Précédence	Délai	Débit	Sureté	Coût	0		

Le protocole IP et Java

- API d'accès: **java.net.***;
- Adresses IP représentées par les instances de la classe **InetAddress**, ou plus précisément de l'une des deux sous-classes
 - **Inet4Address** pour le protocole IPv4 (32 bits)
 - **Inet6Address** pour le protocole IPv6 (128 bits)
- *A priori*, chaque instance représente 2 informations
 - un tableau d'octets (4 ou 16): adresse numérique (**address**)
 - un nom éventuel (DNS, NIS): adresse « littérale » (**host**)
 - Si ce nom existe ou si la résolution de nom inverse a déjà été faite

Représentation des adresses

- L'adresse « littérale » n'est pas nécessairement résolue (peut être remplacée par la chaîne vide)
 - méthode **String toString()** ⇒
www.6bone.net/131.243.129.43 ou encore
www.6bone.net/3ffe:b00:c18:1:0:0:0:10
 - méthodes **String getHostName()** ⇒ « www.6bone.net » ou
String getCanonicalHostName() ⇒ « 6bone.net »
 - méthode **String getHostAddress()**
⇒ "131.243.129.43" (adresse numérique sous forme de String)
 - méthode **byte[] getAddress()**
⇒ {131, 243, 129, 43} (attention, en Java, le type **byte** comme tous les types primitifs numériques est **signé!**)

Récupération d'une instance (méthodes statiques)

- Adresse IP de la machine locale
 - `InetAddress.getLocalHost()` (éventuellement adresse de loopback)
- Étant donnée une adresse littérale ou numérique
 - `InetAddress.getByName(String host)`
 - `InetAddress.getAllByName(String host)` // tableau
- Étant donné un tableau d'octets (pas de résolution DNS)
 - `InetAddress.getByAddress(byte[] addr)` // non bloquant
- Étant donné un nom et un tableau d'octets (idem)
 - `InetAddress.getByAddress(String host, byte[] addr)`
 - création, aucun système de nom interrogé pour vérifier la validité

Exemples de représentations

```
> InetAddress ia1 = InetAddress.getByName("java.sun.com");
System.out.println(ia1.toString());      // java.sun.com/192.18.97.71
System.out.println(ia1.getHostName());   // java.sun.com
System.out.println(ia1.getCanonicalHostName()); // flres.java.Sun.COM
System.out.println(ia1.getHostAddress()); // 192.18.97.71

> InetAddress ia2 = InetAddress.getByName("192.18.97.71");
System.out.println(ia2.toString());      // /192.18.97.71
System.out.println(ia2.getHostName());   // flres.java.Sun.COM
System.out.println(ia2.getCanonicalHostName()); // flres.java.Sun.COM
System.out.println(ia2.getHostAddress()); // 192.18.97.71

> byte[] b = ia2.getAddress();           // affiche b[0]=-64
for(int i=0; i<b.length; i++) {          //          b[1]=18
    System.out.println("b["+i+"]="+b[i]); //          b[2]=97
}                                          //          b[3]=71
```

Exemples de représentation (suite)

```
> InetAddress [] ias = InetAddress.getAllByName("www.w3.org");
for(int i = 0; i < ias.length; i++) // affiche: www.w3.org/18.29.1.34
    System.out.println(ias[i]);      //          www.w3.org/18.29.1.35
                                     //          www.w3.org/18.7.14.127
```

- Toutes ces méthodes peuvent lever des exceptions de classe
`UnknownHostException`

- si le format des arguments est incorrect
- si la résolution de nom n'aboutit pas

```
> InetAddress bidon =
    InetAddress.getByAddress("n.importe.quoi", new byte[]{1,2,3,4});
System.out.println(bidon);          // affiche : n.importe.quoi/1.2.3.4
bidon = InetAddress.getName("n.importe.quoi");
                                     // lève java.net.UnknownHostException
```

Observations sur les adresses

<i>méthode</i>	<i>IP v4</i>	<i>IP v6</i>	<i>adresses concernées</i>
<code>isAnyLocalAddress()</code>	0.0.0.0/8	::0	<i>non spécifiée</i>
<code>isLoopbackAddress()</code>	127.0.0.0/8	::1	<i>Loopback</i>
<code>isLinkLocalAddress()</code>	169.254.0.0/16	fe80::/16	<i>locale au lien (a)</i>
<code>isSiteLocalAddress()</code>	10.0.0.0/8 172.16.0.0/12 192.168.0.0/16	fec0::/16	<i>locale au site (b)</i>
<code>isMulticastAddress()</code>	224.0.0.0/4	ff00::/8	<i>Multicast</i>
<code>isMCGlobal()</code>	de 224.0.1.0 à 238.255.255.255	ffxe::/16	<i>multicast portée globale</i>
<code>isMCOrgLocal()</code>	239.192.0.0/14 (a)	ffx8::/16	<i>mult. port. org.</i>
<code>isMCSiteLocal()</code>	239.255.0.0/16 (a)	ffx5::/16	<i>mult. port. site</i>
<code>isMCLinkLocal()</code>	224.0.0.0/24 (a)	ffx2::/16	<i>mult. port. lien</i>
<code>isMCNodeLocal()</code>	aucune (b)	ffx1::/16	<i>mult. port. noeud</i>

(a) sous réserve de valeurs de TTL adéquates.

(b) Seul un champ TTL à 0 donne une portée locale en IP v4.

