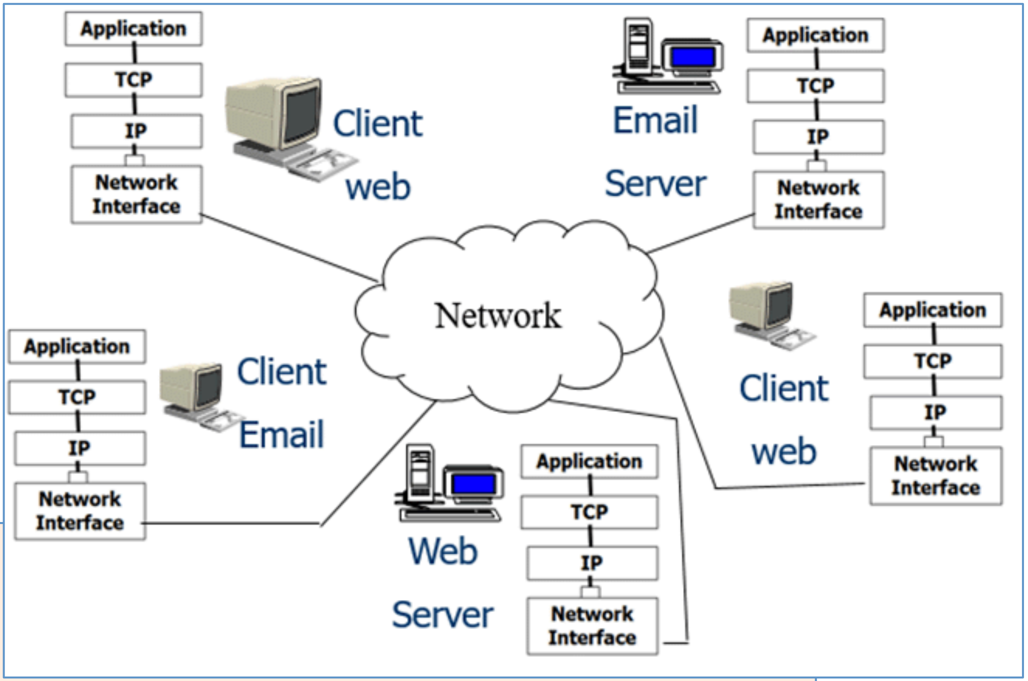
Université Paris 8
2, rue de la Liberté
93526 Saint-Denis cedex

Cours
Programmation concurrente

Interface de communication réseau type SOCKET

Département Informatique et Technologies du Numérique
Master 1 Informatique
Parcours : Informatique
Y. Touati
capaok@gmail.com

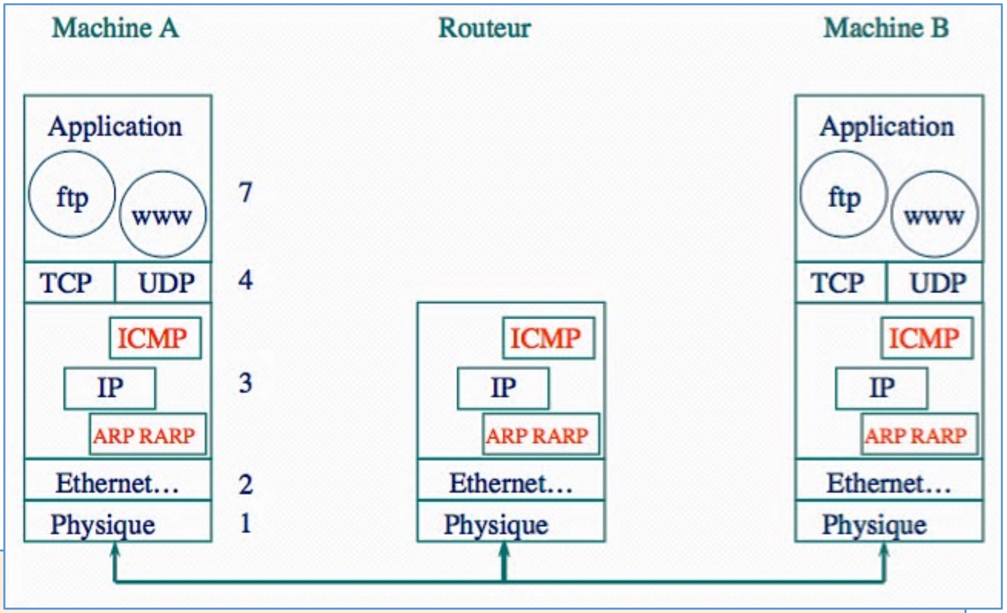
Différents services dans les réseaux



– Plusieurs services :

- WWW pour le WEB.
- FTP pour le transfert de fichier.
- SMTP pour le courrier électronique.
- TELNET / SSH pour l’accès sur des nœuds distants.

