D7024 Lab report
Grand repository in the sky

MARTIN Aline, GERMAIN Timothée

November 6, 2014

1 Frameworks and tools used

1.1 Operating system

We worked exclusively on linux(manjaro and ubuntu), because environement setup for coding was simple. Ubuntu was a 32bits version first, but it had to be changed to a 64bits version in the middle of the project because of docker compatibility.

1.2 Tools

1.2.1 Git

Git is a software for versionning, put together and publishing code. It allows multiple workers on the same code, and the versionning system permits to make experimentations while keeping somewhere stable versions.
This is a really powerfull tool and it allowed us to keep tracks of a lot of things, especially checkpoints for each objective.

1.2.2 Sublime text

Sublime text is a proprietary code editor with unlimited free trial. It contains project file tree vision, autoindentation, autocompletion of code and other possibilities. It's highly extensible with python and keeps on adding functionalities with its big modules library.

1.2.3 GoSublime

GoSublime is a sublime text module for go. This modules compiles go files directly in the sublime text console and allows to have easier and quicker debugging.

1.3 Frameworks

1.3.1 Backend :

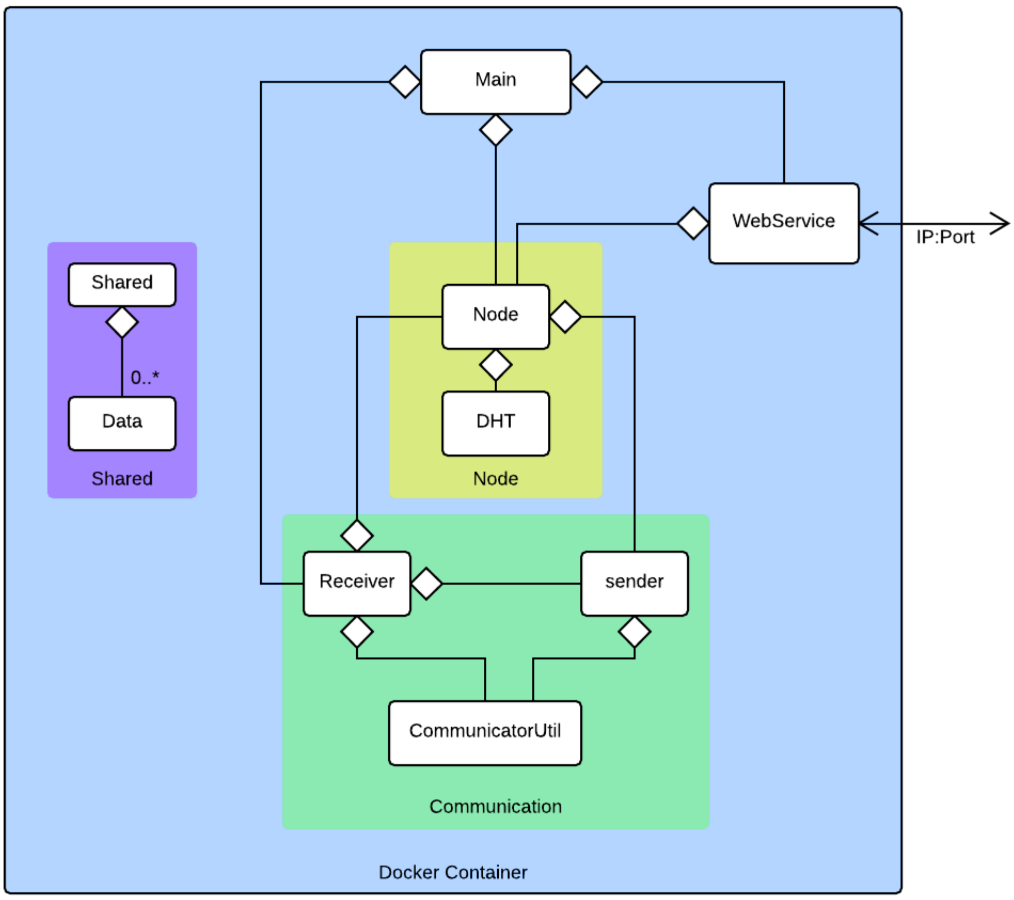
- Go-logging library : This package includes simplified use of logging messages. With different colors, types and filters to adjust quickly verbosity of the project.
- Testify library : This package is specialised in assertions, for generating automatised tests in go.
- Gorilla mux : This framework handles serving functionnalities and is able to associate a go function to an given URL. It was used to make the REST API.

1.3.2 Frontend :

- Angular js : This javascript library is made to handle two way data binding for dynamic websites. Thanks to its high coupling between html and script, this tool is really easy to use for one page website with lots of formulars and user interactions.
- D3 js : D3 is another javascript library which creates and animates dynamics graphs. It is easy to associate with angular.
- Bootstrap : Bootstrap is a ccs library for beautiful default styles. It spares times to recreate everything for each website.

