REPUBLIQUE TUNISIENNE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Compte Rendu TP 4

Spécialité : Réseaux informatiques et télécommunications

Simulation de l'Algorithme de Consensus Paxos

Présenté par

Riadh Ibrahim
Iheb Alimi

Année Universitaire : 2024 / 2025

Table des matières

1. Introduction

Ce TP vise à implémenter une simulation de l'algorithme de consensus Paxos à l'aide d'un projet Java/Maven utilisant gRPC pour la communication entre les agents. L'objectif est de permettre à trois proposers (serveurs) de générer des valeurs aléatoires, qu'un acceptor (client) sélectionne la valeur la plus élevée comme valeur de consensus, puis la diffuse aux proposers pour stockage.

2. Objectifs du TP

1. **Comprendre l'algorithme Paxos** : Simuler un protocole de consensus où plusieurs agents (proposers) proposent des valeurs, et un agent (acceptor) choisit une valeur consensuelle.
2. **Mettre en œuvre une architecture client-serveur** : Utiliser gRPC pour la communication entre les proposers (serveurs) et l'acceptor (client).
3. **Créer un projet Maven** : Configurer un environnement de développement avec IntelliJ IDEA et Maven pour gérer les dépendances et la compilation.
4. **Exécuter et tester** : Lancer plusieurs instances de serveurs et un client, observer les résultats et vérifier le bon fonctionnement du consensus.

3. Architecture du Projet

Le système est composé de deux types d'agents :

1. **Proposers (Serveurs)** :
    - **Rôle** : Générer une valeur aléatoire, participer au protocole Paxos, et stocker les valeurs de consensus.
    - **Nombre** : 3 serveurs, chacun exécuté sur un port distinct (50051, 50052, 50053).
    - **Comportement** :
        - Génère une valeur aléatoire au démarrage (localValue).
        - Répond aux requêtes du client via l'appel gRPC proposeValue.
        - Stocke les valeurs de consensus reçues via l'appel gRPC storeConsensusValue dans un vecteur (acceptedValues).
2. **Acceptor (Client)** :
    - **Rôle** : Coordonner le processus de consensus.
    - **Comportement** :
        - Se connecte aux trois serveurs via leurs ports.
        - Récupère les valeurs proposées (a, b, c) en appelant proposeValue sur chaque serveur.
        - Sélectionne la valeur maximale (R = max(a, b, c)).

        - Diffuse R à tous les serveurs via storeConsensusValue.
        - Affiche la valeur consensuelle et confirme sa diffusion.

Communication

La communication entre les agents est assurée par **gRPC**, un framework RPC basé sur Protocol Buffers. Le fichier paxos.proto définit :

- **Messages** :
    - ValueRequest : Requête vide avec un identifiant.
    - ValueResponse : Contient la valeur proposée et l'identifiant du serveur.
    - ConsensusValue : Contient la valeur de consensus R.
    - StoreAck : Confirme le stockage de la valeur.
- **Services** :
    - ProposeValue : Permet au client de récupérer une valeur proposée.
    - StoreConsensusValue : Permet au client de diffuser la valeur de consensus.

4. Étapes de Mise en Œuvre

Création du Projet Maven

- **Outil** : IntelliJ IDEA.
- **Étapes** :
    1. Créer un nouveau projet Maven avec :
        - **GroupId** : rt4.paxos
        - **ArtifactId** : paxos
        - **Version** : 1.0-SNAPSHOT
    2. Configurer le fichier pom.xml (fourni via Classroom) pour inclure les dépendances gRPC et autres bibliothèques nécessaires.
    3. Vérifier la structure du projet :
        - src/main/java/rt4.paxos : Contient les fichiers Java (PaxosProposer.java, PaxosAcceptor.java, PaxosServiceImpl.java).
        - src/main/proto : Contient paxos.proto.
        - pom.xml : Fichier de configuration Maven.

Définition du Fichier paxos.proto

Le fichier paxos.proto définit la structure des messages et des services gRPC :

- **Messages** :
    - ValueRequest : Requête vide.
    - ValueResponse : Contient la valeur générée et l'identifiant du serveur.

