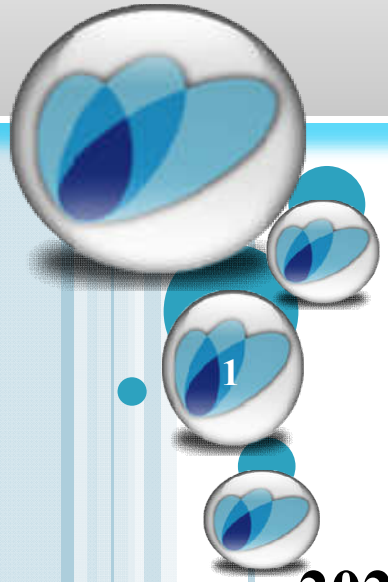
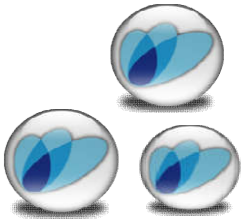
Cours Construction d'Applications Réparties (CAR)

1ère Année Master Génie Logiciel





2020-2021

Mme Z.LAAREDJ

Les sockets

Plan

- Rappel sur les réseaux
- Introduction aux sockets
- Sockets en mode connecté
- Sockets en mode non-connecté
- Adressage IP en JAVA

Rappel sur les réseaux

➢ Le modèle de référence OSI

N°	Couche	Rôle selon la norme ISO
7	Application	Interprétation des données, protocoles applicatifs (HTTP, FTP, SMTP, POP, etc.)
6	Présentation	Mise en forme des informations
5	Session	Gestion de sessions
4	Transport	Transfert de bout en bout, contrôle de flux (transfert fiable entre deux applications)
3	Réseau	Routage, contrôle de congestion
2	Liaison de données	Transfert local des trames de données, gestion d'accès au médium, détection et correction d'erreurs de transmission
1	Physique	Connexion au réseau, transmission des données binaires sur un support physique

Rappel sur les réseaux

➢ OSI - TCP/IP

	OSI		TCP/IP (TCP/IP)
7	Application	4	Application
6	Présentation		
5	Session		
4	Transport	3	Transport
3	Réseau	2	Internet
2	Liaison de données	1	Réseau
1	Physique		

Rappel sur les réseaux

➢ Modèle OSI de communication en couches

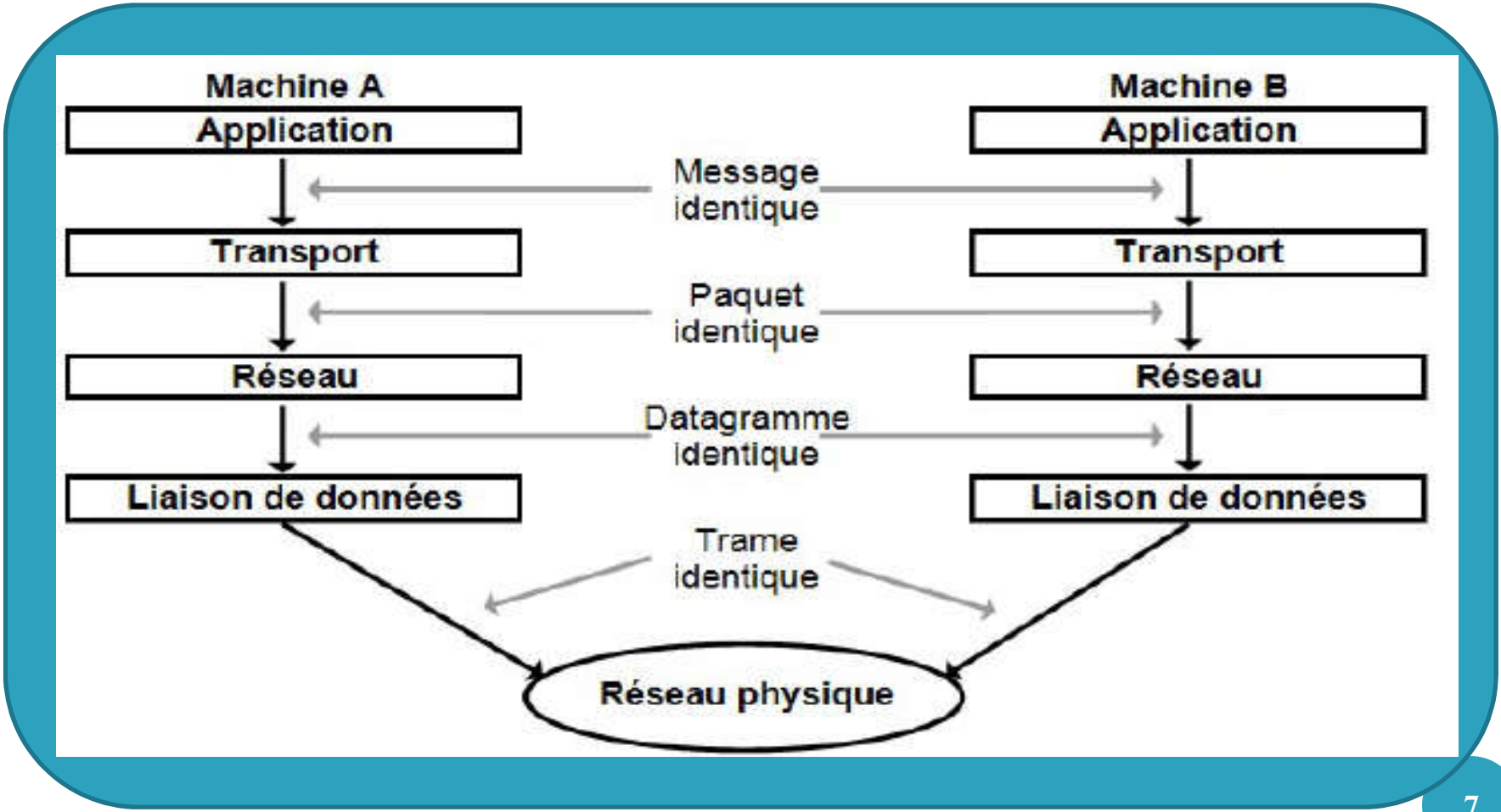
- Chaque couche a un rôle particulier
- Pour communiquer avec une autre entité, une couche utilise les services de sa couche locale inférieure
- Une couche d'une entité communique avec une couche de même niveau d'une autre entité en respectant un certain protocole de communication

