Réseaux TCP/IP et Programmation Java

Emmanuel BRUNO
Université de Toulon, LIS UMR CNRS 7020
emmanuel.bruno@univ-tln.fr

2026-11-01

Table des matières

1 Fondamentaux des réseaux 2
1.1 Adresses IPv4 2
1.2 Adresse Format IPv6 3
1.3 Masques de sous-réseau 4
1.4 Résolution DNS 4
2 Ports et services 4
2.1 Encapsulation des données 5
3 Exemple de topologie réseau 5
3.1 Exemple de Communication 7
4 Sockets TCP 8
4.1 Création d'un serveur TCP 8

1 Fondamentaux des réseaux

Modèle OSI

Couche	Fonction	Protocoles
Application	Accès aux services réseaux	HTTP, FTP, SMTP
Présentation	Format des données	Encodage, Compression
Session	Gestion des sessions	Auth, Autorisation
Transport	Fiabilité bout en bout	TCP, UDP
Réseau	Routage	IP, ICMP
Liaison	Accès au média	Ethernet, WiFi
Physique	Transmission bits	Signaux électriques

Modèle TCP/IP

Couche	Fonction	Protocoles
Application	Services utilisateur	HTTP, FTP, SMTP, DNS
Transport	Connexion bout en bout	TCP, UDP
Internet	Routage et adressage	IP, ICMP, ARP
Accès réseau	Interface physique	Ethernet, Wi-Fi, PPP

1.1 Adresses IPv4

- Historique
- 32 bits divisés en 4 octets (x.x.x.x)
- Plages d'adresses :
  - Privées :
    - 10.0.0.0/8
    - 172.16.0.0/12
    - 192.168.0.0/16
  - Publiques : Tout le reste
  - Spéciales :

  - 127.0.0.1 (localhost)
  - 0.0.0.0 (any)
  - 255.255.255.255 (broadcast)
- Plus d'adresses disponibles au niveau mondial

```java
// Exemple IPv4 en Java

// How to get Hostname and IP Address
String hostname = InetAddress.getLocalHost().getHostName();
String ipAddress = InetAddress.getLocalHost().getHostAddress();

System.out.println("Hostname: " + hostname);
System.out.println("IP Address: " + ipAddress);
```

```
Hostname: fb2abddef070
IP Address: 172.17.0.2
```

```java
// Travailler avec une adresse IPv4 spécifique
InetAddress ipv4 = InetAddress.getByName("192.168.1.1");

System.out.println("Adresse loopback ? " + ipv4.isLoopbackAddress());
System.out.println("Adresse privée ? " + ipv4.isSiteLocalAddress());
```

```
Adresse loopback ? false
Adresse privée ? true
```

1.2 Adresse Format IPv6

- Motivations: épuisement des adresses IPv4, sécurité, qualité de service, mobilité, auto-configuration, ...
- 128 bits divisés en 8 groupes de 16 bits (xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx)
- Notation raccourcie : les zéros peuvent être omis (::1 pour localhost)

```java
// Exemple IPv6 en Java
InetAddress                                                    ipv6
= InetAddress.getByName("2001:0db8:85a3:0000:0000:8a2e:0370:7334");
System.out.println("Est IPv6 ? " + (ipv6 instanceof Inet6Address));
```

```
Est IPv6 ? true
```

1.3 Masques de sous-réseau

- Séparation des adresses réseau et hôte
- Utilité : segmentation du réseau
- Notation CIDR : /24 équivaut à 255.255.255.0
- Calcul des adresses réseau :

```
// Exemple de manipulation de sous-réseau
NetworkInterface ni = NetworkInterface.getByName("eth0");
for (InterfaceAddress ia : ni.getInterfaceAddresses()) {
    System.out.printf("Adresse: %s/%d%n",
        ia.getAddress(),
        ia.getNetworkPrefixLength());
}
```

```
Adresse: /172.17.0.2/16
```

1.4 Résolution DNS

- Traduction des noms de domaine en adresses IP
- DNS : Domain Name System

```
// Résolution DNS complète
InetAddress[] addresses = InetAddress.getAllByName("www.lis-lab.fr");
for (InetAddress addr : addresses) {
    System.out.println("Nom canonique : " + addr.getCanonicalHostName());
    System.out.println("Adresse IP : " + addr.getHostAddress());
}
```

```
Nom canonique : rproxy.lis-lab.fr
Adresse IP : 139.124.22.49
```

2 Ports et services

- Communication entre applications
- Numéro de port (0-65535)
- Convention de numérotation
  - Ports réservés (0 à 1023) : services bien connus
    - 80/443 : HTTP/HTTPS, 22 : SSH
  - Ports enregistrés (1024-49151) : applications enregistrées
  - Ports dynamiques (49152-65535) : applications temporaires

```java
// Vérification de port disponible
try (ServerSocket socket = new ServerSocket(0)) { // 0 = port automatique
    System.out.println("Port libre trouvé : " + socket.getLocalPort());
    System.out.println("Adresse : " + socket.getInetAddress());
}
```

```
Port libre trouvé : 45865
Adresse : 0.0.0.0/0.0.0.0
```

