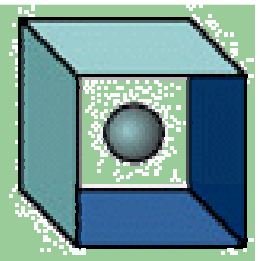
Sockets
Complément de cours



GPA-785

Communication client / serveur à travers des sockets

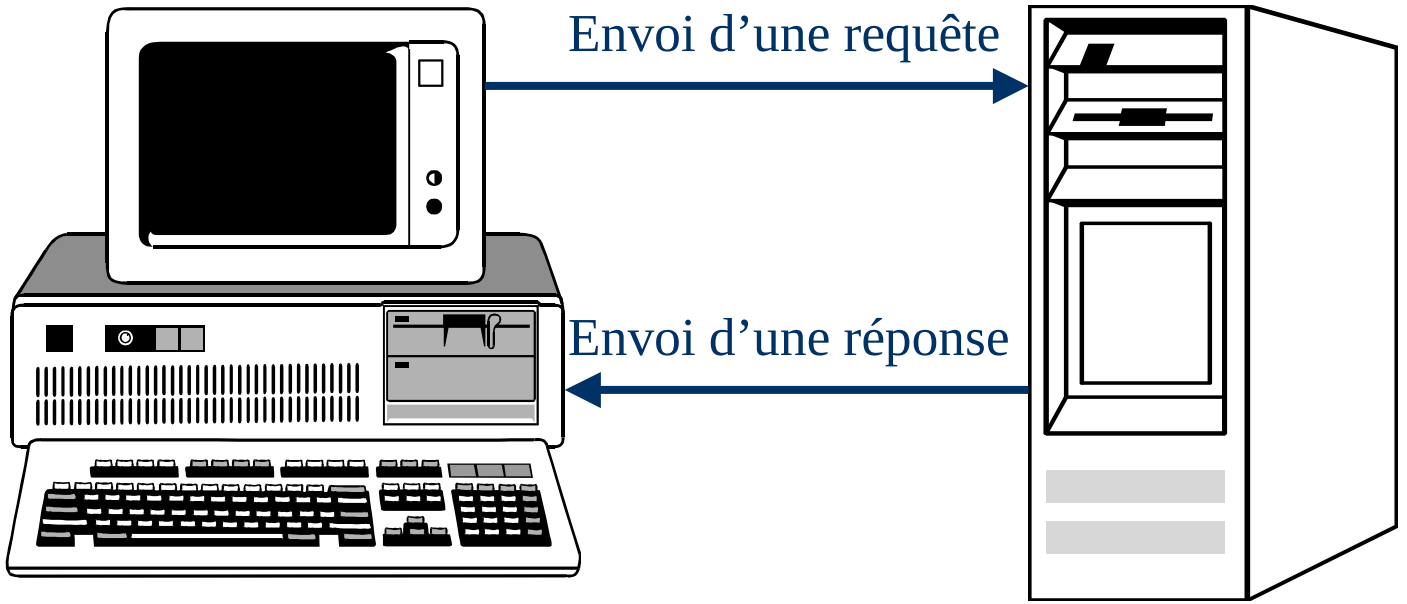
- Introduction
- Les couches du modèle TCP / IP
- Encapsulation et décapsulation de données à travers les couches du modèle TCP / IP
- Schéma d'adressage dans le modèle TCP / IP
- Programmation des sockets dans l'environnement Windows
- Les sockets en mode TCP (orienté connection)
- Les sockets en mode UDP (sans connection)

Introduction

- Dans un modèle client / serveur une application se divise en deux parties :
  - **serveur** : un programme qui s'exécute sur un réseau d'ordinateurs et qui est capable d 'offrir un service. Le serveur reçoit une requête à travers le réseau, effectue le traitement nécessaire et retourne le résultat de la requête.
  - **client** : un programme qui envoi la requête au serveur et attend la réponse.