Pile protocolaire et services réseau



- Objectif :
  - Assurer une interopérabilité au niveau de la couche application.
  - Appel et utilisation d’applications du réseau internet.
  - Indépendamment des technologies et des architectures réseaux.

Couche Transport

- Deux types de protocoles :
  - **TCP** : Transport Control Protocol

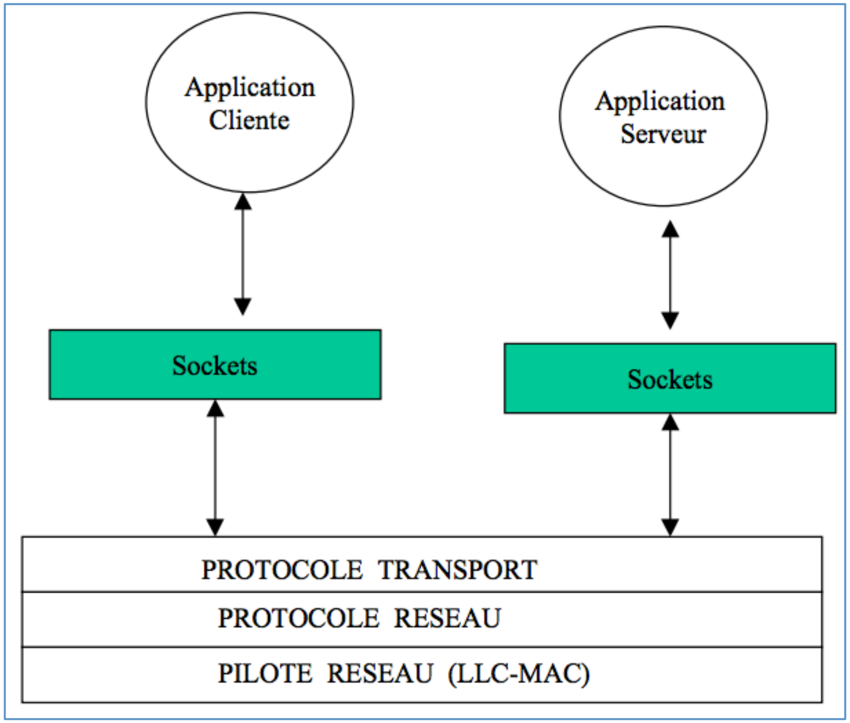
    Mode de communication connecté, très fiable
  - **UDP** : User Datagram Protocol

    Mode communication non connecté, peu fiable
- Port unique pour l'identification d'une application informatique qu'elle soit locale ou distante (codé sur 2 octets)
- Utilisation des <u>sockets</u>

Introduction aux sockets

Établissement de lien de communication entre nœuds distants

- Un PORT de communication identifié par un NUMERO
- Localiser l'ADRESSE INTERNET du nœud distant pour assurer les échanges d'informations

