INFO8002 Large-scale data systems
Exercise session III

Instructions

For the main questions, be ready to:

- Describe the architecture of your solution and its operations.
- Define all its components and their interactions.
- Motivate your design decisions.
- Make diagrams whenever necessary.
- State clearly your choices and assumptions

For feedback, feel free to e-mail joeri.hermans@doct.uliege.be

Question 1.

You are tasked to implement a distributed and fault-tolerant file system. This implies that if one or more processes fail, the normal operation of the file-system and its interface should be guaranteed. We impose the constraint that a distributed hash table needs to be used to store the metadata. The discussion of the actual implementation is up to you. **Data should NOT be copied to all processes.** The implementation of the following user interfaces should be discussed, and a execution needs to be demonstrated by means of a sketch:

put(path, bytes)

> Puts the bytes on the specified path. This method will raise an error when a file is present on the specified path.

copy(source_path, destination_path)

> This will copy the file at the specified source path, to the specified destination path. If the source path does not exists, or the destination path is not available for some reason an error should be raised.

get(path)

> Returns all bytes of the file present at the specified path. If the path does not exist, an error should be raised.

exists(path)

Checks if a file exists at the specified path. Returns true if a file is present at the specified location, false otherwise.

Guidelines

It is recommended to design your system in the following steps:

1. What is the distributed system model you are assuming?
2. Identify all actors in your system. Provide a short description.
3. Sketch the (meta)data storage, i.e., how are you going to store the metadata in the DHT and the data? An important consideration here is how you are going to handle the files. What if the system only needs to store small files, or only large ones?
4. Sketch the implementation of the interfaces specified above while not considering errors.
5. Ask yourself how many **concurrent** errors your system will be able to tolerate and under what conditions? E.g.; In a centralized setting with a master node; my system will be able to tolerate 3 concurrent failures whenever there are 4 datanodes in the network. We can only tolerate 1 failure of a masternode. The state of the masternode is replicated on a backup master through(for instance) a replicated state machine.
6. What happens if a (data)node (containing the data of interest) leaves the network? Sketch an execution, how is the persistence of the data guaranteed?
7. What happens if a (data)node joins the network? Sketch an execution.

Question 2.

Discuss the fault-tolerance in Kademlia and Chord. How does fault-tolerance and persistence of the data and meta-data differ in both implementations? Should one be preferred over the other, and why?