Système réparti

Enseignante: Dr. Olfa Souki

Email: soukiolfa92@gmail.com

Plan du cours

- Chapitre 1: Rappel sur les sockets
- Chapitre 2: Architectures client / serveur
- Chapitre 3: Intergiciels orientés objets (CORBA)
- Chapitre 4: Intergiciels orientés messages
- Chapitre 5: : Intergiciels orientés composants
- Chapitre 6: Problèmes fondamentaux de la répartition

Plan du cours

Chapitre 1: Rappel sur les sockets

- ❑ Introduction aux sockets
- ❑ Socket en mode connecté (au dessus de TCP)
- ❑ Socket en mode paquet (au dessus de UDP)
- ❑ API Java Java.net

Plan du cours

Chapitre 2: Architectures client / serveur

- ❑ Problèmes d'intégration d'applications
- ❑ Fondement des architectures client/serveur
- ❑ Système RMI API
- ❑ Java JNDI

Plan du cours

Chapitre 3: Intergiciels orientés objets (CORBA)

- Introduction aux intergiciels
- Architecture OMA / CORBA
- Composants et services CORBA
- Langage IDL et projection en JAVA

Plan du cours

Chapitre 4: Intergiciels orientés messages

- Intergiciels orientés messages
- API Java JMS
- Création, manipulation et échange de messages
- Fiabilité de communication OM

Plan du cours

Chapitre 5: Intergiciels orientés composants

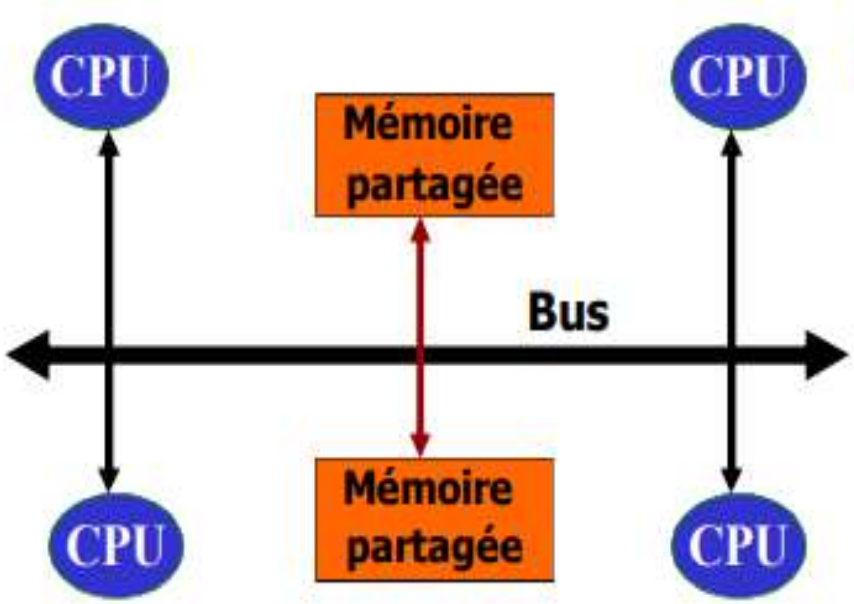
- Fondements de programmation par composants
- Modèle de composants EJB
- Modèle de composants .NET