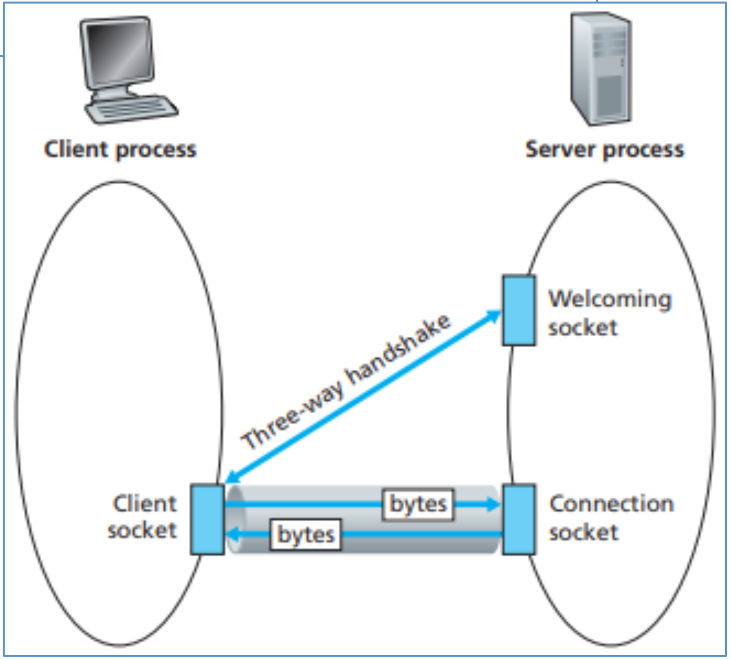


- Interface logicielle et de programmation de protocoles avec les couches réseau API
- Point de transition de l'information :
  lecture/écriture – Envoi/réception
- Berkeley 4.2 Version pour Unix : Inclusion du protocole TCP/IP dans l'OS

Les sockets

- Définition :
  - Couple de valeurs (@ IP, numéro de port)

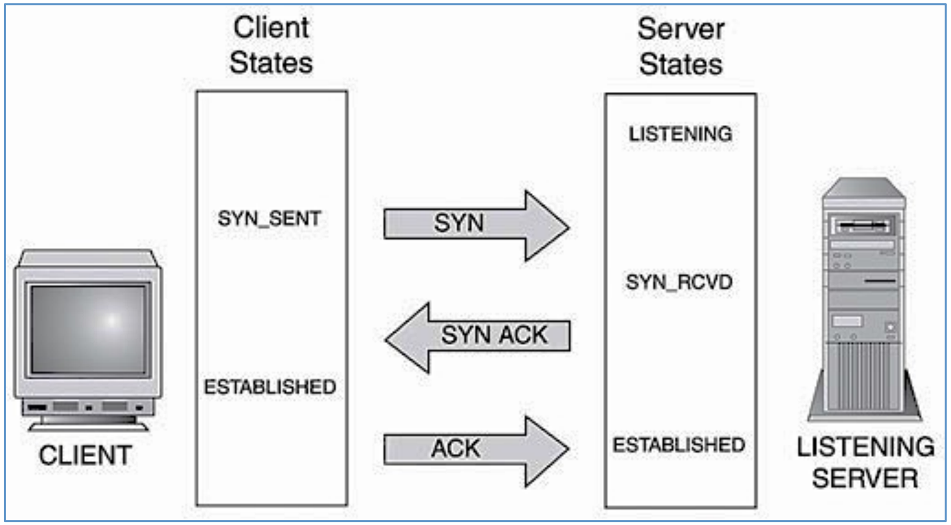
    Exemple d'application **serveur Telnet** sur la machine distante 192.1280.20.9 : (192.1280.20.9, 23)
- L'utilisation d'un couple de *sockets* permet une identification complète des échanges de données entre 2 applications distantes.