➢ **Données échangées entre 2 couches PDU:** trames ou paquets [ou segments ou messages]

- Taille bornée
- Structure de données: deux parties: PDU = PCI + SDU
  - SDU: Données de la couche supérieure à transmettre
  - PCI: Données de contrôle de la communication entre couches

Rappel sur les réseaux

➢ **Unités d'échanges entre les différentes couches**



Rappel sur les réseaux

- Réseaux locaux, internet ...
- **Couche réseau :** IP (Internet Protocol)
  - Identification des machines dans les réseaux
  - Recherche des routes à travers le réseau pour accéder à une machine distante
  - Contrôle de congestion dans le réseau

Rappel sur les réseaux

➢ Couche transport

TCP (Transmission Control Protocol)

- Connexion virtuelle directe et fiable entre 2 applications
- Orienté connexion, point à point, bidirectionnel, fiable, conserve le séquencement
- TSAP : socket : (@ip + port)

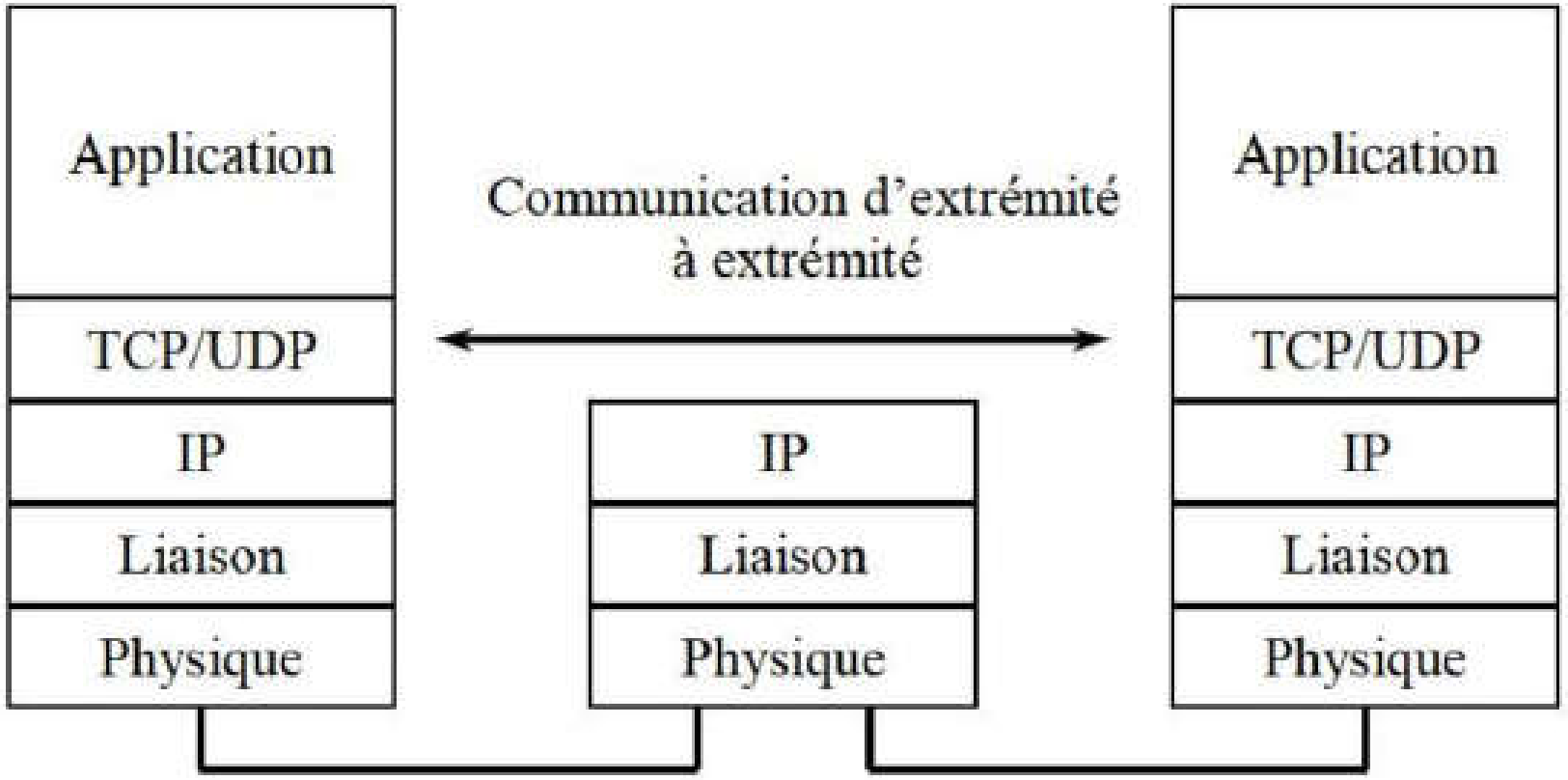
UDP (User Datagram Protocol): mode datagramme

- Envoi de paquets de données
- Pas de gestion de l'ordre d'arrivée, absence de gestion des paquets perdus
- Mode datagramme, UDP rajoute à IP l'information de service (port)