Plan du cours

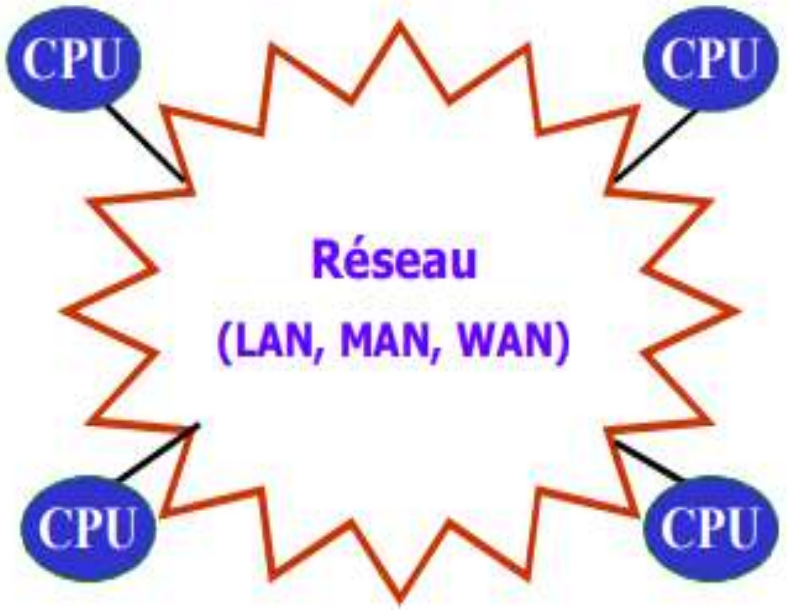
Chapitre 6: Problèmes fondamentaux de la répartition

- Concurrence
- Tolérance aux fautes
- Sécurité et contrôle d'accès Réplication, cohérence et cache
- Synchronisation
- Elasticité

Introduction



- **Systèmes fortement couplés (systèmes parallèles)**



- **Systèmes faiblement couplés (systèmes répartis)**

Système réparti

- *"Un système réparti est un ensemble de machines autonomes connectées par un réseau, et équipées d'un logiciel dédié à la coordination des activités du système ainsi qu'au partage de ses ressources." Coulouris et al. [COU 94].*

- *"Un système réparti est un système qui s'exécute sur un ensemble de machines sans mémoire partagée, mais que pourtant l'utilisateur voit comme une seule et unique machine." Tanenbaum [TAN 94].*

- Systèmes fortement et faiblement couplés.

- Notion d'**image unique du système** $\Longrightarrow$ système généralement incomplet.

Système réparti

- « Un système réparti est un ensemble de processeurs (sites/nœuds) interconnecté par un réseau dans lequel chaque processeur a sa propre mémoire et ses propres périphériques ».
- « Un OS réparti est un OS qui apparaît aux utilisateurs comme un OS centralisé, mais qui s'exécute sur plusieurs CPU indépendantes et interconnectées ».
- « Un système réparti est un système dans lequel une machine dont vous n'avez jamais entendu parler auparavant vous empêche de travailler ».

Fortement Vs Faiblement Couplés

Critère	Systèmes fortement couplés (Parallèles)	Systèmes faiblement couplés (Répartis)
Mémoire	Mémoire **partagée** entre processeurs	Chaque processeur a sa **mémoire locale**
Communication	Par **bus** → rapide	Par **réseau (LAN, MAN, WAN)** → plus lente
Dépendance	Processeurs très dépendants les uns des autres	Processeurs plus **indépendants**
Performance	Très efficace pour le **calcul parallèle**	Adapté aux **traitements distribués**
Scalabilité	Difficile d'ajouter beaucoup de processeurs (limité par le bus)	Facile d'ajouter de nouvelles machines
Exemples	Supercalculateurs, multiprocesseurs (SMP)	Clusters, Grid Computing, Cloud

Chapitre 1 :

Les sockets

Objectifs

Revoir la nature des sockets

Comprendre ce que sont les sockets, leurs caractéristiques et leur fonctionnement de base dans les échanges réseau.

Expliquer le rôle des sockets dans la communication réseau

Identifier comment les sockets facilitent l'échange de données entre machines via un réseau, en servant d'interface de communication.