Côté serveur : 2 types de socket
Côté client : 1 seule

Etats TCP : 3-Way Handshake

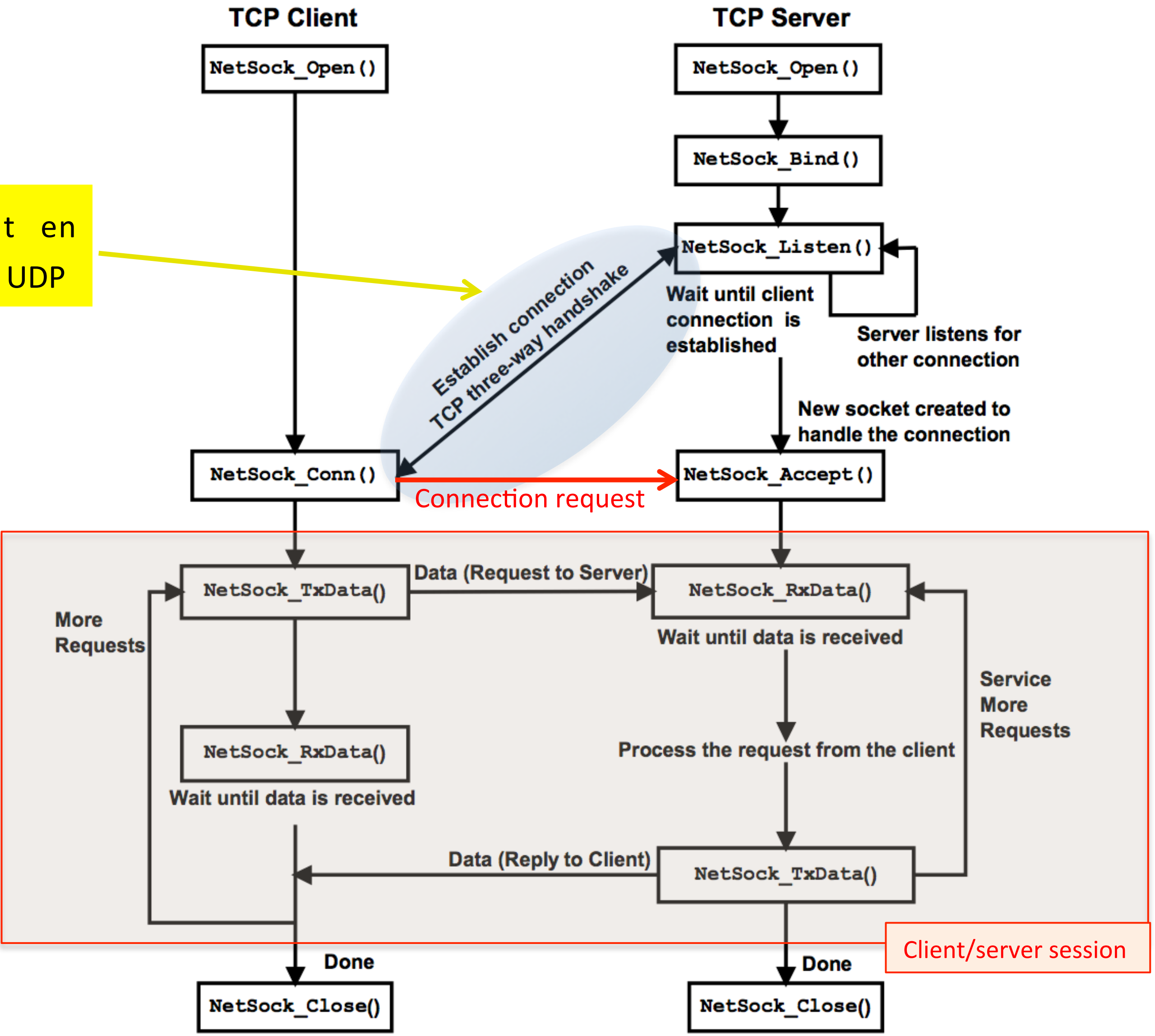
- The initiating computer sends the Connection request, sending a SYN.
- The responding computer grants the request, replying with a SYN-ACK.
- The initiating computer sends an acknowledgment, replying with an ACK.



At that point the connection is established, and data begins to flow. In contrast, a UDP packet is not guaranteed, and is just sent in the hopes it gets there.

Etablissement des sockets Client-Serveur

Echange inexistant en mode non-connecté UDP



TCP Client

NetSock_Open()

NetSock_Conn()

Establish connection TCP three-way handshake

Connection request

NetSock_TxData()

Data (Request to Server)

More Requests

NetSock_RxData()

Wait until data is received

Data (Reply to Client)

Done

NetSock_Close()

TCP Server

NetSock_Open()

NetSock_Bind()

NetSock_Listen()

Wait until client connection is established

Server listens for other connection

New socket created to handle the connection

NetSock_Accept()

NetSock_RxData()

Wait until data is received

Process the request from the client

Service More Requests

NetSock_TxData()

Done

NetSock_Close()

Client/server session

Mise en œuvre des sockets

TCP/IP
Value : 0

Création d'un point de communication

```
int socket(int domain, int type, int protocol);
```

Protocoles locaux à Unix

Internet Protocol
TCP, UDP, ..

Renvoi un descripteur

1. **Domaine** de communication : Unix (AF_UNIX), TCP/IP (AF_INET), ….
2. **Type** de protocole : Fixer la sémantique des communications
    1. **SOCK-DGRAM** : Échange de message sous forme de datagrammes (exemple d'UDP dans le domaine AF_INET).
    2. **SOCK_STREAM** : Envoi de flux d'octets (exemple de TCP dans le domaine AF_INET).