➡ Nécessité d 'une communication inter-processus (IPC)

Exemple de communication client / serveur



Client

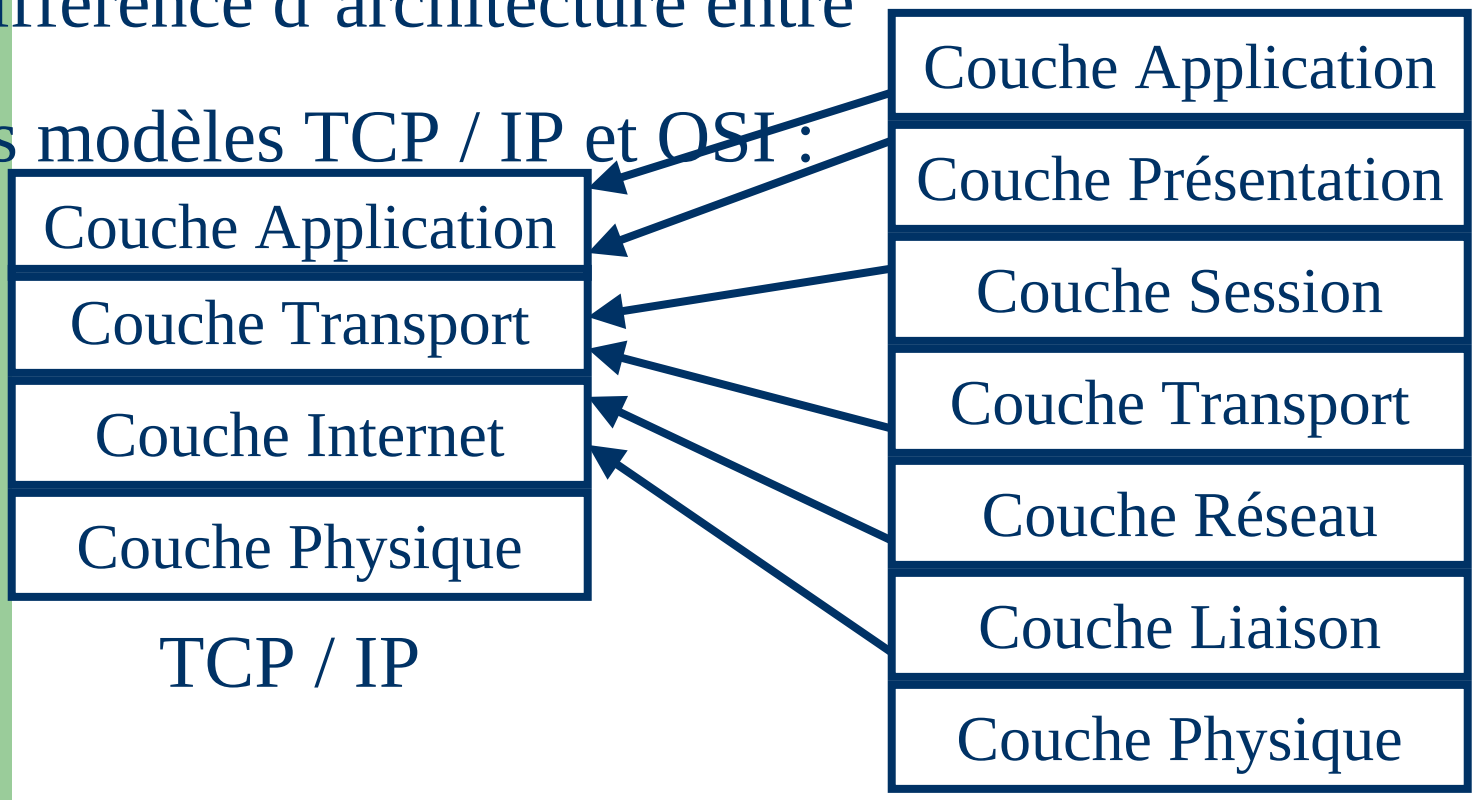
Serveur de base de données

Définition d'un socket

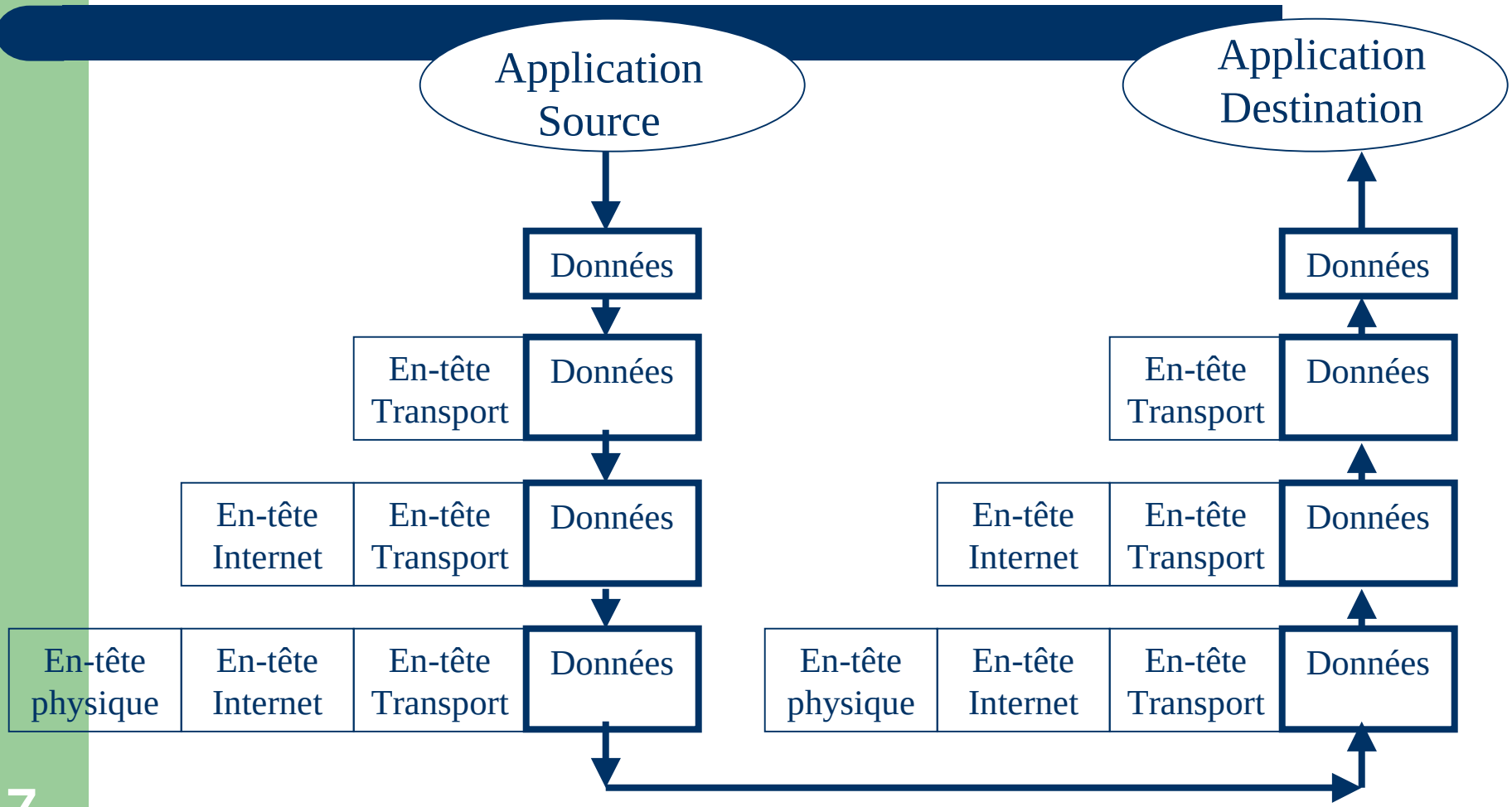
- Un socket est une structure de données abstraite qui est utilisée pour établir un canal de communication permettant l'envoi et la réception d'informations entre des processus qui s'exécutent dans un environnement distribué.
- L'interface **Berkeley Socket** a été introduite au début des années 80 et permet une communication inter-processus dans l'environnement Unix.
- L'interface **Winsock** a été développé récemment et permet une communication inter-processus dans l'environnement Windows TCP / IP.

Les couches du modèle TCP / IP

Différence d'architecture entre les modèles TCP / IP et OSI :



Encapsulation et décapsulation de données dans TCP / IP



Adresses réseau

- Chaque ordinateur connecté à un réseau possède au moins deux adresses uniques :
  - Une adresse *ethernet* de 48 bits assignée par le manufacturier. Elle est écrite en notation hexadécimale et les six groupes de 8 bits sont séparés par un `:`.
  - Une adresse Internet (IP) de 32 bits assignée par le centre d'information réseau qui gère l'abonnement. Elle est écrite en notation décimale et les quatre groupes de 8 bits sont séparés par un `.`.

