

DM2 Peer to Peer : Kademlia

Quentin Dubois

2023-10-01

Références :

- [bmuller/kademlia: A DHT in Python using asyncio \(github.com\)](#)
- [Kademlia Documentation — Kademlia 2.2.1 documentation](#)

Introduction

Le projet GitHub, [bmuller/kademlia](#) est une bibliothèque Python contenant une implémentation asynchrone de la table de hachage distribuée Kademlia. Elle utilise la bibliothèque asyncio de Python 3 pour fournir une communication asynchrone. Les nœuds communiquent en utilisant RPC sur UDP pour communiquer, ce qui signifie qu'elle est capable de fonctionner derrière un NAT.

Installation

Récupération du code

J'ai repris les [exemples des développeurs](#) et les ai un peu modifiés afin de pouvoir passer des paramètres supplémentaires. Ils sont trouvables sur mon GitHub : [QuentinDubois-Polytech/Kademlia \(github.com\)](#)

```
git clone https://github.com/QuentinDubois-Polytech/Kademlia.git
```

Installation de la bibliothèque

Création d'un environnement virtuel Python :

```
python3 -m venv venv
```

Utilisation de l'environnement virtuel :

Unix :

```
source venv/bin/activate
```

Windows :

```
# Besoin d'activer l'exécution des scripts si ce n'est pas activé par défaut
Set-ExecutionPolicy Unrestricted -Scope Process
```

```
# Activation du venv
venv/Scripts/activate.ps1
```

Installation de la bibliothèque Python :

La bibliothèque Python est disponible en tant que paquet Python sur PyPi sous le nom : kademia ([kademlia](#) · [PyPI](#))

Nous allons donc utiliser le paquet Python pour l'installation.

```
pip3 install kademia
```

Scripts

Il existe 3 scripts :

- node.py
- get.py
- set.py

Node.py

Ce script permet d'initialiser le premier noeud d'un réseau Kademia ou d'ajouter un nouveau noeud à un réseau déjà existant

Création d'un réseau :

Créer un nouveau noeud sur le port UDP, <port>.

```
python3 node.py -P <port>
```

Ajout d'un nouveau noeud :

Créer un nouveau noeud sur le port UDP, <port>. <joined_ip> et <joined_port> sont respectivement l'adresse IP et le port d'un noeud présent sur le réseau.

```
python3 node.py -P <port> -i <joined_ip> -p <joined_port>
```

Get.py

Ce script permet de récupérer la valeur associée à la clé <key> dans le réseau.

```
python3 get.py -i <joined_ip> -p <joined_port> -k <key>
```

Pour récupérer une valeur, le script crée un nouveau noeud dans le réseau, effectue l'opération "get", puis se déconnecte du réseau.

Set.py

Ce script permet d'ajouter un nouveau tuple (<key>, <value>) dans le réseau.

```
python3 set.py -i <joined_ip> -p <joined_port> -k <key> -v <value>
```

Pour ajouter une valeur, le script crée un nouveau noeud dans le réseau, effectue l'opération "set", puis se déconnecte du réseau.

Expérimentations

Pour les expérimentations, j'ai testé sous cet environnement :

- Ubuntu 22.04
- Python 3.10.6
- Paquet Python "kademlia" 2.2.2

Création du premier noeud

Nous allons créer notre premier noeud sur le port 8000.

```
python3 node.py -P 8000
```

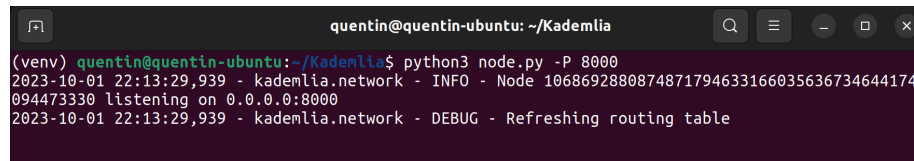
A screenshot of a terminal window with a dark background. The title bar shows 'quentin@quentin-ubuntu: ~/Kademlia'. The prompt is '(venv) quentin@quentin-ubuntu:~/Kademlia\$'. The command 'python3 node.py -P 8000' has been executed. The output shows a timestamp '2023-10-01 22:13:29,939', a log level 'INFO', and a message 'Node 1068692880874871794633166035636734644174094473330 listening on 0.0.0.0:8000'. A second line shows a timestamp '2023-10-01 22:13:29,939', a log level 'DEBUG', and a message 'Refreshing routing table'.