Mécanisme de mise en œuvre côté CLIENT mode TCP

Nœud CLIENT

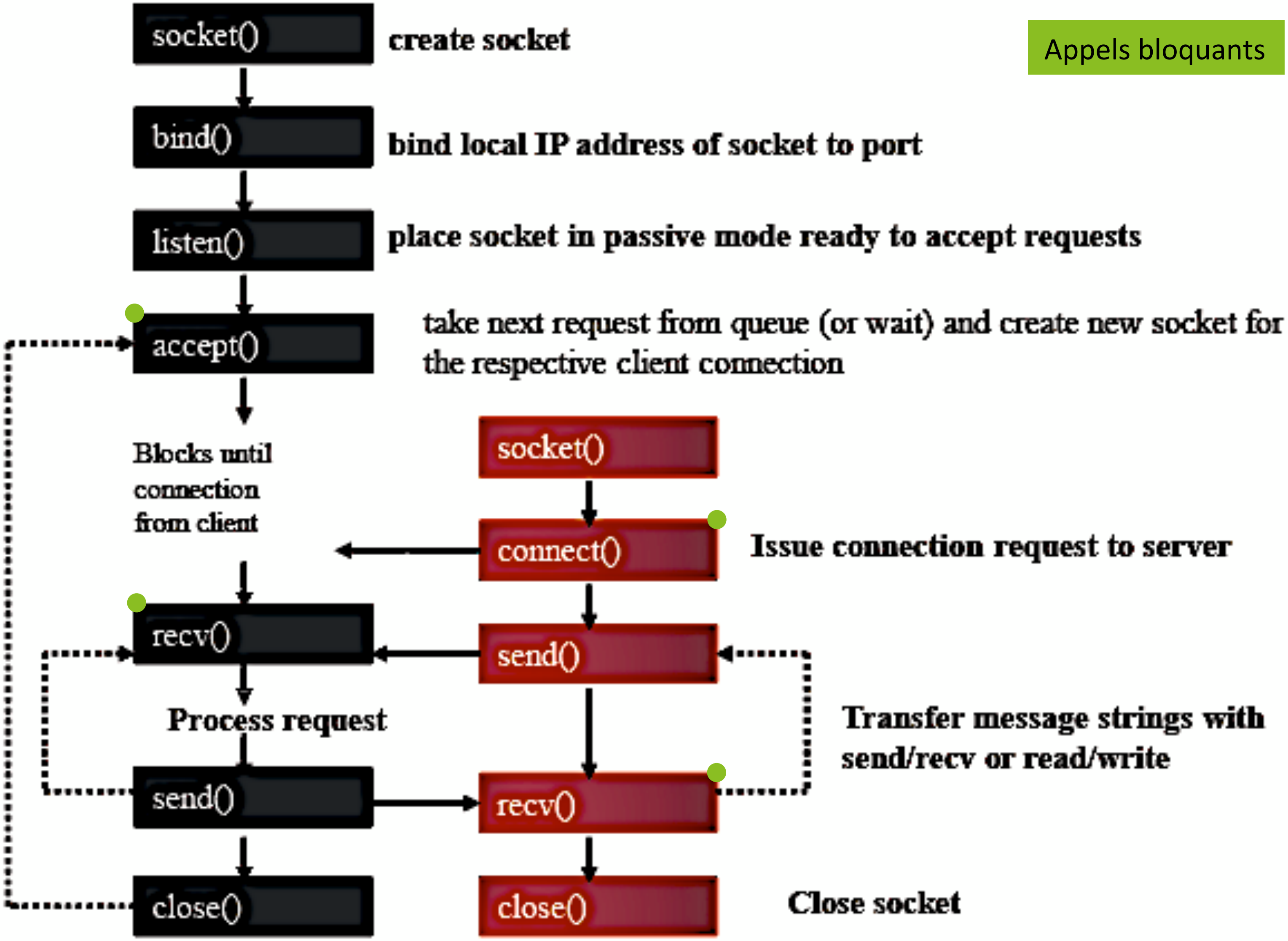
1. Création d'une socket pour la communication.
2. Configuration du protocole TCP avec une adresse IP SERVER et un numéro de port du service (Attribution automatique d'un numéro de port local au client).
3. Connexion au server via la socket.
4. Attente de l'acquittement du SERVER.
5. Pour chaque connexion établie :
    1. Lecture et/ou écriture via la nouvelle socket.
    2. Arrêt de la communication Client/server et fermeture de la socket.