2 General architecture

2.1 Node architecture

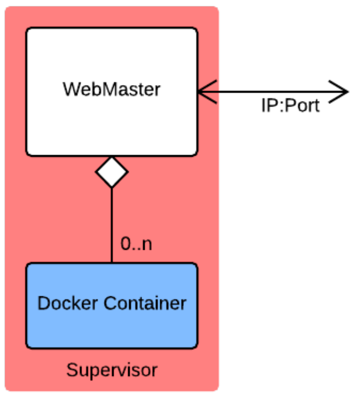


Package Description :

- Node block :
  - **Node** : Contains the intelligence of the application. It's there that the lookup are performed for example.
  - **DHT** : Given code for low level operation such as id computation and hashing, finger computation, etc.
- Communication block
  - **CommunicatorUtil** : Contains the operations (un/marshalling), structures (for messages waiting for a response) and types shared by *sender* and *receiver*.
  - **Sender** : Used to send message. Stateless.
  - **Receiver** : Handles the messages received. More intelligence than in *sender* since some incomes can be the response of previously sent message.
- Shared block
  - **shared** : Contains application-wide informations such as Id, IP/Port and data of the application. It also provides a common logger object.
- **Main** : The entry point of the application. There command line parameters are defined and everything is launched according to them.
- **WebService** : The REST interface to interact with the application. It also presents an html page for easier utilization.

All of the application remains inside an isolated docker container.

2.2 Supervisor architecture



The Supervisor purpose is to orchestrate docker container life cycle. From here, you can launch a new node, stop a running one and access to a specific node.
It provides a REST API to do those operations and also serves an html page to interact with easily.

2.3 Details on network communication

Since we didn't use the golang implementation of *RPC*, we have to handle manually the answer of messages.

When a message which need a response is send, such as *lookup* messages, we create an entry in the table of **pending lookup**.

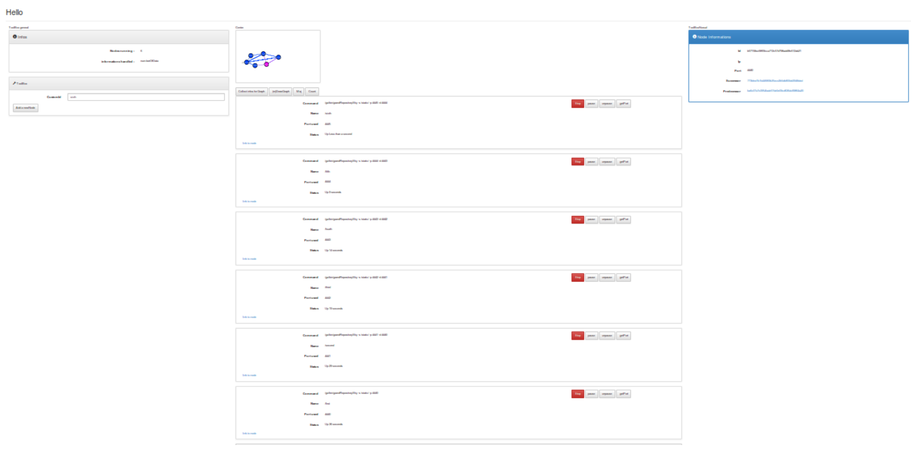
In this table are stored an **id** for this message and a **response channel** associate.

When we receive a lookup response message, we fetch from this table the id of the initial message and send **through the channel** the response.

Using channel allow us to **manage timeout** easily.

3 Fancy interface

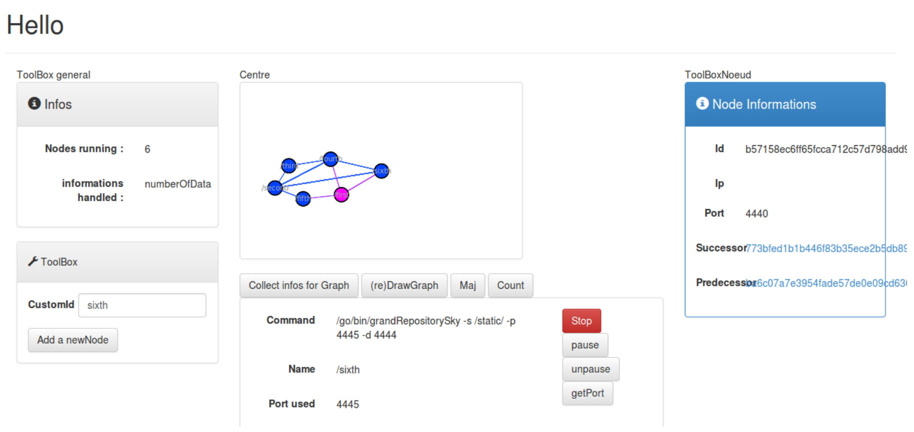
3.1 Master interface



This interface lists the nodes created with docker. For each node it is possible to pause, unpause or completely stop him. They also have a link to their own website.