Schéma d'adressage dans TCP / IP

- Adressage universel : chaque ordinateur connecté à un réseau TCP / IP possède une adresse qui l'identifie de façon unique.
- Chaque ordinateur possède une adresse Internet (IP) composée de 32 bits.
- L'adresse IP contient assez d'information pour identifier de façon unique un réseau donné et un ordinateur spécifique sur le réseau.

Les différentes classes de réseaux

- Il existe 5 classes de réseau dont trois (A, B, C) sont pour un usage général et les deux autres (D, E) sont réservées pour un usage spécial.

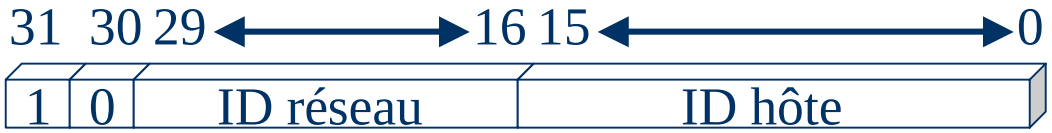
Format d'une adresse IP

Format des classes A, B et C

- Classe A :

➡ 126 réseaux et 16 777 214 hôtes par réseau

- Classe B :



➡ 16 382 réseaux et 65 534 hôtes par réseau

- Classe C :

➡ 2 097 150 réseaux et 254 hôtes par réseau

Association d'un nom de domaine à une adresse IP

- Un nom de domaine est organisé sous forme d'une structure hiérarchique :
  - Un code d'identification de l'organisation (com pour commercial, edu pour education, gov pour gouvernement, etc…).
  - Un code d'identification du pays (fi pour Finlande, ca pour Canada, us pour les États Unis, etc…).
  - Un code d'identification d'un sous domaine (polymtl).
  - Le nom de la machine hôte.

Association d'un numéro de port à un processus

- Pour effectuer une communication inter-processus il faut identifier de façon unique les processus. Un numéro de port de 16 bits est associé à chaque processus.
- La couche Internet utilise l'adresse IP pour transférer des données d'un ordinateur à un autre.
- La couche Transport utilise le numéro de port pour livrer les données au processus approprié sur l'ordinateur de destination.

Les principaux protocoles de la couche transport

- Protocole UDP (User Datagram Protocol)
  - L'entête est minimale.
  - La livraison des données n'est pas fiable.
  - Pas besoin d'établir une connexion.
- Protocole TCP (Transmission Control Protocol)
  - Le total de contrôle (checksum) est calculé et inclu dans chaque paquet.
  - La livraison des données est fiable.
  - Établir la connexion avant la transmission de données.

Format d'un message UDP

Port source	Port destination
Taille	Checksum
Données	

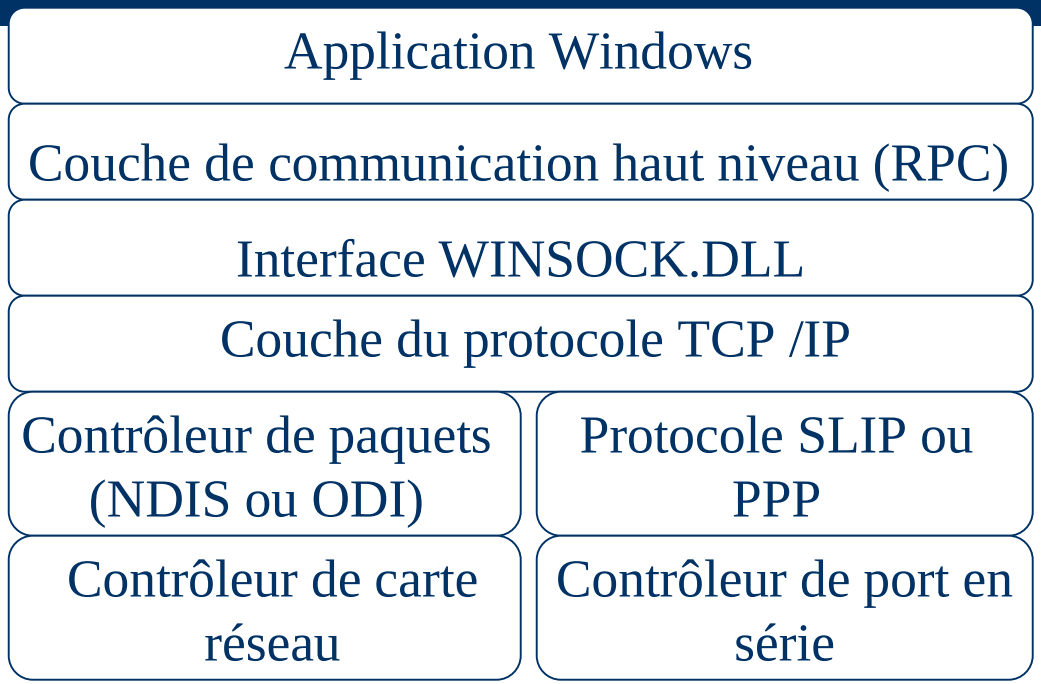
- Le champ Checksum est optionnel. Il est mis à zéro lorsqu'il n'est pas utilisé. Si le total de contrôle n'est pas effectué par le protocole UDP aucun contrôle n'est effectué sur les données puisque le protocole IP effectue le checksum sur l'entête IP et non sur le champ de données