Mécanisme de mise en œuvre côté SERVER mode TCP

Nœud SERVER

1. Création d'une socket.
2. Association d'une adresse IP et configuration du numéro de port au service *Binding*.
3. Mise en écoute des connexions clientes.
4. Pour chaque connexion entrante :
    1. Accepter la connexion par envoi d'un acquittement.
    2. Création d'une nouvelle socket avec les mêmes caractéristiques que la socket initiale.
    3. Lecture et/ou écriture via la nouvelle socket.
    4. Arrêt de la communication Client/server et fermeture de la socket.

Appels sockets TCP bloquants et non-bloquants


socket()
create socket
Appels bloquants
bind()
bind local IP address of socket to port
listen()
place socket in passive mode ready to accept requests
accept()
take next request from queue (or wait) and create new socket for the respective client connection
Blocks until connection from client
socket()
connect()
Issue connection request to server
recv()
send()
Process request
Transfer message strings with send/recv or read/write
send()
recv()
close()
close()
Close socket

SOCKETS Tutorial

by Robert Ingalls

http://www.cs.rpi.edu/~moorthy/Courses/os98/Pgms/socket.html

Socket tutorial

Some notions

1. Most inter-process communication uses the client server model.
2. One of the two processes, the client, connects to the other process, the server, typically to make a request for information.
3. The client needs to know of the existence of and the address of the server, but the server does not need to know the address of (or even the existence of) the client prior to the connection being established.
4. The system calls for establishing a connection are somewhat different for the client and the server, but both involve the basic construct of a socket.
5. A socket is one end of an inter-process communication channel. The two processes each establish their own socket.

Socket tutorial

Establishing a socket on the client side

- Create a socket with the socket() system call.
- Connect the socket to the address of the server using the connect() system call.
- Send and receive data. There are a number of ways to do this, but the simplest is to use the read() and write() system calls.

Socket tutorial

Establishing a socket on the Server side

1. Create a socket with the socket() system call.
2. Bind the socket to an address using the bind() system call. For a server socket on the Internet, an address consists of a port number on the host machine.
3. Listen for connections with the listen() system call.
4. Accept a connection with the accept() system call. This call typically blocks until a client connects with the server.
5. Send and receive data.

Creation of a socket

The **address domain** and the **socket type** are specified

Address domain

- Unix domain (inter-processes communication with a common file system).
- Internet domain (in which two processes running on any two hosts on the Internet communicate).
- The address of a socket in the Internet domain consists of the Internet address of the host machine (every computer on the Internet has a unique 32 bit address, often referred to as its IP address).
- Each socket needs a port number on that host. Port numbers are 16 bit unsigned integers.

