LAB 08 DHT, from simulation to web-based real implementation!!!

Outline

1. Explain solution of last lab –> *advanced DHT with JOIN/LEAVE protocol + FingerTable*
2. **StepByStep implementation of a web-based DHT with Flask**

STEPS 4 a WEB-BASED DHT IMPLEMENTATION

- Define a DHTnode web-API with Flask
- Write down a "bootloader" script responsible of deploying single nodes
  - HINT: start from https://github.com/pipelinedb/pipelinedht
  - HINT: define a Node/Process class

A Process object stores all relevant info about a running DHT process, namely: host, port, name, pred, succ

Careful! pred and succ should be just string in this format host:port:name, not the complete description (in form of Process objects) of pred and succ, otherwise we would store the complete double-linked list in each of our node. In other words, the depth of the knowledge of successor and predecessor nodes must be only 1

"*Installing an app attribute at boot...* " In the nodeAPI file

```
def start_app(host, port, name, pred=None, succ=None):
    app.proc = Process(host, port, name, pred, succ)
    app.run(host=host, port=port)
```

In the bootloader (we need to add possibility of passing pred and succ attributes for a new App/Process)

```
start_app(host=args.host, port=args.port, name=args.name, pred=args.pred, succ=args.succ)
```

1st Checkpoint

- Use the bootloader to deploy one node (bound to localhost on some given port)
- Test dummy responses using POSTMAN

```
snap install postman
```

- Implement “node-info” method, to get all information needed to contact one node
- Implement successor and predecessor GET/PUT methods

2nd Checkpoint

Test with POSTMAN the 3 queries work as expected

Next steps in next LAB :)