Challenge 4: Distributed Hash Table

Your task is to implement a memcached client library that automatically spreads read and write requests across available memcache servers. Use zookeeper to coordinate enable the client library discover the active memcache servers. The health of each memcached server must be watched by a
watchdog process that interacts with ZooKeeper to announce the daemon's liveness.

Write a client that uses the client library and takes as input a commands file of the following format:
GET key
PUT key,value
SLEEP value

- key,value are string literals (no punctuation marks), and the value to sleep is the number of ms.
- the client will execute the associated command. The client will print, in the case of get, the value read from the server (if any).
-

The goal of the client library is to spread data in such a way as to increase the cache hit rate for get requests, while using a single replica per key/value pair. The algorithm to distribute keys to servers must maximize the chance that the same key goes to the same server, even under server churn.
The watchdog is an application that monitors the state of the memcached daemon (whether it is running or not). It must not attempt to restart the daemon in case it crashes.

Implementation requirements/hints

- Your client **MUST** conform to the input file's format. We will be testing your implementation using or own input files while simulating different kinds of failures.
-
- A memcached server needs to be removed from the pool of available servers if:
  - The daemon crashes.
  - The watchdog crashes.
  - The machine loses network connectivity.
- Restarting a crashed daemon, restoring network connectivity etc. should also cause the memcached server to be added back to the pool of available servers. (Hint: ephemeral znodes are not automatically recreated.)
- ZooKeeper cannot be used to store keys or values.
- The client must be stateless with regard to key placement. In other words, it must not "remember" where keys are placed across multiple GET/PUT operations.
- You can use jna-libmemcached (https://github.com/nowelium/jna-libmemcached)
- You can use Mininet to test your implementation. Suggested test scenarios:
  - Kill a memcached server. Restart it after a while.
  - Ditto for its corresponding watchdog.
  - Disable the server's virtual network interface. Re-enable it after a while.