Figure 1: Création du premier noeud sur le port 8000

Ajout d'un nouveau noeud

Nous allons ajouter un noeud sur le port 8001 avec pour noeud d'entrée celui sur le port 8000.

```
python3 node.py -P 8001 -p 8000
```

Dans cet exemple, je ne spécifie pas l'adresse IP du noeud que mon noeud veut rejoindre, car il se trouve sur l'interface réseau, local (0.0.0.0), qui correspond à la valeur par défaut, pour le champ IP

Nous remarquons que le noeud déjà existant, celui qui a permis au nouveau noeud d'entrer dans le réseau, vient d'ajouter ce nouveau noeud dans sa table de routage.

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8001 -p 8000
2023-10-01 22:15:25,666 - kademlia.network - INFO - Node 836309640390698953375430756659753257723337
513363 listening on 0.0.0.0:8001
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Attempting to bootstrap node with 1 initial co
ntacts
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - creating spider with peers: [[1068692880874871
794633166035636734644174094473330, '0.0.0.0', 8000]]
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - crawling network with nearest: ([1068692880874
871794633166035636734644174094473330, '0.0.0.0', 8000],)
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - never seen 0.0.0.0:8000 before, adding to rout
er
```

Figure 2: Logs du nouveau noeud

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8000
2023-10-01 22:13:29,939 - kademlia.network - INFO - Node 1068692880874871794633166035636734644174
094473330 listening on 0.0.0.0:8000
2023-10-01 22:13:29,939 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,670 - kademlia.protocol - INFO - never seen 127.0.0.1:8001 before, adding to
router
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - finding neighbors of 83630964039069895337543
0756659753257723337513363 in local table
```

Figure 3: Logs du noeud existant

Ajout d'une donnée

Nous allons maintenant ajouter une nouvelle donnée, le tuple : $(key, value) = (luigi, +33678358088)$. Vu l'implémentation actuelle, un nouveau noeud va être créé, il a donc besoin de connaître un noeud pour rentrer dans le réseau comme pour le script node.py

```
python3 set.py -P 8000 -k luigi -v '+33 6 78 35 80 88'
```

On remarque les deux noeuds du réseau sur les ports 8000 et 8001, ont découvert un nouveau noeud sur le réseau au port 24926, suite à la demande de ce noeud, pour stocker une donnée. Ils ont ensuite reçu tous les deux une requête pour stocker une donnée. Dans Kademlia, contrairement à Chord, la donnée est répliquée sur un certain nombre de noeud. Dans notre exemple, elle est stockée sur les deux noeuds.

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 set.py -p 8000 -k luigi -v '+33 6 78 35 80 88'
2023-10-01 22:24:34,098 - kademlia.network - INFO - Node 2394865713697053009143490297849276313150
95471460 listening on 0.0.0.0:24926
2023-10-01 22:24:34,099 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:24:34,099 - kademlia.network - DEBUG - Attempting to bootstrap node with 1 initial
contacts
2023-10-01 22:24:34,100 - kademlia.crawling - INFO - creating spider with peers: [[10686928808748
71794633166035636734644174094473330, '0.0.0.0', 8000]]
2023-10-01 22:24:34,100 - kademlia.crawling - INFO - crawling network with nearest: ([10686928808
74871794633166035636734644174094473330, '0.0.0.0', 8000],)
2023-10-01 22:24:34,101 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:24:34,101 - kademlia.protocol - INFO - never seen 0.0.0.0:8000 before, adding to ro
uter
2023-10-01 22:24:34,101 - kademlia.crawling - INFO - crawling network with nearest: ([10686928808
74871794633166035636734644174094473330, '0.0.0.0', 8000], [83630964039069895337543075665975325772
3337513363, '127.0.0.1', 8001])
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - got successful response from 127.0.0.1:8001
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - never seen 127.0.0.1:8001 before, adding to
router
2023-10-01 22:24:34,102 - kademlia.network - INFO - setting 'luigi' = '+33 6 78 35 80 88' on netw
ork
2023-10-01 22:24:34,102 - kademlia.crawling - INFO - creating spider with peers: [[83630964039069
8953375430756659753257723337513363, '127.0.0.1', 8001], [1068692880874871794633166035636734644174
094473330, '0.0.0.0', 8000]]
2023-10-01 22:24:34,102 - kademlia.crawling - INFO - crawling network with nearest: ([83630964039
0698953375430756659753257723337513363, '127.0.0.1', 8001], [1068692880874871794633166035636734644
174094473330, '0.0.0.0', 8000])
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - got successful response from 127.0.0.1:8001
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:24:34,103 - kademlia.network - INFO - setting 'd1878f7213a7afa7d418ec69f4aed914c9e0
7d8d' on ['127.0.0.1:8001', '0.0.0.0:8000']
2023-10-01 22:24:34,104 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:24:34,104 - kademlia.protocol - INFO - got successful response from 127.0.0.1:8001
Key added !
```

Figure 4: Logs du noeud ajoutant une donnée

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8000
2023-10-01 22:13:29,939 - kademlia.network - INFO - Node 1068692880874871794633166035636734644174
094473330 listening on 0.0.0.0:8000
2023-10-01 22:13:29,939 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,670 - kademlia.protocol - INFO - never seen 127.0.0.1:8001 before, adding to
router
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - finding neighbors of 83630964039069895337543
0756659753257723337513363 in local table
2023-10-01 22:24:34,099 - kademlia.protocol - INFO - never seen 127.0.0.1:24926 before, adding to
router
2023-10-01 22:24:34,100 - kademlia.protocol - INFO - finding neighbors of 23948657136970530091434
9029784927631315095471460 in local table
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - finding neighbors of 23948657136970530091434
9029784927631315095471460 in local table
2023-10-01 22:24:34,104 - kademlia.protocol - DEBUG - got a store request from ('127.0.0.1', 2492
6), storing 'd1878f7213a7afa7d418ec69f4aed914c9e07d8d'='+33 6 78 35 80 88'
```