Interfaces de réseau

- Une interface de réseau est représentée par un objet de la classe `java.net.NetworkInterface`
 - un nom (lo, eth0, etc..)
 - une liste d'adresses IP associées à cette interface
- Les méthodes **statiques** permettant d'obtenir des objets de cette classe:
 - `java.util.Enumeration<NetworkInterface> getNetworkInterfaces()`
 - `NetworkInterface getByName(String name)`
 - `NetworkInterface getByInetAddress(InetAddress addr)`
- Peuvent lever des exceptions `SocketException`

Exemple de représentation

- ```
for(Enumeration<NetworkInterface> e =
 NetworkInterface.getNetworkInterfaces();
 e.hasMoreElements();)
 System.out.println(e.nextElement());
// affiche:
// name:eth0 (eth0) index: 2 addresses:
// /193.12.34.56;
//
// name:lo (lo) index: 1 addresses:
// /127.0.0.1;
```
- **getInetAddresses()** sur une instance renvoie une énumération des adresses IP associées.
  - Principalement utilisé pour spécifier un attachement multicast sur une interface donnée, ou pour dissocier une carte Wifi d'une Ethernet avant qu'elles aient une adresse IP.

# Test de connectivité

---

- Depuis jdk1.5, possibilité de faire une sorte de ping
  - Depuis une instance de la classe `InetAddress`, teste si l'adresse qu'elle représente est « atteignable » (*reachable*)
    - Implantation au mieux, mais firewall ou config de serveur peuvent faire en sorte que ça échoue (retourne false) alors que `I@IP` est atteignable sur certains port
    - Typiquement, tente un envoi de ECHO REQUEST en ICMP (si privilège ok), ou alors tente d'établir une connexion TCP avec le port 7 (Echo)
  - `public boolean isReachable(int timeout) throws IOException`
    - Au delà de `timeout` millisecondes, levée de l'exception
  - `public boolean isReachable(NetworkInterface netif, int ttl, int timeout) throws IOException`
    - Permet de spécifier l'interface de sortie (null pour n'importe laquelle)
    - et le nombre de sauts maximum des paquets (0 pour défaut)

# Test de connectivité : exemples

```
public class NameResolution {
 public static void main(String[] args) throws IOException {
 InetAddress inet = InetAddress.getByName(args[0]);
 System.out.println(inet);
 System.out.println("Test de connectivité...");
 if (inet.isReachable(500))
 System.out.println(" réussi.");
 else
 System.out.println(" échoué.");
 }
}
```

➤ **User:** tentative connexion TCP

- Accepte ou refuse, mais en général répond

➤ **Root:** ping (ICMP)

- Si ne réponds pas aux ICMP Request, échec
- Si répond, succès

```
User$> java NameResolution www.w3c.org
www.w3c.org/128.30.52.45
Test de connectivité... réussi.
```

```
Root# java NameResolution www.w3c.org
www.w3c.org/128.30.52.45
Test de connectivité... échoué.
```

```
Root# java NameResolution gaspard
gaspard.univ-mlv.fr/193.55.63.81
Test de connectivité... réussi.
```

(Untitled) - Wireshark

File Edit View Go Capture Analyze Statistics Help

| No. | Time     | Source       | Destination  | Protocol | Info                                                       |
|-----|----------|--------------|--------------|----------|------------------------------------------------------------|
| 4   | 1.882639 | 10.1.54.184  | 10.1.0.3     | DNS      | Standard query A www.w3c.org                               |
| 7   | 1.888399 | 10.1.0.3     | 10.1.54.184  | DNS      | Standard query response CNAME dolph.w3c.org A 128.30.52.45 |
| 3   | 1.888924 | 10.1.54.184  | 128.30.52.45 | TCP      | 40464 → 7 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 TSV=4870259  |
| 2   | 1.892168 | 128.30.52.45 | 10.1.54.184  | ICMP     | Destination unreachable (Port unreachable)                 |

Tentative "user" sur www.w3c.org  
 connection refusée sur port 7 TCP  
 ip.isReachable(500) est vrai!

| No. | Time     | Source      | Destination  | Protocol | Info                                                       |
|-----|----------|-------------|--------------|----------|------------------------------------------------------------|
| 5   | 2.086345 | 10.1.54.184 | 10.1.0.3     | DNS      | Standard query A www.w3c.org                               |
| 7   | 2.086860 | 10.1.0.3    | 10.1.54.184  | DNS      | Standard query response CNAME dolph.w3c.org A 128.30.52.45 |
| 3   | 2.090306 | 10.1.54.184 | 128.30.52.45 | ICMP     | Echo (ping) request                                        |

Tentative "root" sur ip www.w3c.org  
 ICMP request n'a pas de réponse  
 ip.isReachable(500) est faux!

| No. | Time     | Source       | Destination  | Protocol | Info                                   |
|-----|----------|--------------|--------------|----------|----------------------------------------|
| 1   | 1.882639 | 10.1.54.184  | 10.1.0.3     | DNS      | Standard query A gaspard.univ-mlv.fr   |
| 4   | 1.888399 | 10.1.0.3     | 10.1.54.184  | DNS      | Standard query response A 193.55.63.81 |
| 5   | 1.888924 | 10.1.54.184  | 193.55.63.81 | ICMP     | Echo (ping) request                    |
| 3   | 1.892168 | 193.55.63.81 | 10.1.54.184  | ICMP     | Echo (ping) reply                      |

Tentative "root" sur ip gaspard  
 ICMP request obtient une réponse  
 ip.isReachable(500) est vrai!