A graph interface permits to visualise the nodes relations : predecessor, successor and finger tables. When selected, a node displays few informations about it. The selected node and its predecessor/successor links are in pink (yes pink). The fingers of the selected node are in lila.

3.1.1 Node creation



Hello

ToolBox general

Infos

Nodes running :	6
informations handled :	numberOfData

ToolBox

CustomId sixth

Add a newNode

Centre

Collect infos for Graph | (re)DrawGraph | Maj | Count

Command	/go/bin/grandRepositorySky -s /static/ -p 4445 -d 4444
Name	/sixth
Port used	4445

Stop | pause | unpause | getPort

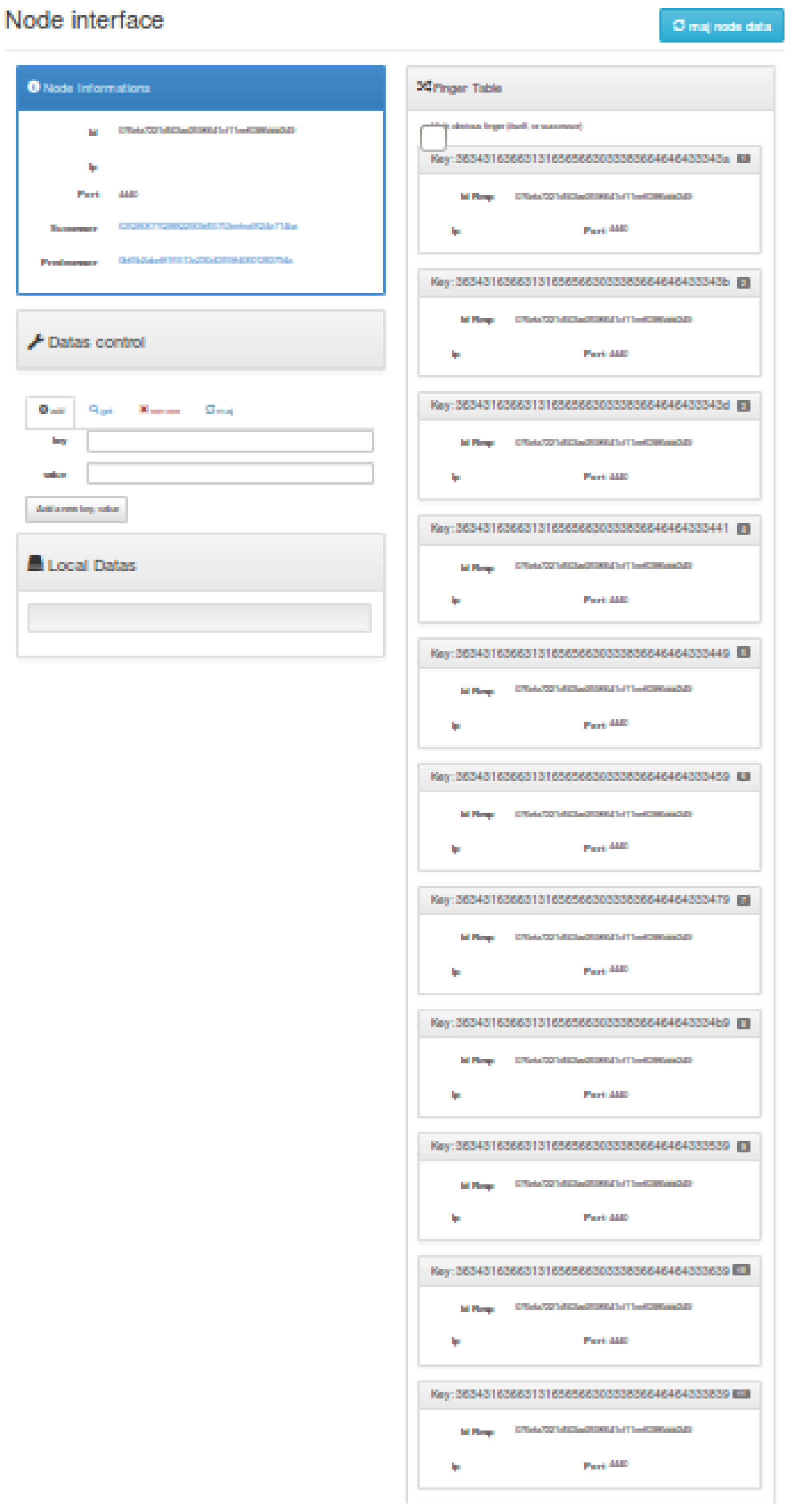
ToolBoxNoeud

Node Informations

Id	b57158ec6ff65fcca712c57d798add
Ip	
Port	4440
Successor	773bfed1b1b446f83b35ece2b5db8
Predecessor	6c07a7e3954fade57de0e09cd63

A litle formular permits to create a node docker with a personnalised surname. If no surname is specified, the server will create an automatic one. This surname is used in graph visualisation.