Figure 5: Logs du noeud originel

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8001 -p 8000
2023-10-01 22:15:25,666 - kademlia.network - INFO - Node 836309640390698953375430756659753257723337
513363 listening on 0.0.0.0:8001
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Attempting to bootstrap node with 1 initial co
ntacts
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - creating spider with peers: [[1068692880874871
794633166035636734644174094473330, '0.0.0.0', 8000]]
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - crawling network with nearest: ([1068692880874
871794633166035636734644174094473330, '0.0.0.0', 8000],)
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - never seen 0.0.0.0:8000 before, adding to rout
er
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - finding neighbors of 2394865713697053009143490
29784927631315095471460 in local table
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - never seen 127.0.0.1:24926 before, adding to r
outer
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - finding neighbors of 2394865713697053009143490
29784927631315095471460 in local table
2023-10-01 22:24:34,104 - kademlia.protocol - DEBUG - got a store request from ('127.0.0.1', 24926)
, storing 'd1878f7213a7afa7d418ec69f4aed914c9e07d8d'='+33 6 78 35 80 88'
```

Figure 6: Logs du second noeud

Récupération de la donnée

Nous allons désormais récupérer la valeur stockée sous la clé : *key = luigi*
Nous allons récupérer cette valeur en contactant le noeud sur le port 8001.

```
python3 get.py -P 8001 -k luigi
```

Nous arrivons bien à récupérer la valeur stockée sous la clé *luigi* qui est +33678358088. Nous remarquons également que la demande GET n'est arrivée que sur le noeud au port 8001 et que celui-ci a répondu directement. Ce phénomène est normal puisque que ce noeud possède la donnée demandée. Finalement, le nouveau noeud a été ajouté aux tables de routages respectives, des noeuds sur les ports 8000 et 8001.

Points d'amélioration

- Remplacer les scripts “get.py” et “set.py” par un client Python qui puisse se connecter à un noeud s'exécutant en local afin de lui spécifier interactivement les actions à effectuer. La communication s'effectuerait par un mécanisme d'IPC (Inter Communication Process).
- Ajouter une commande pour lister la table des clés et valeurs d'un noeud
- Ajouter une commande pour afficher les tables de routages (k-buckets) d'un noeud

Si une personne est motivée et désire continuer ce travail, ce serait avec plaisir.

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 get.py -p 8001 -k luigi
2023-10-01 22:36:11,481 - kademlia.network - INFO - Node 1009906128638425658120807366413548477780975
925250 listening on 0.0.0.0:35039
2023-10-01 22:36:11,481 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:36:11,481 - kademlia.network - DEBUG - Attempting to bootstrap node with 1 initial con
tacts
2023-10-01 22:36:11,483 - kademlia.protocol - INFO - never seen 127.0.0.1:8001 before, adding to rou
ter
2023-10-01 22:36:11,484 - kademlia.protocol - DEBUG - got a store request from ('127.0.0.1', 8001),
storing 'd1878f7213a7afa7d418ec69f4aed914c9e07d8d'='+33 6 78 35 80 88'
2023-10-01 22:36:11,484 - kademlia.crawling - INFO - creating spider with peers: [[83630964039069895
3375430756659753257723337513363, '0.0.0.0', 8001]]
2023-10-01 22:36:11,484 - kademlia.crawling - INFO - crawling network with nearest: ([83630964039069
8953375430756659753257723337513363, '0.0.0.0', 8001],)
2023-10-01 22:36:11,485 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8001
2023-10-01 22:36:11,485 - kademlia.crawling - INFO - crawling network with nearest: ([10686928808748
71794633166035636734644174094473330, '0.0.0.0', 8000], [83630964039069895337543075665975325772333751
3363, '0.0.0.0', 8001], [239486571369705300914349029784927631315095471460, '127.0.0.1', 24926])
2023-10-01 22:36:11,486 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:36:11,486 - kademlia.protocol - INFO - never seen 0.0.0.0:8000 before, adding to route
r
Did not receive reply for msg id b'DhVRZ1KaIqZ4Ifsv75LT6dSEak=' within 5 seconds
2023-10-01 22:36:16,491 - kademlia.protocol - WARNING - no response from 127.0.0.1:24926, removing f
rom router
2023-10-01 22:36:16,491 - kademlia.network - INFO - Looking up key luigi
Key found ! -> luigi = +33 6 78 35 80 88
```

