**Kademlia DHT implementation :**

Hello,

For this homework, I wrote my own implementation of Kademlia DHT in Rust language. I used the **kademlia-dht** library available at this address: <https://crates.io/crates/kademlia-dht>

**How to install?**

If you use Windows on a 64 bits platform, you can just download the executable here: <https://github.com/thomasarmel/simple_kademlia_implementation/releases/tag/v1>

Otherwise, run the following commands:

*git clone* [*https://github.com/thomasarmel/simple\_kademlia\_implementation.git*](https://github.com/thomasarmel/simple_kademlia_implementation.git)

*cd simple\_kademlia\_implementation*

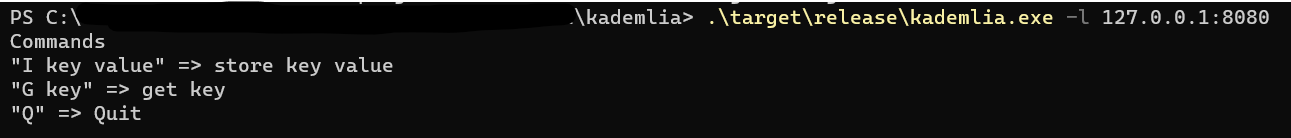
*cargo build –release*

**How to use it?**

At first, run the first node:

*.\target\release\kademlia.exe -l 127.0.0.1:8080*

In case you want to listen on local interface on port 8080.



Then another node can join using the first as entry point:

.\target\release\kademlia.exe -l 127.0.0.1:8081 -r 127.0.0.1:8080

It will listen on local interface, on port 8081

Finally, a third node can join using the second one as entry point:

.\target\release\kademlia.exe -l 127.0.0.1:8082 -r 127.0.0.1:8081

It will listen on local interface, on port 8082

**Commands:**

*I my\_key my\_value* => store **my\_value** using key **my\_key**.

G my\_key => Read value stored with key **my\_key**.

*Q* => Quit.

**Example:**

Create first node and insert **key1 => value1**:

Une image contenant texte

Description générée automatiquement

Create second node, insert **key2 => value2** and read **key1**:

Une image contenant texte

Description générée automatiquement

Create third node, insert **key3 => value3** and read **key1** and **key2**:

Une image contenant texte

Description générée automatiquement

On first node, read **key2** and **key3**:

Une image contenant texte

Description générée automatiquement

If we modify a value on a node:



Then it will be modified for all nodes:

Une image contenant texte

Description générée automatiquement

**How it works?**

**Hashing**

Each node will have an unique identifier, **sha256(ip:port):**



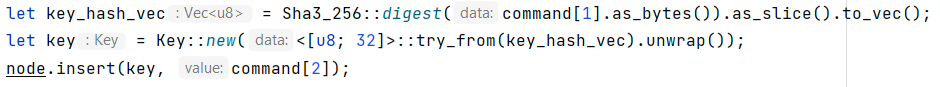
When a node wants to join an existing DHT, it needs to know the entry point and the entry point ID:

Une image contenant texte

Description générée automatiquement

The same hashing algorithm is using for the keys.

Here the insertion:



And the reading:

Une image contenant texte

Description générée automatiquement

**XOR distance**

Here you can see the closest node is calculated using the XOR distance:

Une image contenant texte

Description générée automatiquement

Une image contenant texte

Description générée automatiquement

**Buckets:**

We can see buckets work using the node hash as identifier.

Une image contenant texte

Description générée automatiquement

**Find a value:**

If the value is stored on local node, return it. Otherwise, find the closest node for the value in the routing table.

Une image contenant texte

Description générée automatiquement

And then ask this node for the value.