# Rapport de TP : Implémentation d'un Système RPC en Java via Sockets

## Informations générales

**Nom des étudiants : Bassma Alaoui Sossey, Safae Mamouni Alaoui, Sara Zahidi**
**Filière : Cybersécurité**
**Professeur :AMAMOU Ahmed**

---

## Objectifs du TP

Ce TP avait pour but de nous initier aux systèmes RPC (Remote Procedure Call) en Java en utilisant des sockets TCP. Les objectifs principaux étaient :

- Créer un serveur RPC qui exécute des opérations simples (add, sub, mul, div)
- Faire communiquer un client avec le serveur en utilisant une chaîne formatée
- Mettre en place la logique d'écoute, traitement et réponse entre client et serveur

---

## Structure du projet

Le projet est composé de deux fichiers Java :

```
RPCProject/
├── RPCServer.java
└── RPCClient.java
```

---

## 1. Code Serveur (RPCServer.java)

Le serveur ouvre une socket TCP sur le port 12345, attend les connexions des clients, puis traite leurs requêtes dans un thread séparé pour chaque client.

Les opérations prises en charge sont : addition, soustraction, multiplication, division. La requête envoyée par le client est une chaîne formatée de type : `operation:a:b`

Extrait du code :

```java
public static void handleClient(Socket socket) {
    String request = in.readLine(); // Ex: "add:4:5"
```

```
    String[] parts = request.split(":");
    String function = parts[0];
    double a = Double.parseDouble(parts[1]);
    double b = Double.parseDouble(parts[2]);
    // calcul et renvoi du résultat selon l'opération
}
```

---

## 2. Code Client (RPCClient.java)

Le client se connecte au serveur sur le port 12345, demande à l'utilisateur une opération et deux opérandes, envoie la requête au serveur, et affiche la réponse retournée.

Extrait du code :

```
System.out.print("Enter operation (add/sub/mul/div): ");
String op = userIn.readLine();
String request = op + ":" + a + ":" + b;
out.println(request);
String response = in.readLine();
System.out.println("Result: " + response);
```

---

## 3. Instructions d'exécution

1. Compilation :

```
javac RPCServer.java RPCClient.java
```

2. Lancer le serveur :

```
java RPCServer
```

3. Dans un autre terminal, lancer le client :

```
java RPCClient
```

---

## Difficultés rencontrées

- Problèmes de synchronisation client-serveur lors de la première exécution
- Erreurs de parsing dues à des espaces ou mauvaises chaînes envoyées
- Plusieurs postes à l'IUT étaient en panne, ce qui a décalé les tests

- Gestion du multithreading sur le serveur au début mal implémentée (correction via Thread)

---

## Conclusion

Ce TP nous a permis de comprendre le fonctionnement basique d'un système RPC, d'implémenter une communication réseau en Java, et d'organiser un code serveur/client utilisant des sockets TCP. Nous avons aussi consolidé nos compétences en manipulation de chaînes, gestion des flux, et traitement des entrées utilisateur.

Ce type de TP est fondamental pour comprendre les bases des systèmes distribués et l'architecture client-serveur.