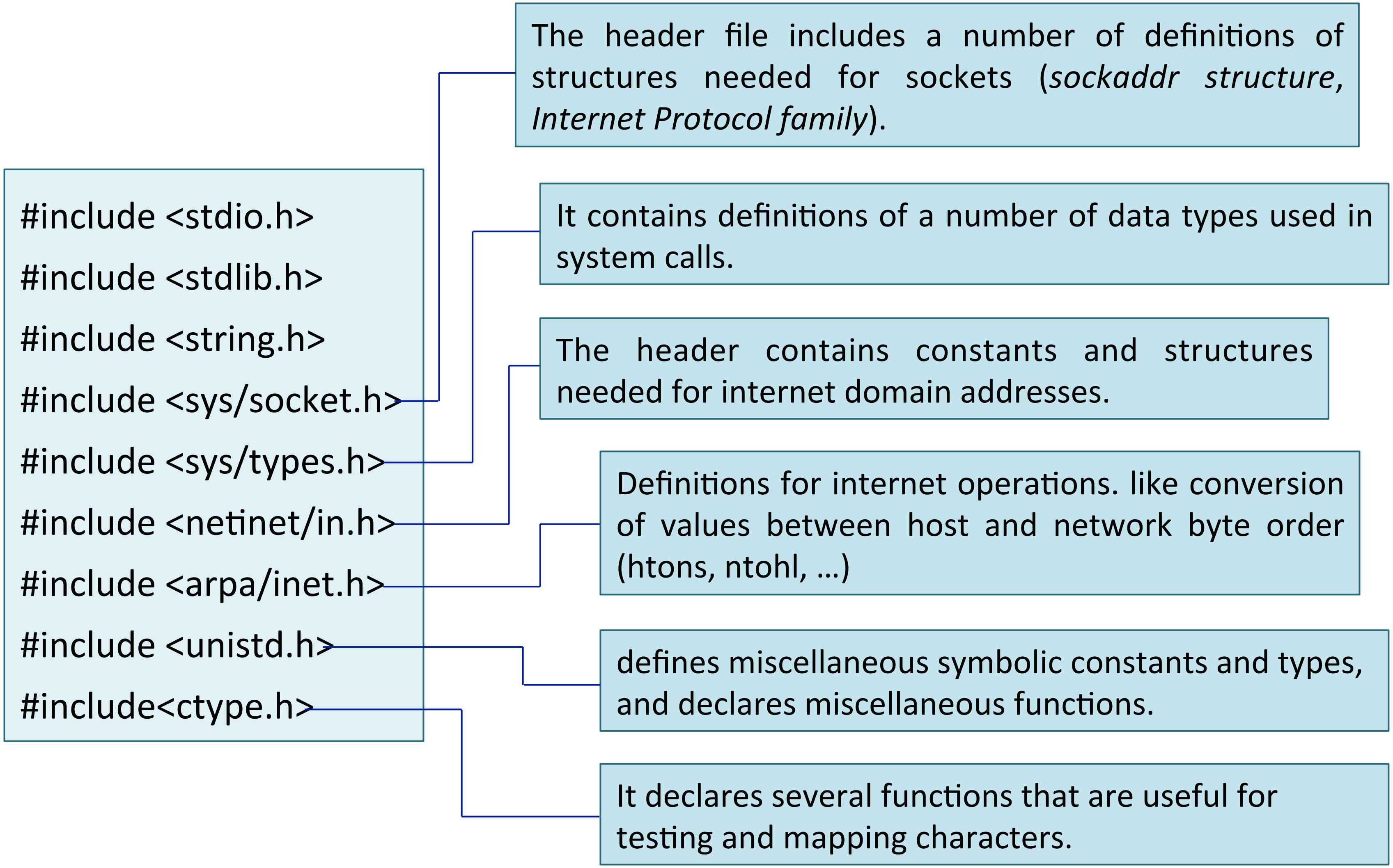
Creation of a socket

The **address domain** and the **socket type** are specified

Socket type

- Stream sockets (communications consider stream of characters and use TCP, which is a reliable, stream oriented protocol).
- Datagram sockets (read entire messages at once, use UDP (Unix Datagram Protocol), which is unreliable and message oriented).

Bibliothèque standards et packages



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <unistd.h>
#include<ctype.h>
```

The header file includes a number of definitions of structures needed for sockets (*sockaddr structure, Internet Protocol family*).

It contains definitions of a number of data types used in system calls.

The header contains constants and structures needed for internet domain addresses.

Definitions for internet operations. like conversion of values between host and network byte order (htons, ntohl, …)

defines miscellaneous symbolic constants and types, and declares miscellaneous functions.

It declares several functions that are useful for testing and mapping characters.

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Parameters ’ initialization of the socket

1

int sockfd, newsockfd; File descriptors (store the values returned by the socket system call).

int portno; Stores the port number on which the server accepts connections.

int clilen; Stores the size of the address of the client.

int n; Return value for the read()/write() calls containing the number of characters read / written.

char buffer[256]; The server reads characters from the socket connection into this buffer.

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

A structure containing an internet address (defined in <netinet/in.h>)

2

struct sockaddr_in **serv_addr, cli_addr**; The address of the **server** and the address of the **client** which connects to the server.