3.2 Individual node interface


Node interface

maj node data

Node Informations

Id	[illegible]
Ip	
Port	[illegible]
Successor	[illegible]
Predecessor	[illegible]

Datas control

add get remove maj

key

value

Add a new key, value

Local Datas

Finger Table

[illegible]

Key: 36343163663131656566303338366464643333433a

Id Resp [illegible]

Ip Port [illegible]

Key: 36343163663131656566303338366464643333433b

Id Resp [illegible]

Ip Port [illegible]

Key: 36343163663131656566303338366464643333433d

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333441

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333449

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333459

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333479

Id Resp [illegible]

Ip Port [illegible]

Key: 3634316366313165656630333836646464333334b9

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333539

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333639

Id Resp [illegible]

Ip Port [illegible]

Key: 363431636631316565663033383664646433333839

Id Resp [illegible]

Ip Port [illegible]

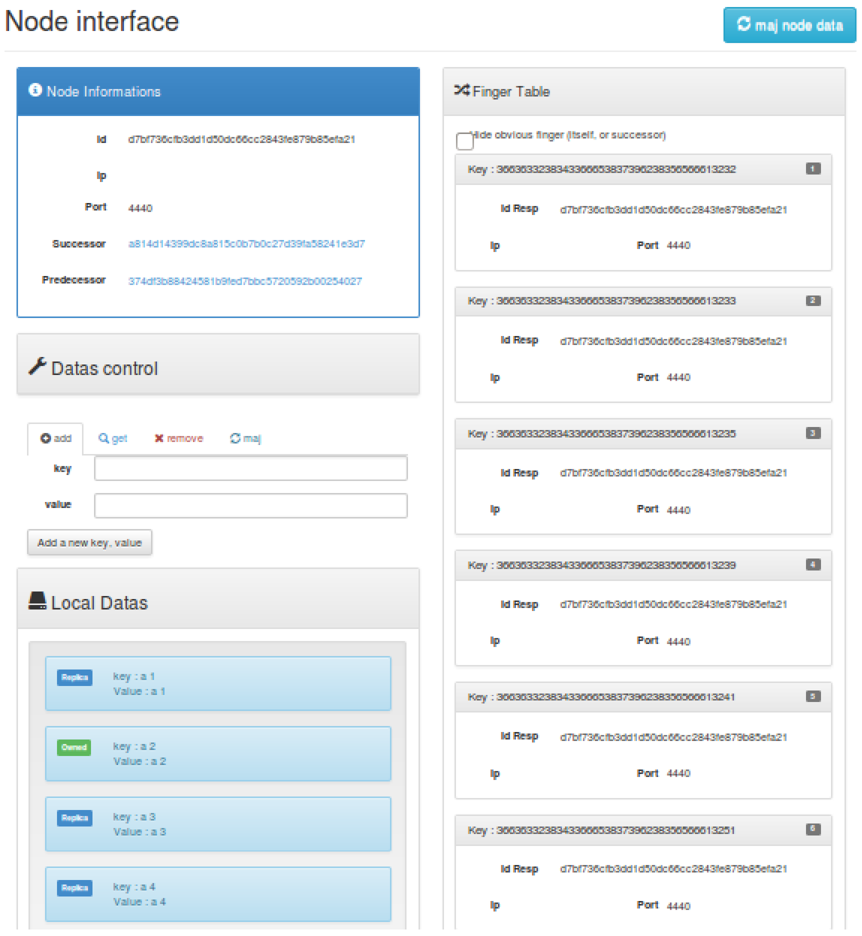
The node website is divided in three parts.
First is the node general informations : id, ip(always localhost), port, predecessor and successor.

The second part, on the left, is about the finger table of the node. Its size doesn't change, even for small rings. Since there is a lot of repetitions, a functionnality has been added to remove the obvious fingers, such as itself or the direct successor. Fingers as itself have a green label at the left size of their header, succesors are blue, non obvious are grey.

The third part is the data visualisation and manipulation. It is possible to manipulate data from all nodes. The "local datas" panel visualisation concern only the data hold by the node. Owned data have a green label, data that it replicates for its neighboors have a blue one. For more informations about data policy, see chapter 5 : handling data and consistency.