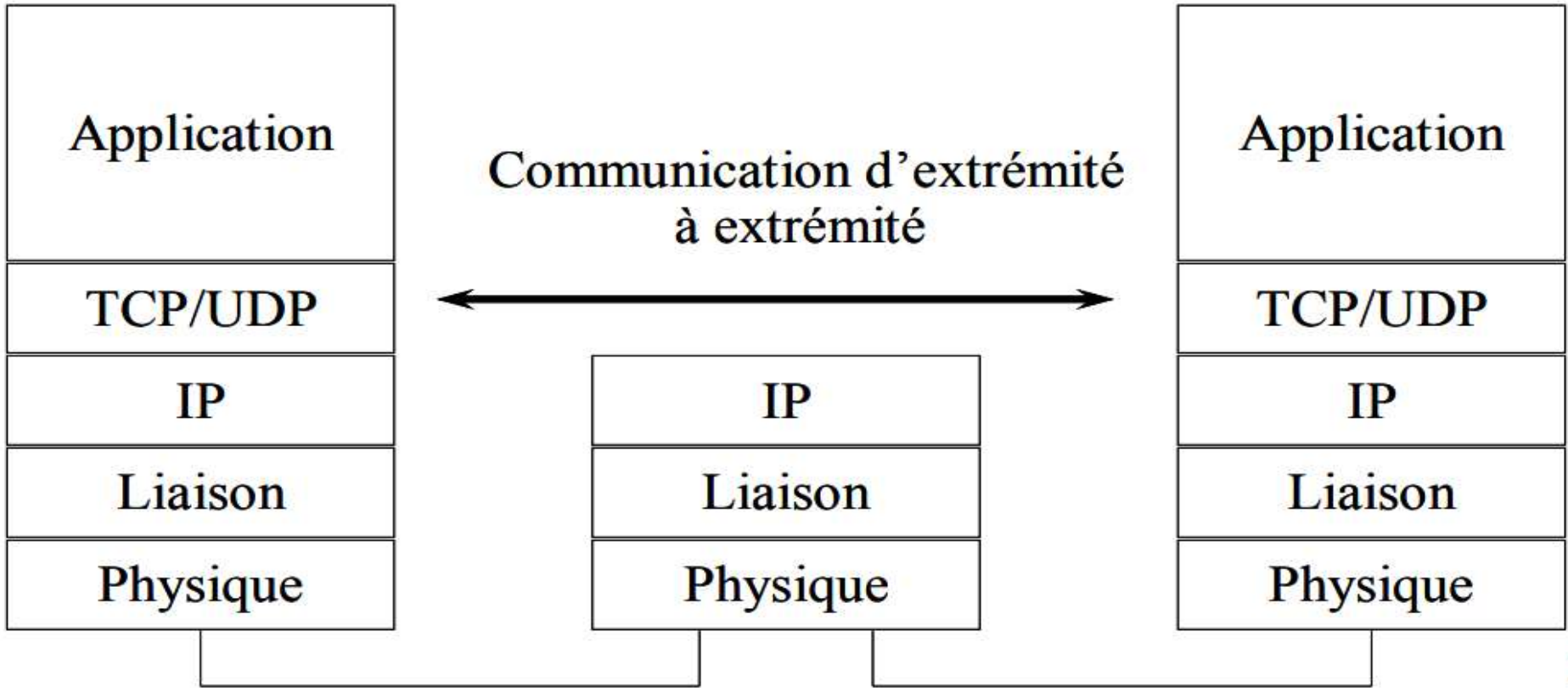
Distinguer socket et protocole réseau

Clarifier la différence entre une socket, qui est une interface de communication, et les protocoles réseau, qui définissent les règles d'échange.

Rappel

Rappel sur les réseaux

- TCP ou UDP
  - Communication entre systèmes aux extrémités
  - Pas de visibilité des systèmes intermédiaires



Définition d'une socket

Qu'est-ce qu'une Socket ?

Comprendre les composants clés d'une socket dans la communication réseau

Interface logicielle de communication

Permet à deux programmes de communiquer via un réseau en offrant un point d'interaction.

Point de terminaison réseau

Agit comme un point final dans la communication entre applications sur un réseau.