Rappel sur les réseaux

➢ Protocoles de transport Internet: TCP et UDP

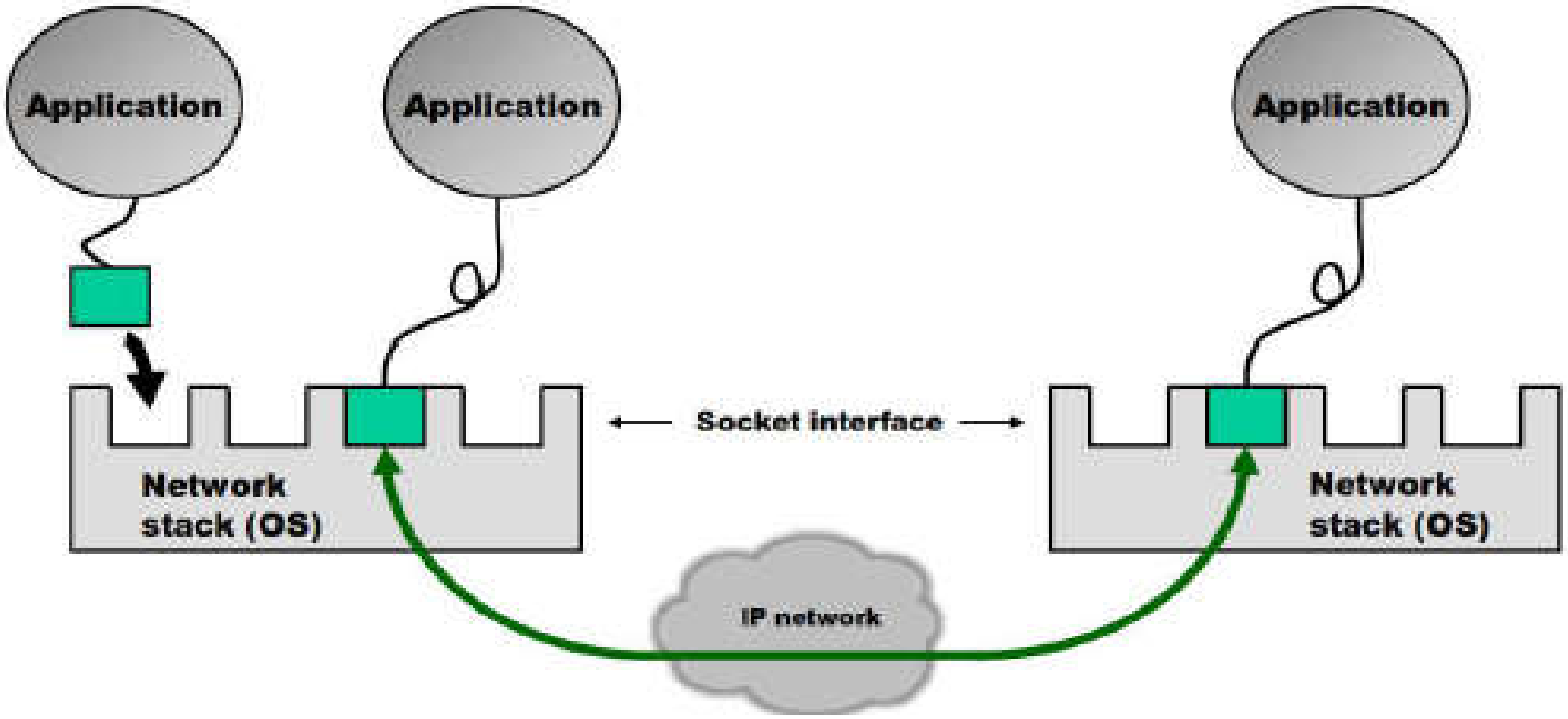
- Communication entre deux systèmes de bout en bout
- Transparence (pas de visibilité) des systèmes intermédiaires (routeurs)



Introduction aux sockets

➢ Définition:

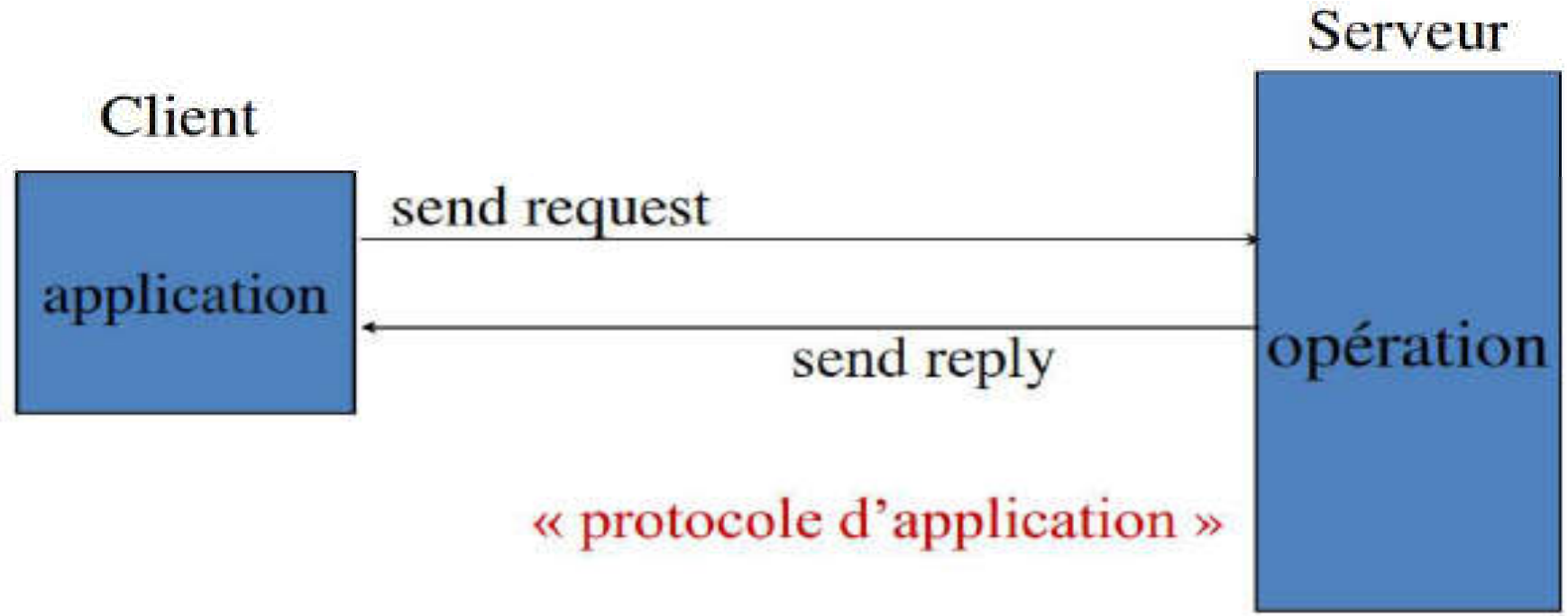
- Une socket est une interface qui permet à une application de se connecter à la pile du réseau d'un hôte
- Une fois connectée, l'application peut échanger des données avec d'autres processus d'une manière « bidirectionnelle »