3.2.1 Data manipulation

Add



Node interface

maj node data

Node Informations

Id	d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip	
Port	4440
Successor	a814d14399dc8a815c0b7b0c27d39fa58241e3d7
Predecessor	374df3b88424581b9fed7bbc5720592b00254027

Datas control

add get remove maj

key

value

Add a new key, value

Local Datas

Replica key : a 1
Value : a 1

Owned key : a 2
Value : a 2

Replica key : a 3
Value : a 3

Replica key : a 4
Value : a 4

Finger Table

Hide obvious finger (itself, or successor)

Key : 366363332383433666538373962383565666613232 1
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 366363332383433666538373962383565666613233 2
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 366363332383433666538373962383565666613235 3
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

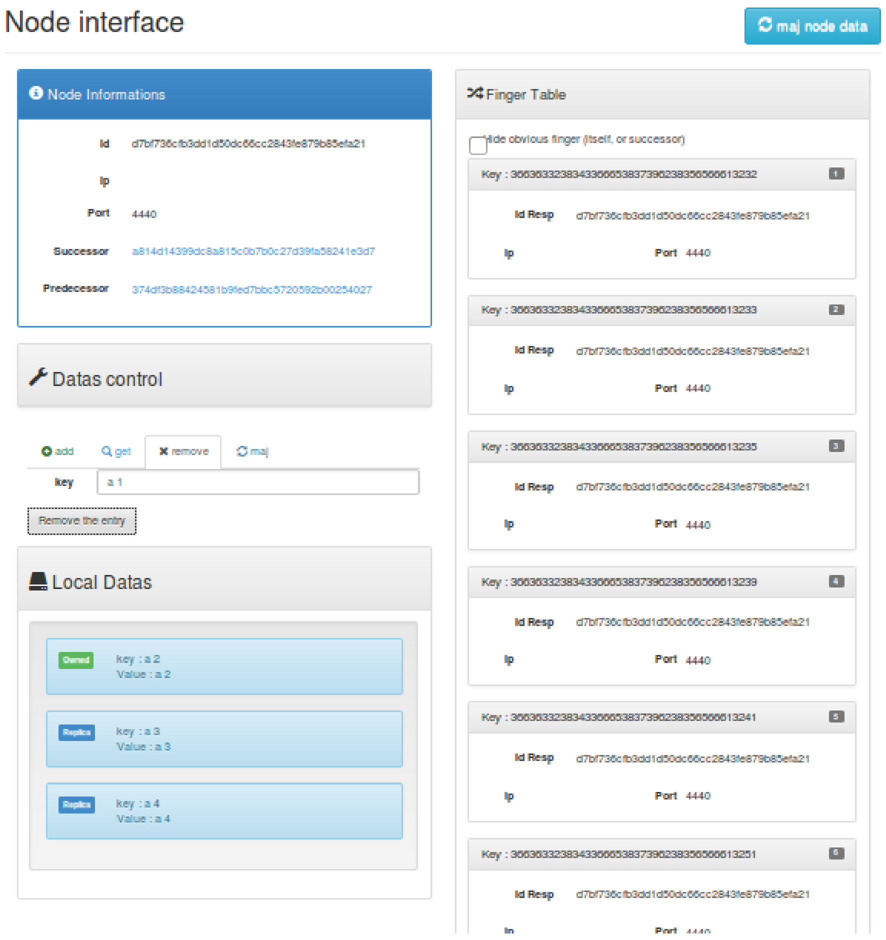
Key : 366363332383433666538373962383565666613239 4
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 366363332383433666538373962383565666613241 5
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 366363332383433666538373962383565666613251 6
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

The add function needs a key and a value. The node server will hash the key and calculate the node responsible, then he sends the create instruction to the concerned node.

Delete



Node interface

maj node data

Node Informations

Id	d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip	
Port	4440
Successor	a814d14399dc8a815c0b7b0c27d39fa58241e3d7
Predecessor	374df3b88424581b9fed7bbc5720592b00254027

Datas control

add get remove maj

key: a 1

Remove the entry

Local Datas

Owned key : a 2 Value : a 2

Replica key : a 3 Value : a 3

Replica key : a 4 Value : a 4

Finger Table

Hide obvious finger (itself, or successor)

Key : 3663633238343366653837396238356566613232 — 1
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 3663633238343366653837396238356566613233 — 2
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 3663633238343366653837396238356566613235 — 3
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