Format d'un message TCP

Port source			Port destination
Numéro de séquence			
Numéro d'accusé de réception			
Offset	Reserved	Flags	Fenêtre
Checksum			Pointeur urgent
Options			Padding
Données			

Les champs supplémentaires (voir Stallings pp. 611) permettent d'assurer la fiabilité de la transmission des données

Programmation des sockets sous l'environnement Windows

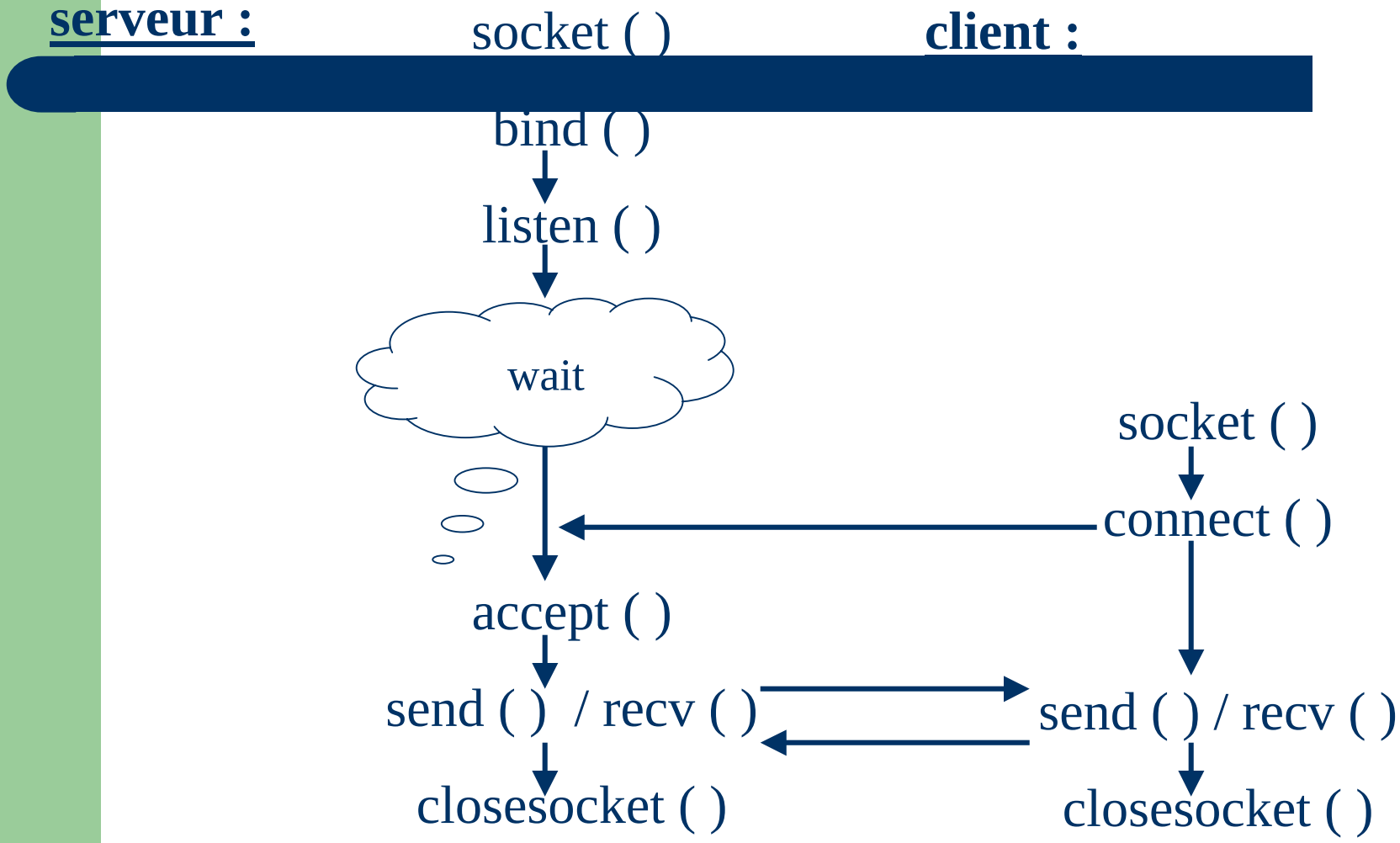


Architecture des couches dans Winsock

Initialisation et fermeture d'une communication sous Windows

- Fichiers nécessaires :
  - winsock.lib
  - winsock.h
  - winsock.dll
- **WSAStartup ( )** permet d'initialiser winsock.dll et de confirmer que la version winsock.dll est compatible avec l'application considérée.
- **WSACleanup ( )** est utilisée pour mettre fin à l'utilisation de Winsock par une application donnée.

Communication client / serveur en mode TCP



Les fonctions principales

- socket ( ) permet de créer une socket pour établir un cannal de communication. Ses arguments spécifient :
  - **af** : la famille d'adresses qui est AF_INET pour Winsock 1.1.
  - **type** : le type de socket qui est SOCK_STREAM en mode TCP et SOCK_DGRAM en mode UDP.
  - **protocole :** la valeur de protocole est généralement mise à zéro pour considérer le protocole par défaut correspondant au type spécifié.

Les fonctions principales (suite)

- bind ( ) permet d'associer un nom au descripteur de socket retourné par la fonction socket ( ). Ses arguments sont :