Introduction aux sockets

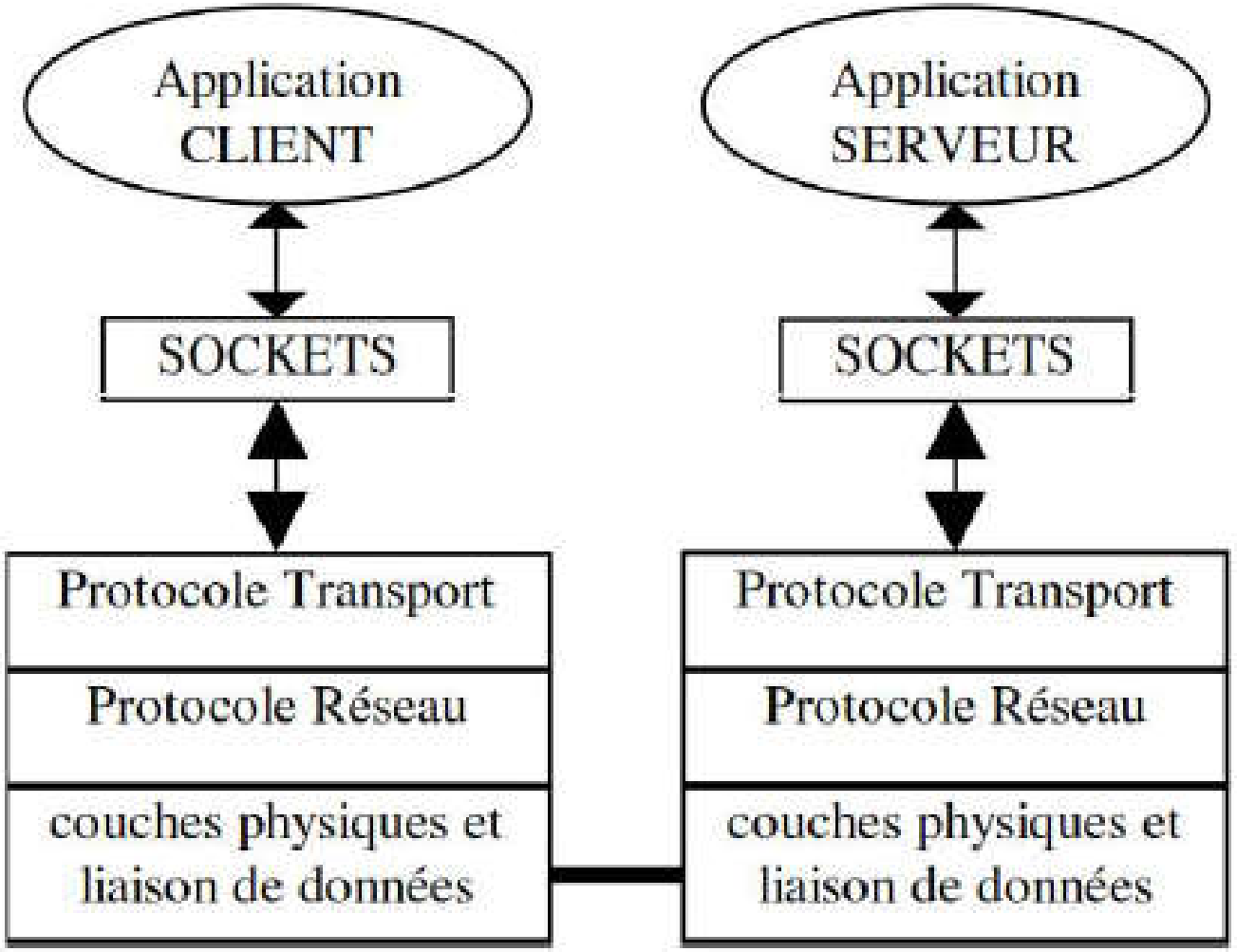
➢ Modèle C/S

- Mode de communication qu'un hôte établit avec un autre hôte qui fournit un service quelconque