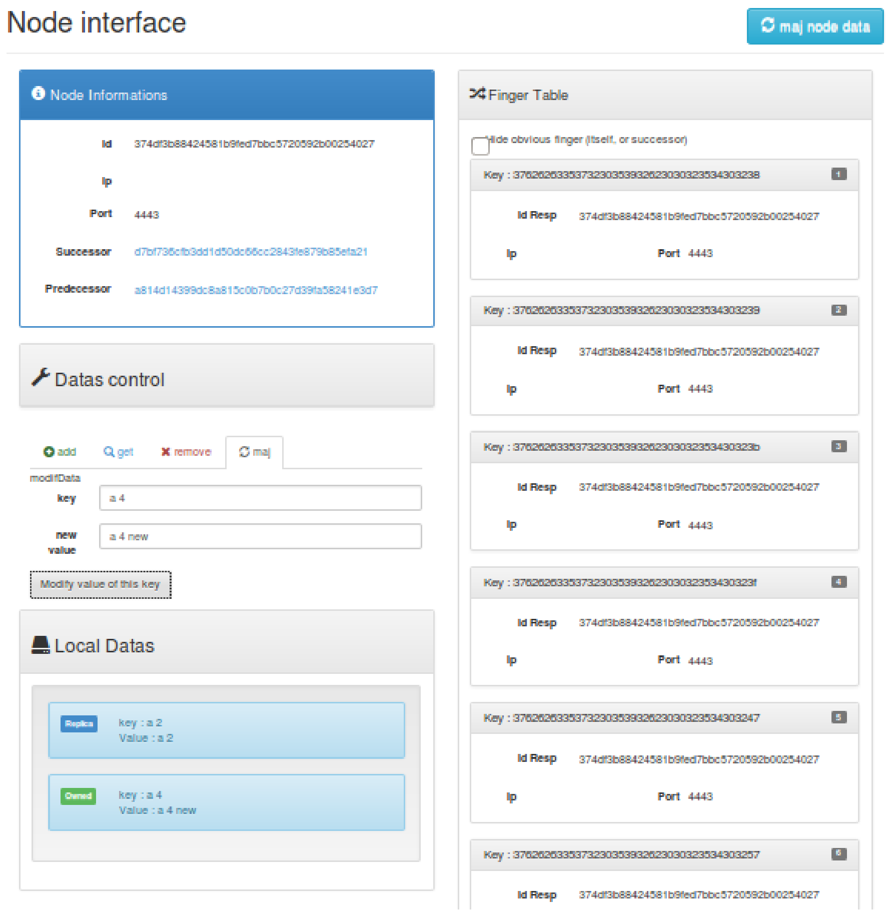
Key : 3663633238343366653837396238356566613239 — 4
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 3663633238343366653837396238356566613241 — 5
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

Key : 3663633238343366653837396238356566613251 — 6
Id Resp d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Ip Port 4440

The remove interface only needs the non hashed key, any node can receive this instruction and will relay it to the good part of the ring.

Update



Id	374df3b88424581b9fed7bbc5720592b00254027
Ip	
Port	4443
Successor	d7bf736cfb3dd1d50dc66cc2843fe879b85efa21
Predecessor	a814d14399dc8a815c0b7b0c27d39fa58241e3d7

modifData

key: a 4

new value: a 4 new

key : a 2
Value : a 2

key : a 4
Value : a 4 new

Key : 3762626335373230353932623030323534303238 — 1
Id Resp 374df3b88424581b9fed7bbc5720592b00254027
Ip Port 4443

Key : 3762626335373230353932623030323534303239 — 2
Id Resp 374df3b88424581b9fed7bbc5720592b00254027
Ip Port 4443

Key : 376262633537323035393262303032353430323b — 3
Id Resp 374df3b88424581b9fed7bbc5720592b00254027
Ip Port 4443

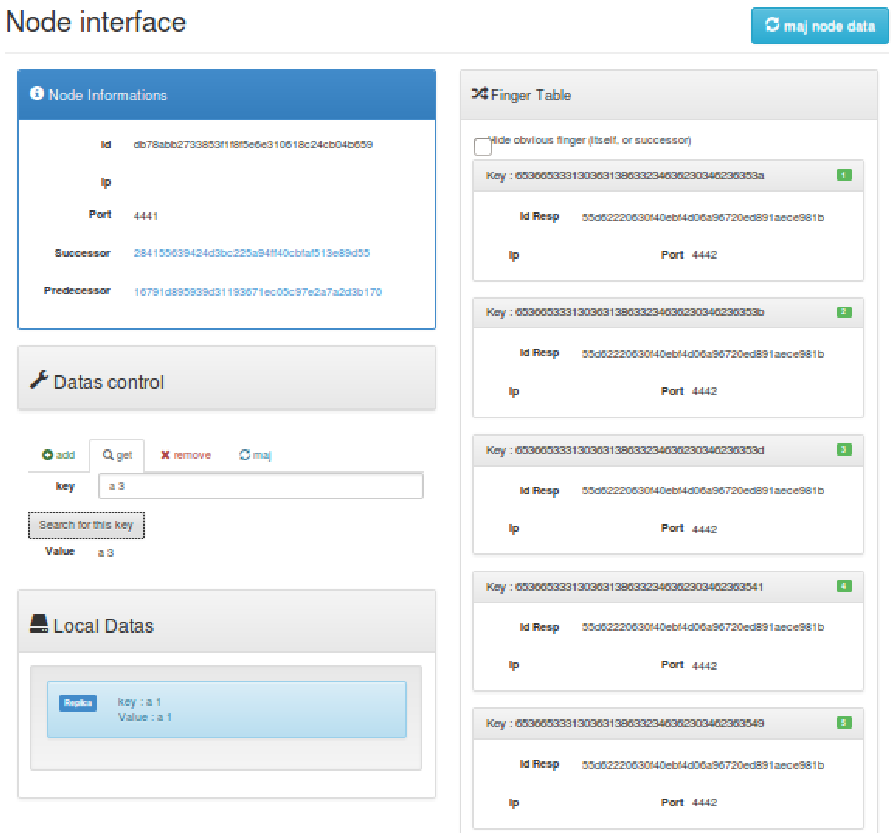
Key : 376262633537323035393262303032353430323f — 4
Id Resp 374df3b88424581b9fed7bbc5720592b00254027
Ip Port 4443