    - ConsensusValue : Contient la valeur de consensus.
    - StoreAck : Confirme le stockage.
- **Services** :
    - PaxosService avec deux méthodes : proposeValue et storeConsensusValue.

Implémentation des Composants

a. PaxosServiceImpl.java

- **Rôle** : Implémentation des méthodes gRPC côté serveur.
- **Fonctionnalités** :
    - Génère une valeur aléatoire (localValue) au démarrage.
    - proposeValue : Retourne localValue et l'identifiant du serveur.
    - storeConsensusValue : Ajoute la valeur de consensus reçue à acceptedValues.

b. PaxosProposer.java

- **Rôle** : Lancer un serveur gRPC.
- **Fonctionnalités** :
    - Prend un port en argument (ex. 50051).
    - Crée un serveur gRPC avec PaxosServiceImpl.
    - Affiche un message confirmant le démarrage.
    - Reste actif indéfiniment.

c. PaxosAcceptor.java

- **Rôle** : Coordonner le consensus.
- **Fonctionnalités** :
    - Établit une connexion gRPC avec les trois serveurs.
    - Appelle proposeValue pour récupérer les valeurs proposées.
    - Calcule la valeur maximale (R).
    - Diffuse R via storeConsensusValue.
    - Affiche les résultats.

Compilation et Exécution

Compilation

```
Terminal  Local  Server 1  Server 2  Server 3
[WARNING] mvn dependency:tree -Ddetail=true and the above output.
[WARNING] See https://maven.apache.org/plugins/maven-shade-plugin/
[INFO] Replacing original artifact with shaded artifact.
[INFO] Replacing C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus\target\paxos-1.0-SNAPSHOT.jar with C:\Use
.jar
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ paxos ---
[INFO] Installing C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus\dependency-reduced-pom.xml to C:\Users\h
[INFO] Installing C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus\target\paxos-1.0-SNAPSHOT.jar to C:\User
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  39.486 s
[INFO] Finished at: 2025-04-26T14:59:42+01:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus>
```

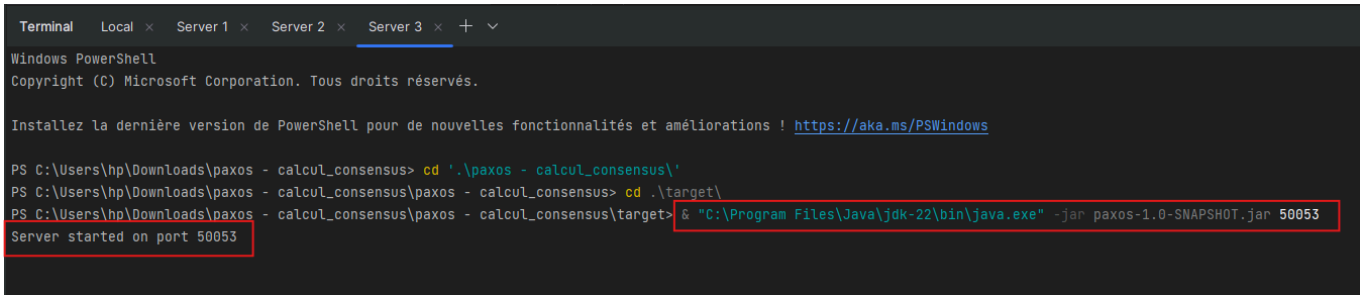
Lancement des serveurs : Dans trois terminaux distincts

```
Terminal  Local  Server 1  Server 2  Server 3
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(Unknown Source)
        at java.security.SecureClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.access$100(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at sun.launcher.LauncherHelper.checkAndLoadMain(Unknown Source)
PS C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus\target> & "C:\Program Files\Java\jdk-22\bin\java.exe" -jar paxos-1.0-SNAPSHOT.jar 50051
Server started on port 50051
```

```
Terminal  Local  Server 1  Server 2  Server 3
Windows PowerShell
Copyright (C) Microsoft Corporation. Tous droits réservés.

Installez la dernière version de PowerShell pour de nouvelles fonctionnalités et améliorations ! https://aka.ms/PSWindows

PS C:\Users\hp\Downloads\paxos - calcul_consensus> cd '.\paxos - calcul_consensus\'
PS C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus> cd .\target\
PS C:\Users\hp\Downloads\paxos - calcul_consensus\paxos - calcul_consensus\target> & "C:\Program Files\Java\jdk-22\bin\java.exe" -jar paxos-1.0-SNAPSHOT.jar 50052
Server started on port 50052
```