Introduction aux sockets

➢ Modèle C/S



Les sockets –les N°de ports

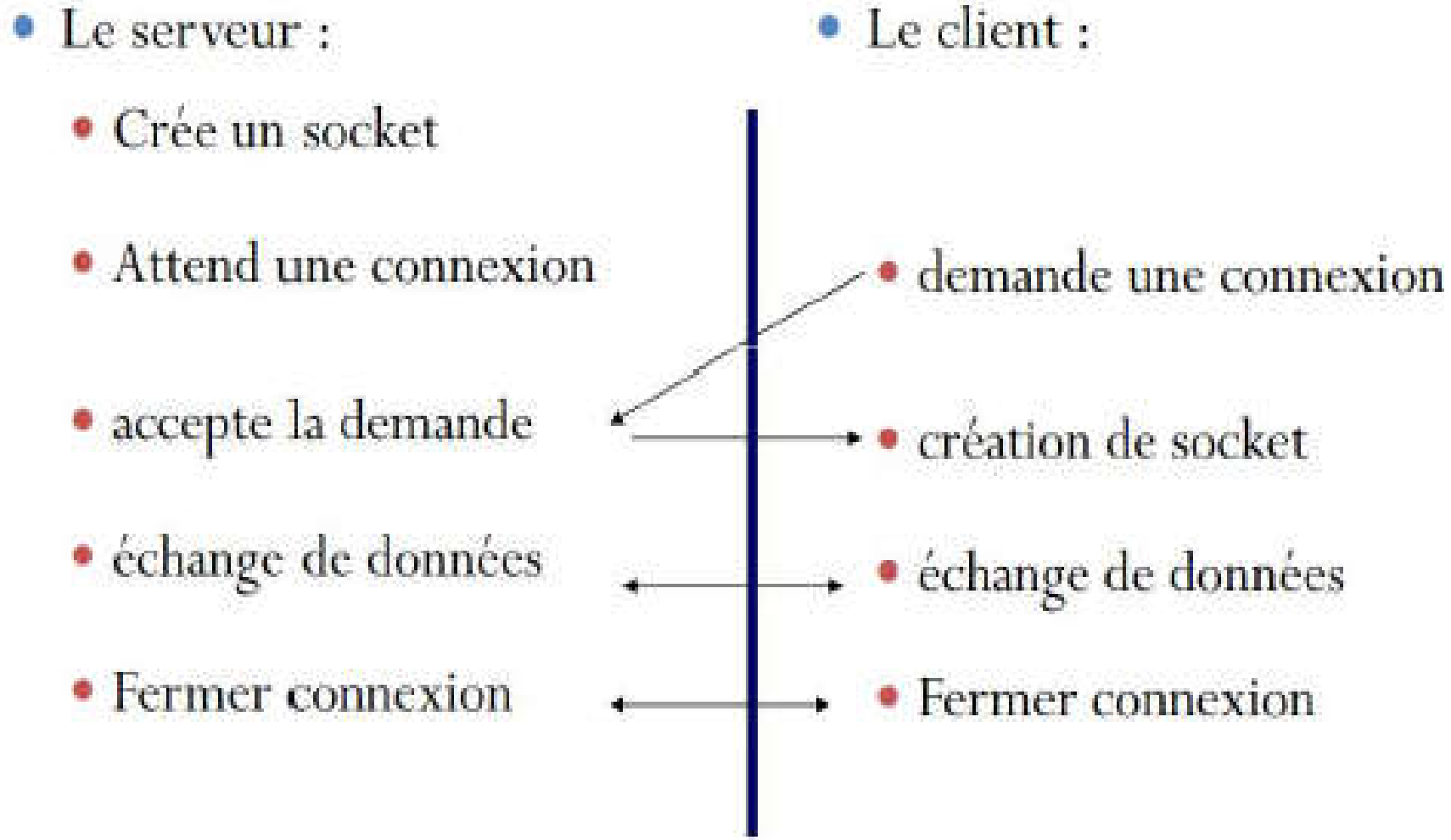
➢ Numéros de ports des applications

- Assignation standard mise au point par l'**IANA** (**I**nternet **A**ssigned **N**umbers **A**uthority) :
  - Les ports 0 à 1023 sont les «ports reconnus» ou réservés («Well Known Ports»). Ils sont, de manière générale, réservés aux processus système (démons) ou aux programmes exécutés par des utilisateurs privilégiés. Un administrateur réseau peut néanmoins lier des services aux ports de son choix,
  - Les ports 1024 à 49151 sont appelés «ports enregistrés» («Registered Ports»),
  - Les ports 49152 à 65535 sont les «ports dynamiques et/ou privés» («Dynamic and/or Private Ports»)

Les sockets en mode connecté

Client	Serveur (en écoute du canal)
1. Crée une socket 2. Se connecte au serveur en donnant l'adresse socket distante (adresse IP du serveur et numéro de port du service). Le système d'exploitation attribue pour cette connexion automatiquement un numéro de port local au client 3. Lit et/ou écrit (envoi et réception des messages) sur la socket 4. Ferme la socket	1. Crée une socket 2. Associe une adresse socket (adresse Internet et numéro de port) au service: « binding » 3. Se met à l'écoute des connexions entrantes 4. Pour chaque connexion entrante : 1) Accepte la connexion (un nouveau socket est créé avec les mêmes caractéristiques que le socket d'origine, ce qui permet de lancer un thread ou un processus fils pour gérer cette connexion) 2) Lit et/ou écrit sur la nouvelle socket 1. Ferme la socket créé

Les sockets en mode connecté



Les sockets en mode connecté

Client:

socket()	Create client socket
connect()	Create a connection
send()	Send data
receive()	Blocking receive data
close()	Close client socket

Server:

serversocket()	Create server socket
bind()	Bind server socket to socket address (IP+port)
listen()	Create queues for requests
accept()	Block on incoming requests
close()	Close server socket

Les sockets en mode connecté

- **Classes Java utilisées pour communication via TCP**
  - **InetAddress** : codage des adresses IP
  - **Socket** : socket mode connecté
  - **ServerSocket** : socket d'attente de connexion du coté serveur

Les sockets en mode connecté

Classe Socket

- Socket mode connecté
- **Constructeurs**
  - public Socket(InetAddress address, int port) throws IOException : crée une socket locale et la connecte à un port distant d'une machine distante identifié par le couple address/port

  - public Socket(String address, int port) throws IOException,UnknownHostException : idem mais avec nom de la machine au lieu de son adresse IP codée et lève l'exception UnknownHostException si le service de nom ne parvient pas à identifier la machine

Les sockets en mode connecté

➢ Classe Socket

- **Méthodes d'émission/réception de données :** on récupère les flux d'entrée/sorties associés à la socket à travers :
  - **OutputStream getOutputStream()** : retourne le flux de sortie permettant d'envoyer des données via la socket
  - **InputStream getInputStream()** : retourne le flux d'entrée permettant de recevoir des données via la socket
  - Fermeture d'une socket : public close() qui sert à fermer la socket et rompt la connexion avec la machine distante

Les sockets en mode connecté

- **Classe Socket**
  - **Méthodes « get »**
    - **int getPort()** : renvoie le port distant avec lequel est connecté la socket
    - **InetAddress getAddress()** : renvoie l'adresse IP de la machine distante
    - **int getLocalPort()** : renvoie le port local sur lequel est liée la socket

Les sockets en mode connecté

Classe ServerSocket

- Socket d'attente de connexion, coté serveur uniquement
- Constructeurs
  - public ServerSocket(int port) throws IOException :
    - Crée une socket d'écoute (d'attente de connexion de la part de clients)
    - La socket est liée au port dont le numéro est passé en paramètre
    - L'exception est levée notamment si ce port est déjà lié à une socket