Échange de données entre applications

Permet l'envoi et la réception de données entre différents programmes.

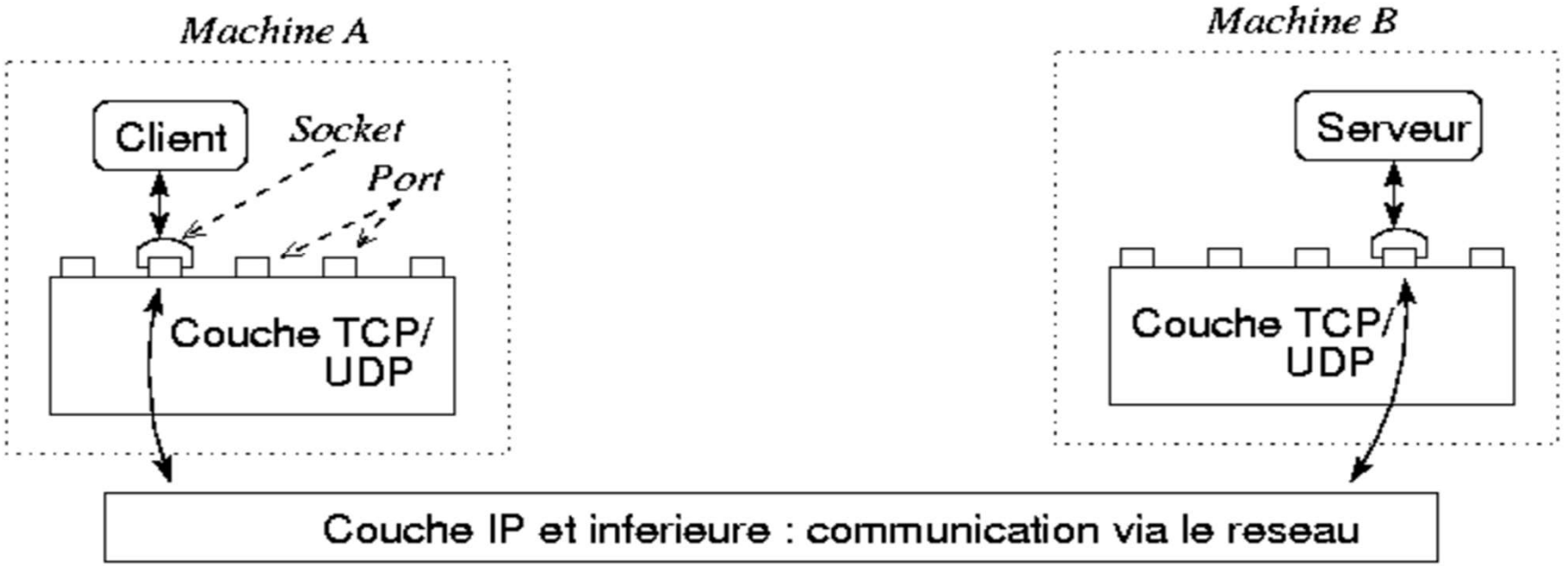
Abstraction des protocoles TCP et UDP

Masque la complexité des protocoles sous-jacents pour simplifier la communication.

Base de la communication réseau moderne

Comprendre la socket est essentiel pour maîtriser les échanges réseau actuels.

Sockets



- Une socket est
  - Un point d'accès aux couches réseau TCP/UDP
  - Liée localement à un port
    - Adressage de l'application sur le réseau : son couple @IP:port
- Elle permet la communication avec un port distant sur une machine distante : c'est-à-dire avec une application distante

Client/ Serveur avec Sockets

Client/serveur avec sockets

- Il y a toujours différenciation entre une partie client et une partie serveur
  - Deux rôles distincts au niveau de la communication via TCP/UDP
  - Mais possibilité que les éléments communiquant jouent un autre rôle ou les 2 en même temps
- Différenciation pour plusieurs raisons
  - Identification : on doit connaître précisément la localisation d'un des 2 éléments communiquants
    - Le coté serveur communique via une socket liée à un port précis : port d'écoute
  - Dissymétrie de la communication/connexion
    - Le client initie la connexion ou la communication

Socket en Mode Connecté : Base sur TCP

Comprendre les caractéristiques et usages du mode connecté TCP

01 Utiliser le mode connecté avec le protocole TCP

Le mode connecté repose sur TCP, assurant une communication fiable entre deux points.