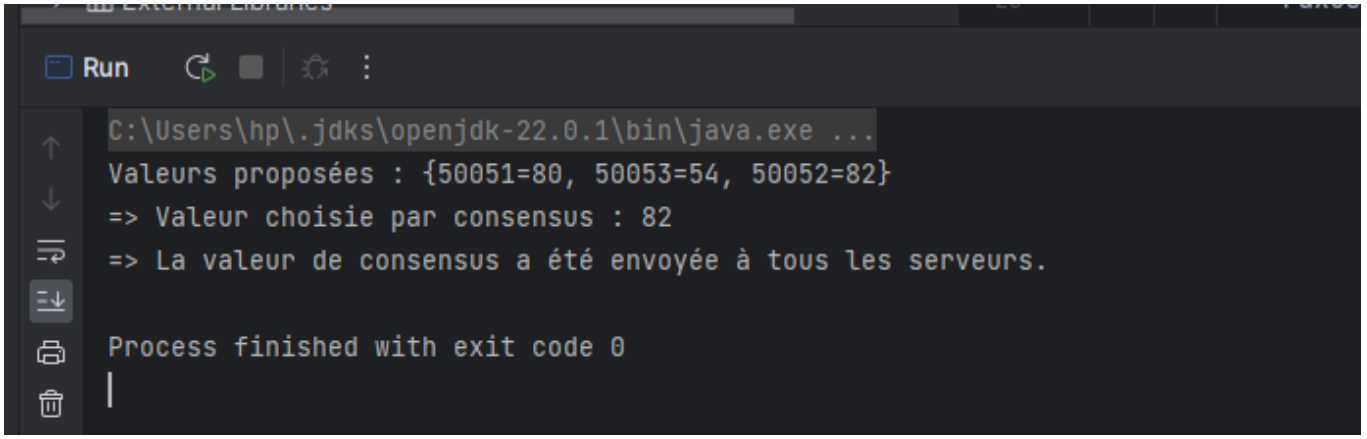


Lancement de l'Acceptor

Résultats Obtenus

Exécution de l'Acceptor

La sortie de l'Acceptor est :



Valeurs proposées : {50051=80, 50053=54, 50052=82}
=> Valeur choisie par consensus : 82
=> La valeur de consensus a été envoyée à tous les serveurs.

- **Analyse** :
  - Les serveurs ont proposé les valeurs 80, 82 et 54.
  - L'Acceptor a correctement identifié la valeur maximale (82).
  - La valeur de consensus a été diffusée à tous les serveurs.

```
Server ID : S50051 - Proposed Value :80
---> List of Accepted Values in ServerID : S50051
82
```

```
Server started on port 50052
Server ID : S50052 - Proposed Value :82
---> List of Accepted Values in ServerID : S50052
82
```

```
Server started on port 50053
Server ID : S50053 - Proposed Value :54
---> List of Accepted Values in ServerID : S50053
82
```

5. Modification du Projet Initial : Passage à une Interface Graphique de Bureau

Dans la configuration initiale du projet Paxos, il était prévu d'intégrer une application web pour visualiser les valeurs proposées par les serveurs (ports 50051, 50052, 50053) et la valeur de consensus calculée par l'Accepteur. Cependant, afin de simplifier l'intégration et de rester dans l'écosystème Java/Maven tout en évitant les complexités liées aux technologies web, cette approche a été abandonnée. À la place, une interface graphique de bureau (GUI) a été développée en utilisant Java Swing, directement intégrée dans IntelliJ IDEA. L'objectif principal de cette modification était de fournir une visualisation en temps réel des données du protocole Paxos tout en facilitant l'exécution et le débogage dans un environnement de développement familier.