Les sockets en mode connecté

Classe ServerSocket

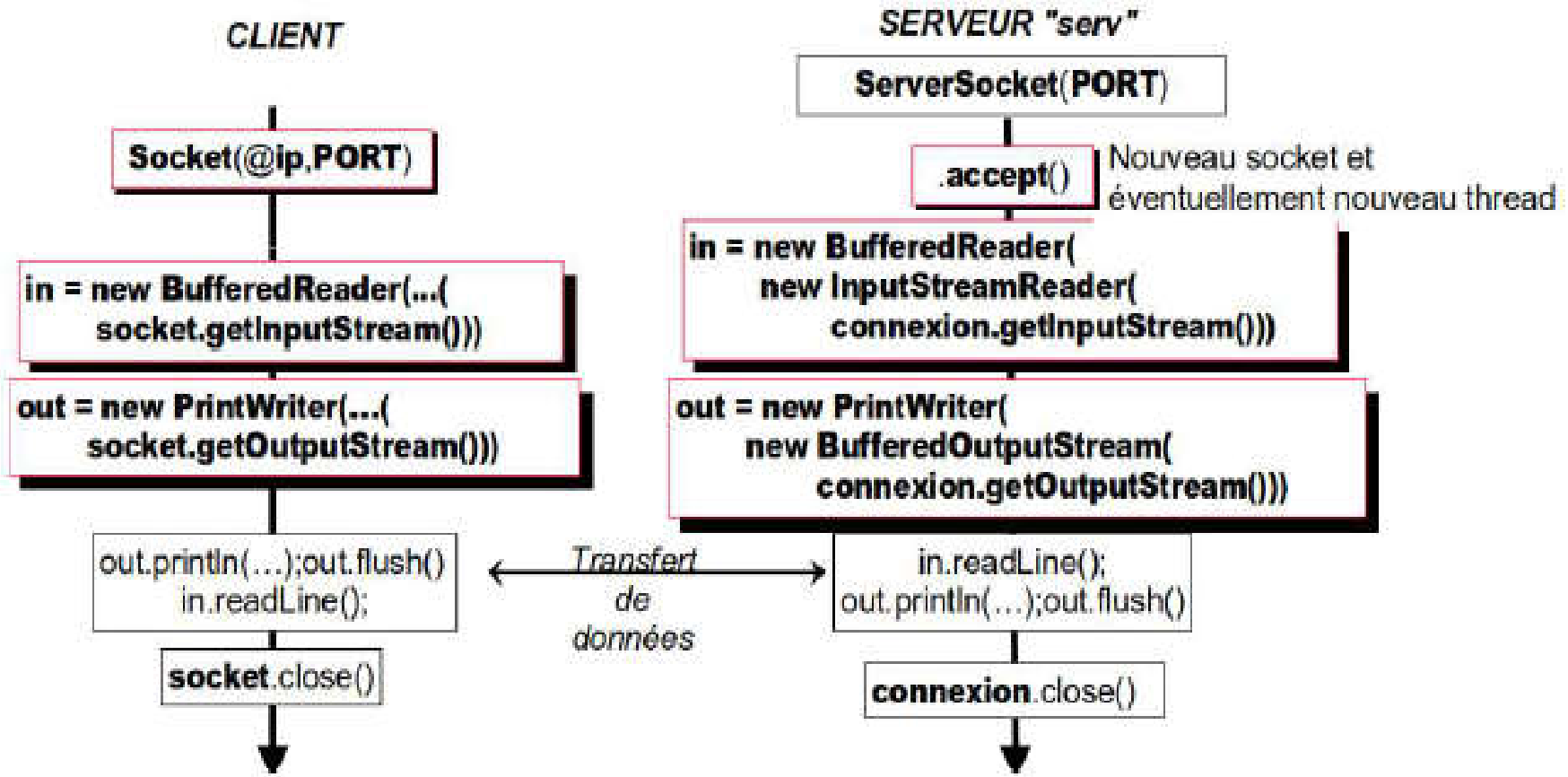
Méthodes

- Socket accept() throws IOException
  - Attente de connexion d'un client distant
  - Quand connexion est faite, retourne une socket permettant de communiquer avec le client : socket de service

- void setSoTimeout(int timeout) throws SocketException
  - Positionne le temps maximum d'attente de connexion sur un accept
  - Si temps écoulé, l'accept lève l'exception SocketTimeoutException
  - Par défaut, attente infinie sur l'accept

Les sockets en mode connecté

➢ Exemple d'utilisation de socket java en mode connecté



Les sockets en mode non connecté (Mode datagramme)

Client (émetteur)	Serveur (récepteur)
1. Crée un socket 2. Associe une adresse socket au service (adresse IP et numéro de port du service): « binding » (opération qui n'est nécessaire que si le processus va recevoir des données) 3. Lit et/ou écrit (envoi et réception des messages) sur le socket.	1. Crée un socket 2. Associe une adresse socket au service : « binding » (opération nécessaire) 3. Lit et/ ou écrit (envoi et réception des messages) sur le socket .

Les sockets en mode non connecté (Mode datagramme)

➢ Le client et le serveur ont les mêmes fonctions pour transférer les données via les sockets

socket()	create client / server socket
bind()	bind client / server to socket address (IP+port)
send()	send data (client and server)
receive()	receive data (client and server)
close()	close client / server socket

Les sockets en mode non connecté (Mode datagramme)

- **Classes Java utilisées pour communication via UDP**
  - **InetAddress** : codage des adresses IP
  - **DatagramPacket** : paquet de données envoyé via une socket sans connexion (UDP)
  - **DatagramSocket** : socket mode non connecté (UDP)

Les sockets en mode non connecté (Mode datagramme)

Classe DatagramPacket

- Structure des données en mode datagramme
- Constructeurs : public DatagramPacket(byte[] buf, int length)
  - Création d'un paquet pour recevoir des données (sous forme d'un tableau d'octets)
  - Les données reçues seront placées dans buf
  - length précise la taille max de données à lire : ne pas préciser une taille plus grande que celle du tableau. En général, length = taille de buf
  - Variante du constructeur : avec un offset pour ne pas commencer au début du tableau

Les sockets en mode non connecté (Mode datagramme)