Key : 3762626335373230353932623030323534303247 — 5
Id Resp 374df3b88424581b9fed7bbc5720592b00254027
Ip Port 4443

Key : 3762626335373230353932623030323534303257 — 6
Id Resp 374df3b88424581b9fed7bbc5720592b00254027

The update interface needs the non hashed key and the new value, any node can receive this instruction and will relay it to the good part of the ring.

Get


Node interface

maj node data

Node Informations

Id	db78abb2733853f1f8f5e6e310618c24cb04b659
Ip	
Port	4441
Successor	284155639424d3bc225a94ff40cbfaf513e89d55
Predecessor	16791d895939d31193671ec05c97e2a7a2d3b170

Datas control

add get remove maj

key: a 3

Search for this key

Value: a 3

Local Datas

Replica key : a 1
Value : a 1

Finger Table

Hide obvious finger (itself, or successor)

Key : 65366533313036313863323463623034623636353a 1
Id Resp 55d62220630f40ebf4d06a96720ed891aece981b
Ip Port 4442

Key : 65366533313036313863323463623034623636353b 2
Id Resp 55d62220630f40ebf4d06a96720ed891aece981b
Ip Port 4442

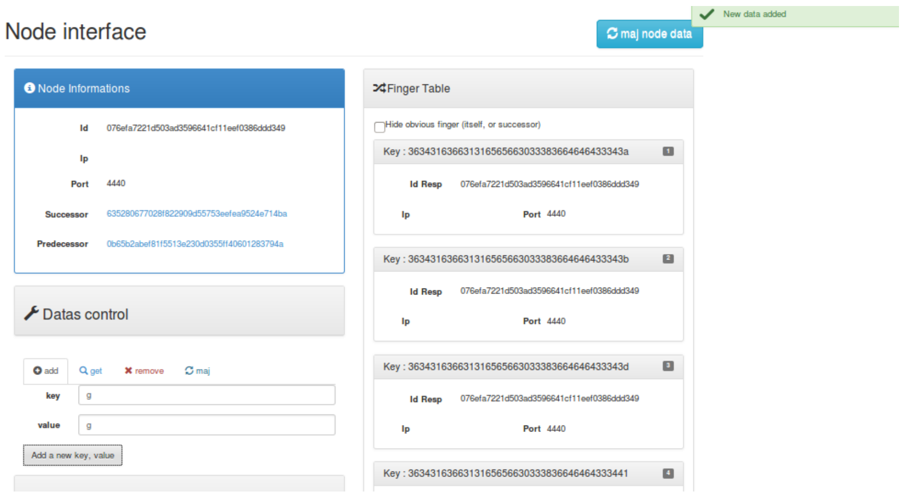
Key : 65366533313036313863323463623034623636353d 3
Id Resp 55d62220630f40ebf4d06a96720ed891aece981b
Ip Port 4442

Key : 653665333130363138633234636230346236363541 4
Id Resp 55d62220630f40ebf4d06a96720ed891aece981b
Ip Port 4442

Key : 653665333130363138633234636230346236363549 5
Id Resp 55d62220630f40ebf4d06a96720ed891aece981b
Ip Port 4442

The get interface gives the value from a non hashed key. A node is able to get a value even from an entry he doesn't possess or replicates.

3.3 Growl system


Node interface

maj node data

New data added

Node Informations

Id	076efa7221d503ad3596641cf11eef0386ddd349
Ip	
Port	4440
Successor	635280677028f822909d55753eefea9524e714ba
Predecessor	0b65b2abef81f5513e230d0355ff40601283794a

Datas control

add get remove maj

key: g

value: g

Add a new key, value

Finger Table

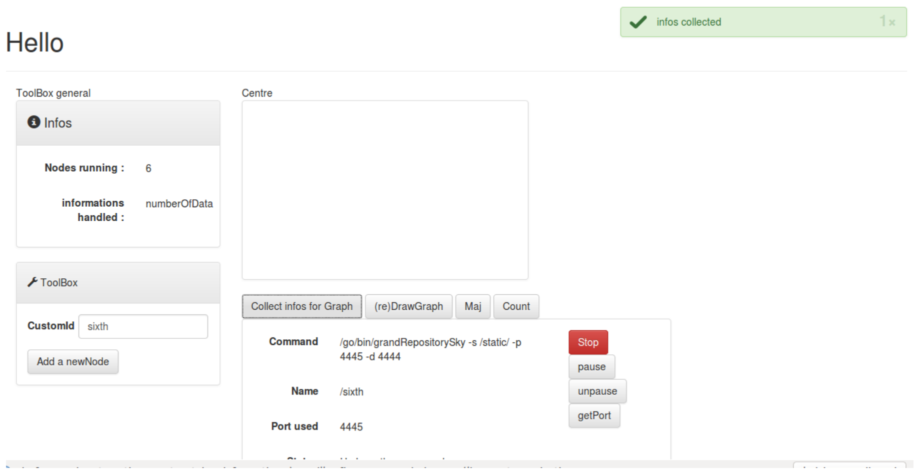
Hide obvious finger (itself, or successor)