02 Établir une connexion avant l'échange des données

La connexion est établie préalablement à la transmission, garantissant un échange structuré.

03 Garantir une transmission ordonnée et sans perte

TCP assure que les données sont reçues dans l'ordre exact et sans aucune perte durant la transmission.

04 Implémenter le contrôle de flux et la gestion des erreurs

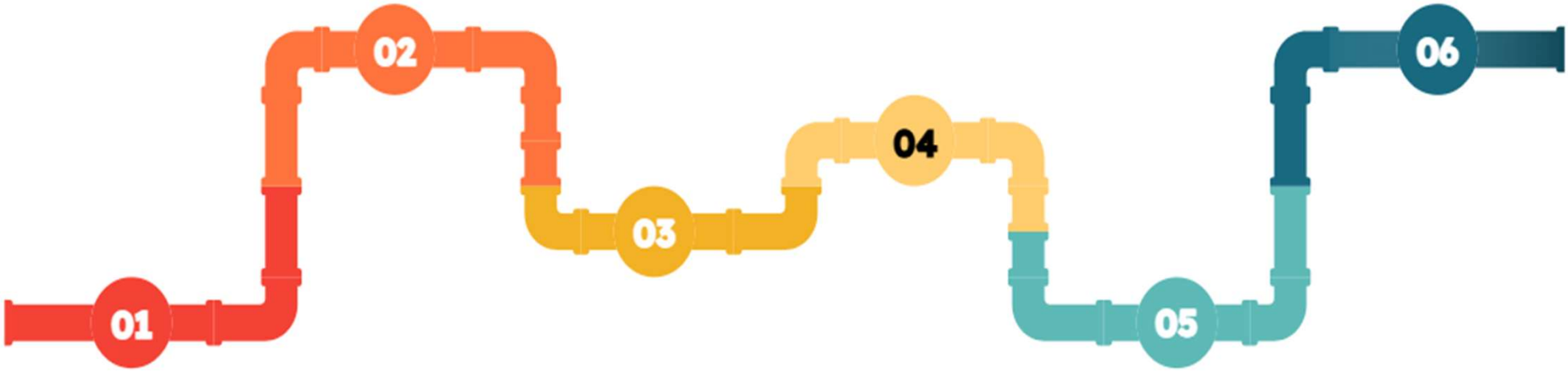
Le protocole gère le contrôle de flux et corrige les erreurs pour maintenir l'intégrité des données.

05 Appliquer dans des applications nécessitant fiabilité et intégrité

Les applications comme HTTP et FTP utilisent ce mode pour assurer la fiabilité et l'intégrité des données échangées.

Fonctionnement d’une Socket TCP

Les étapes clés garantissant une communication fiable entre applications



01 Création de la socket

Initialisation de la socket permettant la communication réseau entre applications.

02 Liaison à une adresse IP et un port

Association de la socket à une adresse IP locale et un port spécifique pour l’identification.

03 Écoute des connexions entrantes (serveur)

Le serveur attend les demandes de connexion des clients via la socket.

04 Établissement de la connexion (client)

Le client initie la connexion vers le serveur pour établir un canal de communication.

05 Échange de données bidirectionnel

Transmission fiable et ordonnée des données entre les deux applications connectées.

06 Fermeture de la connexion

Terminaison propre de la communication pour libérer les ressources réseau.