  - le descripteur **s** de la socket considérée.
  - Un pointeur **addr** vers l'adresse ou le nom assigné.
  - La taille **namelen** de la structure vers laquelle pointe **addr**.
- Des fonctions de conversion d'adresses sont utilisées pour passer d 'une représentation binaire à une représentation ASCII et inversement.

Les fonctions principales (suite)

- Listen ( ) permet d'écouter des connections à une socket de la part des applications client. Ses arguments sont :
  - **s** : le descripteur de la socket à travers laquelle on veut écouter.
  - **backlog** : le nombre de connections que l'application serveur peut gérer en file d'attente avant de les traiter. La valeur de **backlog** est entre 1 et 5. La valeur de cet argument permet de prévenir la couche WinSock du nombre de ressources qu'elle peut allouer.

Les fonctions principales (suite)

- accept ( ) permet d'accepter la connection d'un client à un serveur. Ses arguments sont :
  - **s** : le descripteur de la socket sur laquelle on accepte la connection demandée.
  - **addr** : est un pointeur vers la structure qui accepte l'adresse du client.
  - **addrlen** : est la taille de la structure vers laquelle pointe **addr**.
- La fonction accept ( ) retourne le descripteur de la socket utilisée pour communiquer avec le client car la socket d'origine doit rester libre pour attendre d'autres connections.