Key : 3634316366313165656630333836646464333343a 1
Id Resp 076efa7221d503ad3596641cf11eef0386ddd349
Ip Port 4440

Key : 3634316366313165656630333836646464333343b 2
Id Resp 076efa7221d503ad3596641cf11eef0386ddd349
Ip Port 4440

Key : 3634316366313165656630333836646464333343d 3
Id Resp 076efa7221d503ad3596641cf11eef0386ddd349
Ip Port 4440

Key : 363431636631316565663033383664646433441 4



Nodes running : 6

informations handled : numberOfData

CustomId sixth

Command /go/bin/grandRepositorySky -s /static/ -p 4445 -d 4444

Name /sixth

Port used 4445

The two websites are potentially hightly asynchronous. To detect that a command has been executed, when the server answer, a growl panel pops for a few seconds.

If the answer from the server is an error, depending of the case, either a growl or a console message displays the problem.

4 Handling data and consistency

Operation on data in this system are made like in a *Tuple Space* :

- Data are **immutable**.
- Modifying a data means remove the old data and add the new one.

4.1 Replication

When a data is added to the system (to a node) it's also replicated to the node's predecessor and successor.
That replication operation occurs when a data is written but also periodically to make sure it is effectively replicated. If the initial replication fails other attempts are made.

It's the same when deleting a data.
The delete request is sent to the node which hold the data and this node relays the request to the nodes which hold replicas of its data.
Periodically, each node checks if the data it holds is owned by its successor or predecessor in order to remove the outdated data.

5 Limitations

We have to compromise between acknowledgement of request, performances and time to code.

The nodes exchange a lot of informations all the time in order to have up to date data and fingers tables. But all those communications will reduce the scalability of the system.

The fingers table of a node is not guaranteed to be up to date when routing. You can change the rate of update for fingers table but it leads to more workload for each node and for the network.

5.1 Limit cases

5.1.1 First node of the ring

The first node of the ring initialises his whole fingers table with himself. If you add a new data to the ring, the data is not replicated.

5.1.2 Only two nodes yet in ring

When there is only 2 nodes in the ring, the data of one node is replicated only once, on its predecessor.

5.1.3 Too many nodes in ring

A ring can't possess more nodes than the number of differents id possible. But scalability problems will occur before this number is achieved anyway.

6 What can be improved

6.1 Network

The use of docker is hard for dynamically link containers with each other. Currently, each docker container uses the host internet interface (and not an isolated interface, by default in Docker).

6.2 Website

The website could still be perfected. Interface could be improved with adding new functionalities, especially in data visualisation like the finger table of a node. The size of the id makes difficult the identification of nodes.

Moreover some functionalities are quite slow for the moment.

By spending more time on it, it should be possible to have a better use of the technology and the communications between website and server(s).

7 What should be redone

7.1 Architecture

We are not satisfied with the general architecture. Especially the communication part. The intelligence is spread between *node* and *receiver* and the relations between those blocks are not clear.

7.2 Use golang RPC capabilities

It could really simplify the architecture and the boilerplate code of sender and receiver. But underline, it uses tcp so we abandoned the full udp idea.

8 Link to the code

Here on gitHub : https://github.com/tgermain/grandRepositorySky

9 General conclusion on technologies

Golang is a good choice for this project.
It come with :

- RPC capability
- Web server
- Network stack
- Testing framework
- Package manager
- WebSockets
- And a lot more of package

The language is not hard to learn and understand. The ecosystem is a bit sparse but it's enough for what we have to do.

Docker works for this project but should maybe be replaced by another solution, since it's not really adapted to handle dynamic network communications.

Git was really helpful.

The frontend technologies as angularjs, d3js and bootstrap worked well, but a better use of them could improve the performances of the interface.

10 General conclusion on the project

The whole project was really interesting but the point of some requirements was not very clear, especially :
Why doing the algorithm in one objective and the communication on another ?
We think they are both part of the core of this project and do it in 2 objectives mean that you do some work twice.

Some of the solutions we implemented work for this project but may cause problems in real situation application or with huge scalability needs, such as the limits of docker technology or the fact that we didn't try to test the result on others operating systems that the ones we developped on.

11 Helpful resources

- Golang documentation on concurrency
- Commented snippets of code
- Golang examples