Classe DatagramPacket

- public DatagramPacket(byte[] buf, int length, InetAddress address, int port)
  - Création d'un paquet pour envoyer des données (sous forme d'un tableau d'octets)
  - buf : contient les données à envoyer
  - length : longueur des données à envoyer : ne pas préciser une taille supérieure à celle de buf
  - address : adresse IP de la machine destinataire des données
  - port : numéro de port distant (sur la machine destinataire) où envoyer les données

Les sockets en mode non connecté (Mode datagramme)

- **Classe DatagramPacket**
  - **Méthodes « get »**
    - **InetAddress getAddress()** : si paquet à envoyer : adresse de la machine destinataire et si le paquet reçu : adresse de la machine qui a envoyé le paquet
    - **int getPort()** : si le paquet à envoyer : port destinataire sur la machine distante et si le paquet reçu : port utilisé par le programme distant pour envoyer le paquet
    - **byte[] getData** : données contenues dans le paquet
    - **int getLength()** : si le paquet à envoyer : longueur des données à envoyer et si le paquet reçu : longueur des données reçues

Les sockets en mode non connecté (Mode datagramme)

- **Classe DatagramPacket**
  - **Méthodes « set »**
    - **void setAddress(InetAdress adr)** : positionne l'adresse IP de la machine destinataire du paquet
    - **void setPort(int port)** : positionne le port destinataire du paquet pour la machine distante
    - **void setData(byte[] data)** : positionne les données à envoyer
    - **int setLength(int length)** : positionne la longueur des données à envoyer

Les sockets en mode non connecté (Mode datagramme)

Classe DatagramSocket

- Socket en mode datagramme
- **Constructeurs**
  - public DatagramSocket() throws SocketException
    - Crée une nouvelle socket en la liant à un port quelconque libre
    - Exception levée en cas de problème
  - public DatagramSocket(int port) throws SocketException
    - Crée une nouvelle socket en la liant au port local précisé par le paramètre port
    - Exception levée en cas de problème : notamment quand le port est déjà occupé

Les sockets en mode non connecté (Mode datagramme)

- **Classe DatagramSocket**
  - **Méthodes d'émission/réception de paquet**
    - public void send(DatagramPacket p) throws IOException
      - Envoie le paquet passé en paramètre. Le destinataire est identifié par le couple @IP/port précisé dans le paquet
      - Exception levée en cas de problème d'entrée/sortie
    - public void receive(DatagramPacket p) throws IOException
      - Reçoit un paquet de données
      - Bloquant tant qu'un paquet n'est pas reçu
      - Quand paquet arrive, les attributs de p sont modifiés

Les sockets en mode non connecté (Mode datagramme)

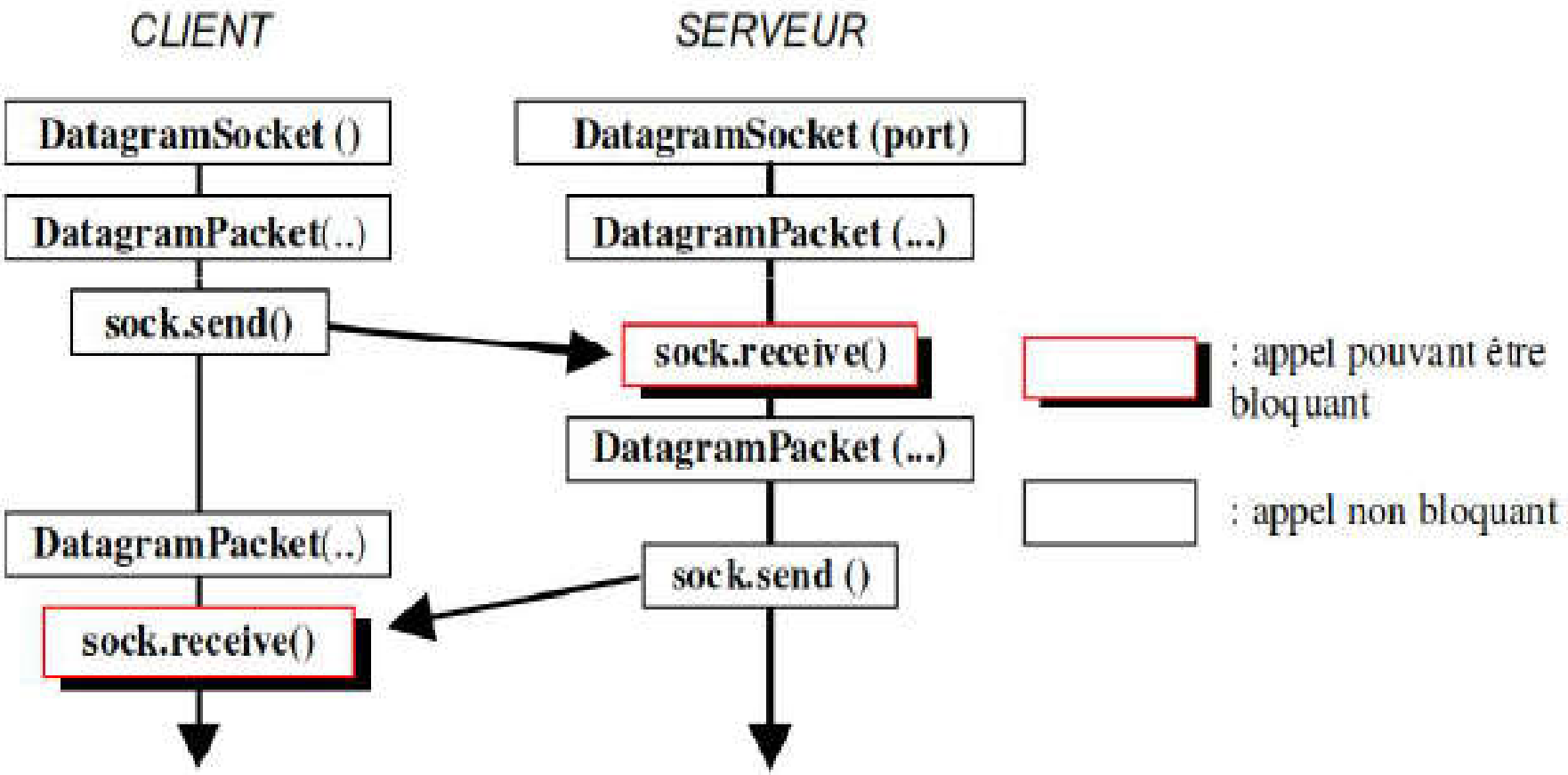
Classe DatagramSocket

Autres méthodes

- public void close() : ferme la socket et libère le port à laquelle elle était liée
- public int getLocalPort() : retourne le port local sur lequel est liée la socket