Figure 7: Logs du noeud récupérant la donnée

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8001 -p 8000
2023-10-01 22:15:25,666 - kademlia.network - INFO - Node 836309640390698953375430756659753257723337
513363 listening on 0.0.0.0:8001
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,667 - kademlia.network - DEBUG - Attempting to bootstrap node with 1 initial co
ntacts
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - creating spider with peers: [[1068692880874871
794633166035636734644174094473330, '0.0.0.0', 8000]]
2023-10-01 22:15:25,672 - kademlia.crawling - INFO - crawling network with nearest: ([1068692880874
871794633166035636734644174094473330, '0.0.0.0', 8000],)
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - got successful response from 0.0.0.0:8000
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - never seen 0.0.0.0:8000 before, adding to rout
er
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - finding neighbors of 2394865713697053009143490
29784927631315095471460 in local table
2023-10-01 22:24:34,102 - kademlia.protocol - INFO - never seen 127.0.0.1:24926 before, adding to r
outer
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - finding neighbors of 2394865713697053009143490
29784927631315095471460 in local table
2023-10-01 22:24:34,104 - kademlia.protocol - DEBUG - got a store request from ('127.0.0.1', 24926)
, storing 'd1878f7213a7afa7d418ec69f4aed914c9e07d8d'='+33 6 78 35 80 88'
2023-10-01 22:36:11,482 - kademlia.protocol - INFO - never seen 127.0.0.1:35039 before, adding to
router
2023-10-01 22:36:11,484 - kademlia.protocol - INFO - got successful response from 127.0.0.1:35039
2023-10-01 22:36:11,484 - kademlia.protocol - INFO - finding neighbors of 1009906128638425658120807
366413548477780975925250 in local table
```

Figure 8: Logs du noeud recevant la demande GET

```
quentin@quentin-ubuntu: ~/Kademlia
(venv) quentin@quentin-ubuntu:~/Kademlia$ python3 node.py -P 8000
2023-10-01 22:13:29,939 - kademlia.network - INFO - Node 1068692880874871794633166035636734644174
094473330 listening on 0.0.0.0:8000
2023-10-01 22:13:29,939 - kademlia.network - DEBUG - Refreshing routing table
2023-10-01 22:15:25,670 - kademlia.protocol - INFO - never seen 127.0.0.1:8001 before, adding to
router
2023-10-01 22:15:25,673 - kademlia.protocol - INFO - finding neighbors of 83630964039069895337543
0756659753257723337513363 in local table
2023-10-01 22:24:34,099 - kademlia.protocol - INFO - never seen 127.0.0.1:24926 before, adding to
router
2023-10-01 22:24:34,100 - kademlia.protocol - INFO - finding neighbors of 23948657136970530091434
9029784927631315095471460 in local table
2023-10-01 22:24:34,103 - kademlia.protocol - INFO - finding neighbors of 23948657136970530091434
9029784927631315095471460 in local table
2023-10-01 22:24:34,104 - kademlia.protocol - DEBUG - got a store request from ('127.0.0.1', 2492
6), storing 'd1878f7213a7afa7d418ec69f4aed914c9e07d8d'='+33 6 78 35 80 88'
2023-10-01 22:36:11,486 - kademlia.protocol - INFO - finding neighbors of 10099061286384256581208
07366413548477780975925250 in local table
2023-10-01 22:36:11,486 - kademlia.protocol - INFO - never seen 127.0.0.1:35039 before, adding to
router
```

Figure 9: Logs de l'autre noeud