2.1 Encapsulation des données

```
//| echo: false
//| output: true
%%plantUML
@startuml
skinparam componentStyle rectangle
skinparam backgroundColor white
skinparam defaultFontSize 14

rectangle "Encapsulation" {
  [Données HTTP] as HTTP
  [Segment TCP\nPort src/dst] as TCP
  [Paquet IP\nIP src/dst] as IP
  [Trame Ethernet\nMAC src/dst] as ETH

  HTTP -down-> TCP : encapsule
  TCP -down-> IP : encapsule
  IP -down-> ETH : encapsule

  note right of HTTP : En-têtes + Data
  note right of TCP : Port 80, 443...
  note right of IP : IPv4/IPv6
  note right of ETH : 00:1A:2B:3C:4D:5E
}
@enduml
```

3 Exemple de topologie réseau

```
//| echo: false
//| output: true
%%plantUML
@startuml Network Topology
```

```
set namespaceSeparator none

!define RECTANGLE class

' Custom styles
skinparam class {
    BackgroundColor White
    BorderColor Black
    ArrowColor Black
}

package "Réseau Local 1 (192.168.1.0/24)" {
    RECTANGLE PC1 {
        MAC: 00:1A:2B:3C:4D:5E
        IP: 192.168.1.10
    }
    RECTANGLE PC2 {
        MAC: 00:1A:2B:3C:4D:5F
        IP: 192.168.1.11
    }
    RECTANGLE SW1 {
        Switch 1
    }
    RECTANGLE GW1 {
        MAC: 00:1A:2B:3C:4D:1A
        IP LAN: 192.168.1.1
        IP WAN: 203.0.113.1
    }
    RECTANGLE DHCP1 {
        IP: 192.168.1.2
        Plage: 192.168.1.100-200
    }
}

package "Réseau Local 2 (192.168.2.0/24)" {
    RECTANGLE PC3 {
        MAC: 00:2B:3C:4D:5E:6F
        IP: 192.168.2.10
    }
    RECTANGLE PC4 {
        MAC: 00:2B:3C:4D:5E:70
        IP: 192.168.2.11
    }
    RECTANGLE SW2 {
```

```
        Switch 2
    }
    RECTANGLE GW2 {
        MAC: 00:2B:3C:4D:5E:1B
        IP LAN: 192.168.2.1
        IP WAN: 203.0.113.2
    }
    RECTANGLE DHCP2 {
        IP: 192.168.2.2
        Plage: 192.168.2.100-200
    }
}

cloud Internet {
    RECTANGLE SRV {
        IP: 198.51.100.10
        DNS: server.example.com
    }
    RECTANGLE DNS {
        IP: 198.51.100.53
        DNS Racine
    }
}

' Connections
PC1 -- SW1
PC2 -- SW1
SW1 -- GW1
SW1 -- DHCP1
PC3 -- SW2
PC4 -- SW2
SW2 -- GW2
SW2 -- DHCP2
GW1 -- Internet
GW2 -- Internet
Internet -- SRV
Internet -- DNS

@enduml
```

3.1 Exemple de Communication

```
//| echo: false
//| output: true
```

```
%%plantUML
@startuml
skinparam backgroundColor white
skinparam sequenceMessageAlign center
skinparam defaultFontSize 14

participant "Client\n192.168.1.10" as C
participant "Router\n203.0.113.1" as R
participant "Server\n198.51.100.10" as S

note over C, S: Établissement connexion TCP (3-way handshake)

C → R: SYN (seq=x)
R → S: SYN (seq=x)
S → R: SYN-ACK (seq=y, ack=x+1)
R → C: SYN-ACK (seq=y, ack=x+1)
C → R: ACK (ack=y+1)
R → S: ACK (ack=y+1)

note over C, S: HTTP Request/Response

C → R: GET /index.html HTTP/1.1
R → S: GET /index.html HTTP/1.1
S → R: HTTP/1.1 200 OK
R → C: HTTP/1.1 200 OK

@enduml
```

4 Sockets TCP

- Communication orientée connexion
- Fiabilité et ordre des messages
- Exemple : HTTP, FTP, SSH

4.1 Création d'un serveur TCP

```
[INFO] Target directory: /home/jovyan/work/github/ebpro/sample-java-tcp-client-server
[INFO] Cloning new repository to /home/jovyan/work/github/ebpro/sample-java-tcp-client-server
[INFO] Repository cloned successfully
```

Les exemples suivants sont accessibles dans le dépôt :

: ebpro/sample-java-tcp-client-server

Détails

- **URL**: https://github.com/ebpro/sample-java-tcp-client-server
- **Branch**: develop
- **Local Path**: `/home/jovyan/work/github/ebpro/sample-java-tcp-client-server`
- **Latest Commit**: acd9484 - chore: add _quarto-utils submodule + .devcontainer (SOTA pattern)
- **Timestamp**: 2026-11-01T12:17:57Z

Clone Command:

```
git clone -b develop https://github.com/ebpro/sample-java-tcp-client-server
```