Socket en Mode Paquet : Base sur UDP

Comprendre les caractéristiques et applications des sockets UDP en mode paquet

Utiliser UDP pour rapidité

Le mode paquet utilise **UDP**, favorisant la **rapidité** au détriment de la fiabilité réseau.

Fonctionnement sans connexion

Les sockets en mode paquet n'exigent pas de **connexion** préalable, ce qui permet des échanges rapides.

Envoyer des datagrammes uniques

Chaque message est un **datagramme** indépendant, simplifiant la gestion mais réduisant la fiabilité.

Pas de garantie de livraison

Le protocole **UDP** ne garantit ni la livraison ni l'ordre des paquets, pouvant causer des pertes.

Moins de surcharge pour rapidité

Limiter les contrôles améliore la **performance** pour des applications nécessitant un traitement rapide.

Usage en streaming et jeux

Le mode UDP est idéal pour le **streaming**, jeux en ligne et **DNS** grâce à sa faible latence.

Fonctionnement d’une Socket UDP

Étapes et caractéristiques clés de la communication via une socket UDP

Création de la socket

Initialiser une socket UDP pour permettre l'envoi et la réception de datagrammes.

Association à une adresse IP et port (optionnel)

Lier la socket à une adresse IP et un port pour recevoir des datagrammes spécifiques.

Envoi et réception de datagrammes indépendants

Échanger des messages indépendants sans nécessité d'une connexion préalable.

Pas d’établissement ou fermeture de connexion

La communication ne nécessite ni connexion ni fermeture formelle, simplifiant le protocole.

Échange rapide adapté aux applications à faible latence

Ce mode de fonctionnement permet un transfert efficace pour les applications temps réel.

Socket UDP

Manipulation des Sockets UDP avec DatagramSocket

Créer un DatagramSocket pour l'envoi et la réception de datagrammes

Le DatagramSocket permet d'envoyer et de recevoir des paquets UDP sans connexion préalable, facilitant une communication rapide et flexible.

Utiliser DatagramPacket pour encapsuler les données

Les données sont encapsulées dans des DatagramPacket, qui contiennent les informations nécessaires pour l'envoi et la réception des paquets UDP.

Socket UDP

Manipulation des Sockets UDP avec DatagramSocket

Envoyer et recevoir des paquets avec adressage explicite

La communication UDP est non connectée, nécessitant l'adressage explicite de chaque paquet pour indiquer la destination ou la source.

Permettre une communication rapide adaptée aux applications multi-diffusion

L'utilisation des sockets UDP est idéale pour les applications nécessitant une diffusion rapide et efficace vers plusieurs destinataires, comme la multi-diffusion.

Exemple Simple d'Utilisation UDP en Java

01

ENVOYER UN PAQUET UDP SANS CONNEXION PRÉALABLE

LE CODE CRÉE UN DATAGRAMSOCKET, PRÉPARE UN MESSAGE SOUS FORME DE TABLEAU DE BYTES, PUIS ENVOIE UN DATAGRAMPACKET DIRECTEMENT À UNE ADRESSE IP ET UN PORT SPÉCIFIÉS.

02

ENVOYER DIRECTEMENT LE PAQUET À LA DESTINATION

L'ENVOI DU PAQUET NE NÉCESSITE PAS D'ÉTABLISSEMENT DE CONNEXION, CE QUI SIMPLIFIE LA COMMUNICATION MAIS DEMANDE UNE GESTION EXPLICITE DES PAQUETS.

03

GÉRER MANUELLEMENT LES ERREURS ET PERTES POSSIBLES

L'UTILISATION D'UDP IMPLIQUE QUE LE PROGRAMME DOIT PRENDRE EN CHARGE LA DÉTECTION ET LA GESTION DES ERREURS, CAR IL N'Y A PAS DE MÉCANISME AUTOMATIQUE DE RETRANSMISSION OU DE CONFIRMATION.