Les sockets en mode non connecté (Mode datagramme)

- **Exemple d'utilisation de socket java en mode non connecté**



Les sockets en mode non connecté (Mode datagramme)

➢ Critique sockets UDP

Avantages

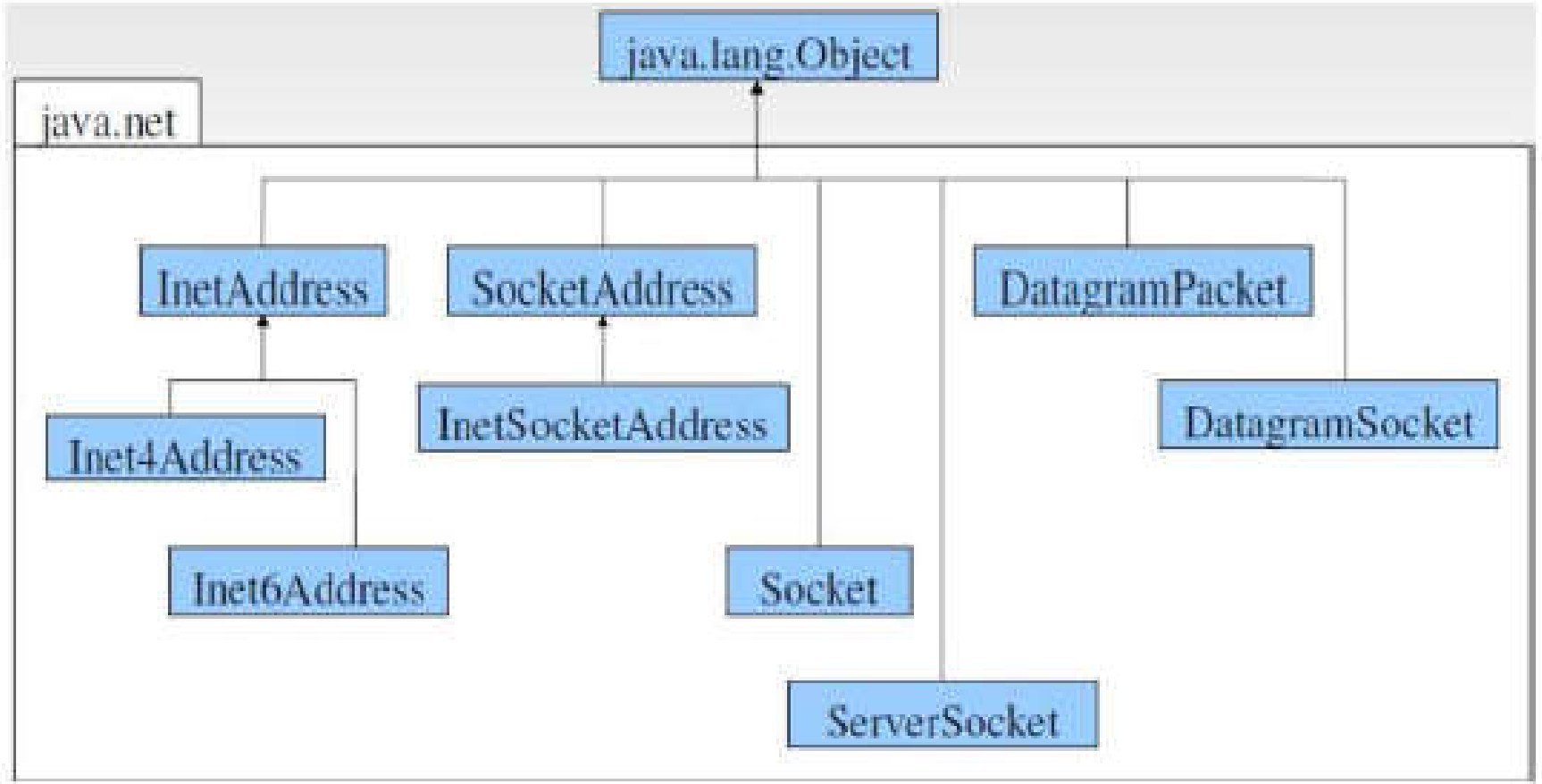
- Simple à programmer (et à appréhender)

Inconvénients

- Pas fiable
- Ne permet d'envoyer que des tableaux de byte

Implémentation des sockets en Java

- **Paquetage java.net**



Adressage IP en JAVA

➢ L'adresse IP est fournie par le biais d'objets de la classe **InetAddress** dont on retiendra les méthodes suivantes :

- **getAddress()** : retourne l'adresse IP sous forme de tableau de « byte »
- **getLocalhost()** : retourne un objet InetAddress contenant l'adresse IP de l'hôte (pas 127.0.0.1)
- **getHostName()** : retourne une chaîne de caractère : l'adresse Symbolique de l'objet
- **InetAddress.getByName**(chaîne) : permet de récupérer l'adresse IP sous forme d'objet InetAddress correspondant au nom symbolique donné en paramètre
- La méthode .toString permet d'afficher un objet InetAddress sous la forme nom symbolique/adresse IP

Adressage IP en JAVA

- Classe **InetAddress**: représente les adresses IP
  - Encapsule l'accès au serveur de noms DNS
- **Exemple de méthodes utilisées**
  - public static InetAddress getLocalHost() throws UnknownHostException
  - public static InetAddress getByName(String host) throws UnknownHostException
  - public static InetAddress[] getAllByName(String host) throws UnknownHostException