# Système de gestion de données distribué

**Rapport de projet**

Cours de systèmes distribués
M1 Informatique Fondamentale
ENS de Lyon
Printemps 2018

guillaume.coiffier@ens-lyon.fr

L'intégralité du code source est disponible à l'adresse suivante :
https://github.com/GCoiffier/Distributed-Data-Manager-Erlang

# Table des matières

**1 Description du programme** **2**
1.1 Topologie du réseau . . . . . 2
1.2 Initialisation du server . . . . . 2
1.3 Méthode de stockage . . . . . 2

**2 Fonctionnement du programme** **3**
2.1 Interface client . . . . . 3
2.2 Gestion des requêtes d'un client par les noeuds de requête . . . . . 3

**3 Discussion sur les performances et la robustesse de la solution choisie** **4**
3.1 Robustesse du réseau . . . . . 4
3.2 Complexité de communication . . . . . 4

# 1. Description du programme

## 1.1 Topologie du réseau

Nous avons fait le choix d'imposer une topologie précise à notre réseau, plutôt que de gérer une topologie arbitraire. Les conséquences de ce choix sont discutées en chapitre 3.
On distinguera, dans le réseau, trois types d'agents :

— Le noeud **master**, responsable des connections entrantes au réseau.
— Les noeuds de requête (**query**), responsables de la gestion des requêtes de stockage et d'accès aux données.
— Les noeuds de stockage (**storage**) responsables... du stockage des données.

Le noeud **master** est relié à l'intégralité des noeuds **query**. Ces derniers forment une clique. Dans la pratique, ils ont simplement accès à la liste des Pid de tous les autres noeuds query. Enfin, les noeuds de stockages sont reliés à un ou plusieurs noeuds **query** et forment leurs fils.

## 1.2 Initialisation du server

Le noeud master (module `server.erl`) est responsable de l'Initialisation du serveur. Il est lancé lorsque l'on compile le module `server`. Il a la tâche d'initialiser N noeuds de requêtes, qui vont à leur tour initialiser M noeuds de stockage chacun.
Les valeurs par défaut de N et M sont respectivement 10 et 5.

## 1.3 Méthode de stockage

Le stockage des données est assuré par un

# 2. Fonctionnement du programme

## 2.1 Interface client

L'interface du client est décrite par le module `client.erl`. Elle comprend les fonctions suivantes :

— `connect/1`
— `send_data/2`
— `fetch_data/`
— `release_data`

## 2.2 Gestion des requêtes d'un client par les noeuds de requête

# 3. Discussion sur les performances et la robustesse de la solution choisie

## 3.1 Robustesse du réseau

## 3.2 Complexité de communication