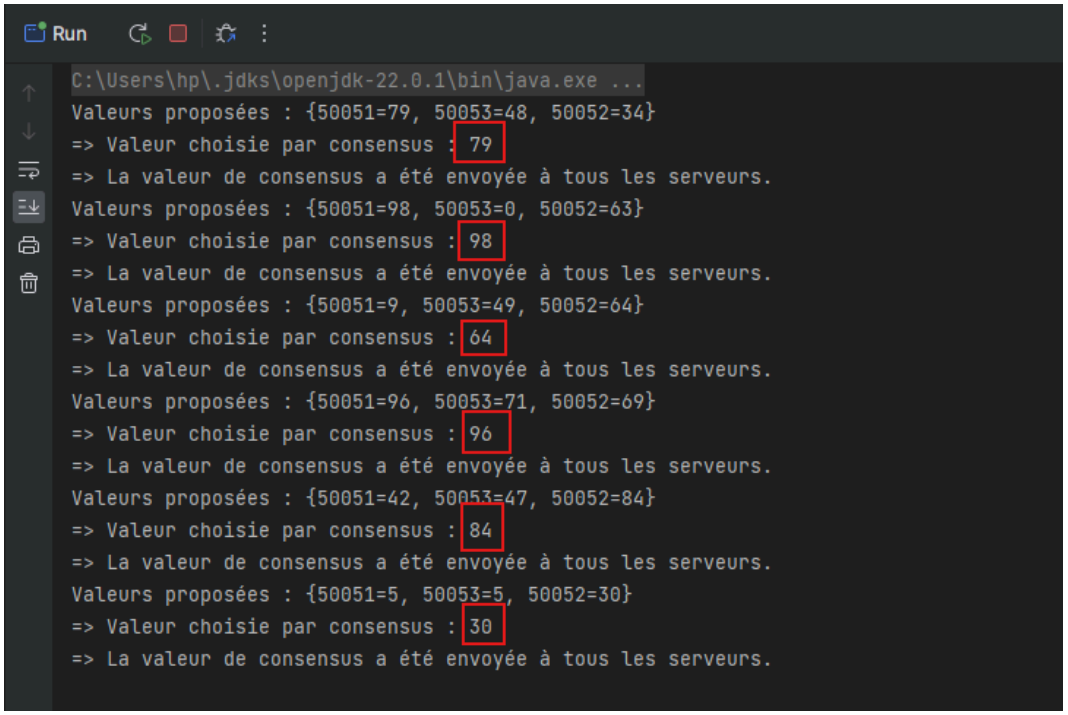
Mise en Œuvre

La modification a impliqué plusieurs étapes clés :

1. **Création de l'Interface Graphique avec Java Swing** :
    - Une nouvelle classe, PaxosVisualizer.java, a été ajoutée pour implémenter une interface graphique utilisant Java Swing. Cette interface affiche les valeurs proposées par chaque serveur dans des panneaux distincts, ainsi que la valeur de consensus dans un panneau mis en évidence. Un bouton "Rafraîchir les Valeurs" permet à l'utilisateur de déclencher de nouveaux cycles de consensus.

- L'interface est intégrée directement dans le projet Paxos, évitant ainsi le besoin de composants externes ou de serveurs web.

2. **Modification de** PaxosAcceptor.java :
   - La classe PaxosAcceptor a été adaptée pour stocker les valeurs proposées et la valeur de consensus dans des champs accessibles (proposedValues et consensusValue). Des méthodes d'accès (getProposedValues, getConsensusValue, getTargetPorts) ont été ajoutées pour permettre à l'interface graphique de récupérer ces données.
   - La logique de consensus a été extraite de la méthode main et placée dans une méthode runConsensus, rendant le processus réutilisable pour les rafraîchissements manuels via l'interface.
   - L'interface graphique est lancée automatiquement à la fin de l'exécution initiale de PaxosAcceptor, en utilisant SwingUtilities.invokeLater pour garantir la sécurité des threads.




```
---> List of Accepted Values in ServerID : S50051
79
98
64
96
84
30
```

```
---> List of Accepted Values in ServerID : S50052
79
98
64
96
84
30
```

```
---> List of Accepted Values in ServerID : S50053
79
98
64
96
84
30
```