```
struct sockaddr_in {
        short      sin_family;
        u_short    sin_port;
        struct     in_addr sin_addr;
        char       sin_zero[8]
    };
```

serv_addr.sin_family = AF_INET;

AF_INET : domaine d'adresse IPV4 ou IPV6
sin_port : port de communication
INADDR_ANY: Adresse IP du Host (server)

serv_addr.sin_port = htons(portno); The port number is converted into a port number in network byte order.

serv_addr.sin_addr.s_addr = INADDR_ANY; Type of structure containing the IP address of the host.

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Creation of the socket

3

```
sockfd = socket(AF_INET ,  SOCK_STREAM , 0);
```

- The socket() system call creates a **new socket**

  Three arguments:

  1. The address domain of the socket (Internet domain AF_INET)
  2. The type of socket which is symbolic constant (Characters are read in a continuous stream as if from a file : SOCK_STREAM)
  3. The protocol (0 for TCP or UDP)

- The socket system call returns an a small integer as an entry into the file descriptor table
- This value is used for all subsequent references to this socket

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Association de l’ @ IP et du numéro de port au service *Binding*

4

```
bind(sockfd , (struct sockaddr *) &serv_addr , sizeof(serv_addr));
```

- It binds a socket to an address (the @ of the current host and port number on which the server will run)

Three arguments:

1. The socket file descriptor.
2. The address to which is bound (pointer to a structure of type **sockaddr**)

   (what is passed in is a structure of type sockaddr_in, and so this must be cast to the correct type).
3. The size of the address to which it is bound.

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Ecoute du serveur sur son socket sur 5 connexions par exemple

5

```
listen(sockfd , 5);
```

- This system call allows the process to listen on the socket for connections.

Two arguments:

1. The socket file descriptor
2. The size of the queue (number of connections limited to 5, that can be waiting while the process is handling a particular).

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Socket de connexion Client-server

6

```
clilen = sizeof(cli_addr);

newsockfd = accept(sockfd , (struct sockaddr *) &cli_addr , &clilen);
```

- This system call causes the process to block until a client connects to the server.
- It returns a new file descriptor, and all communication on this connection should be done using the new file descriptor

    1. First file descriptor
    2. Reference pointer to the address of the client
    3. The size of this structure

Example of sockets in the Internet domain using the TCP protocol
SERVER Part

Client-server communication

7

```
bzero(buffer , 1024);
```

- Initialization the buffer using the bzero() function

```
n = read(newsockfd , buffer , 1024);
```

- Reads from the socket until the client has executed a write()
- It returns the total number of characters read

```
n = write(newsockfd , "I got your message” , 18);
```

- Once a connection has been established, both ends can both read and write to the connection

- Everything written by the client will be read by the server, and everything written by the server will be read by the client

Example of sockets in the Internet domain using the TCP protocol
CLIENT Part

Parameters ' initialization of the socket

1

int sockfd; File descriptors (store the values returned by the socket system call)

int portno; Stores the port number on which the server accepts connections

struct sockaddr_in serv_addr; The variable **serv_addr** will contain the @server to which we want to connect.

It is of type **struct sockaddr_in**

char buffer[1024]; Defines a buffer with 1024 bytes

Example of sockets in the Internet domain using the TCP protocol
CLIENT Part

Establishment of a socket

2

```
sockfd = socket(AF_INET ,  SOCK_STREAM , 0);
```

- Create a **new socket**

  Three arguments:

  1. The address domain of the socket (Internet domain AF_INET)
  2. The type of socket which is symbolic constant (Characters are read in a continuous stream as if from a file : SOCK_STREAM)
  3. The protocol (0 for TCP or UDP)

- The socket system call returns an a small integer as an entry into the file descriptor table

Sockets Tutorial

Example of sockets in the Internet domain using the TCP protocol

CLIENT Part

Connection to the server

3

```
serv_addr.sin_port = htons(portno);
```

The port number is converted into a port number in network byte order.

```
connect(sockfd , &serv_addr , sizeof(serv_addr));
```

- Function is called by the client to establish a connection to the server

  Tree arguments:

  1. The socket file descriptor
  2. The address of the host to which it wants to connect (including the port number)
  3. The size of this address

Sockets Tutorial

Example of sockets in the Internet domain using the TCP protocol

CLIENT Part

Client-server communication

4

```
printf("Please enter the message: ");       message to forward to server
bzero(buffer , 1024);                        Initialization of the buffer
fgets(buffer , 1024 , stdin);                read the message from stdin
n = write(sockfd , buffer , strlen(buffer));  write the message to the socket
bzero(buffer , 1024);                        Initialization of the buffer
n = read(sockfd , buffer , 1024);            Reads the replay from the socket
printf("%s\n",buffer);
```