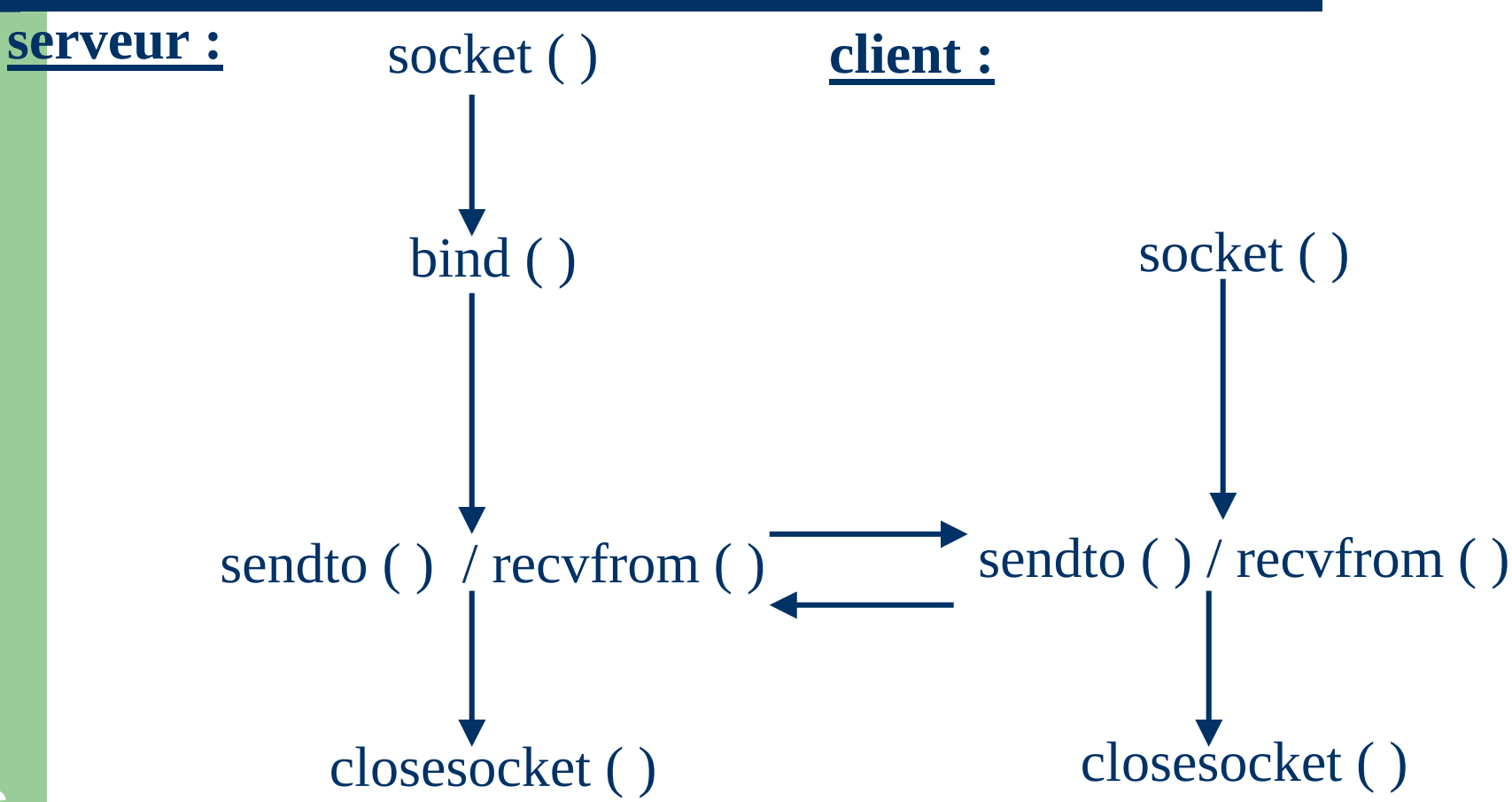
Les fonctions principales (suite)

- connect ( ) permet à un client de se connecter à un serveur. Ses arguments sont :
  - **s** : le descripteur de la socket à utiliser pour établir la connexion.
  - **name** : l'adresse du serveur sur lequel le client veut se connecter.
  - **namelen** : la longueur de **name**.
- Contrairement à une socket serveur Winsock peut assigner n'importe quel port à un client. getsockname ( ) peut être utilisé pour connaître le port assigné.

Les fonctions principales (suite)

- send ( ) et recv ( ) permettent respectivement d'envoyer et de recevoir des données à travers une socket. Leurs arguments sont :
  - **s** : le descripteur de la socket à travers laquelles les données seront envoyées ou reçues.
  - **buf :** le pointeur vers un buffer qui contient les données à transmettre ou qui est destiné pour recevoir les données.
  - **len** : le nombre d'octets du buffer.
  - **flags** : pour spécifier certaines options lors de l'envoi ou de la réception.

Communication client / serveur en mode UDP



Les fonctions principales (suite)

- sendto ( ) et recvfrom ( ) permettent respectivement d'envoyer et de recevoir des datagrammes. Leurs arguments sont :
  - **s, buf, len, et flags** comme dans send ( ) et recv ( ).
  - **to** ou **from** : est un pointeur vers l'adresse internet de la socket à travers laquelle on reçoit ou on envoie.
  - **tolen ou fromlen** : est la taille de l'adresse de réception ou d'envoi.
- closesocket ( ) permet de libérer la socket utilisée à la fin de la communication.