Distributed Key-Value Store

MEI Distributed Systems
Project – Stage 1

2012/2013

1 General Conditions

- Each group is composed by 2 to 4 students registered in the e-Learning site.
- This stage is concluded by delivering a short report (8 pages), and the source code in the e-Learning site.
- The deadline for submitting the report and the source code is 2011/12/13.
- The work is presented and discussed in 2011/12/17.

2 Goal

The goal of the project is to design, implement, and evaluate a distributed data storage with a client/server architecture. Each data item must be stored persistently in disk in one of several remote servers. A client uses a library to update and query the data, shielding the client from its actual location. This should allow a client to associate keys (`String`) to values (`byte[]`) , much as a `Map` does. The system should ensure that the amount of data stored in each server is approximately the same. The set of servers is known beforehand by other servers and clients. Both the client and the server should be written in Java, using sockets, threads, and files.

3 Client API

The client should implement the following interface:

```
public interface KeyValueStore {
    public void put(String key, byte[] value);
    public byte[] get(String key);

    public void putAll(<Map<String,byte[]>> pairs);
    public Map<String,byte[]> getAll(Collection<String> keys);
}
```

4 Evaluation Criteria

- Full implementation of the required interface and functionality.
- Simplicity of the proposed solution: It is more important to identify and document issues in the report than trying to solve them.
- Critical assessment of the proposed solution: The report should contain performance measurements and discussion of limitations.

5 Discussion Topics

The main goal of the report is to perform a critical assessment of the proposed solution, not a bland comment on design and implementation. In particular, this project should raise the awareness for challenges involved in topics such as:

- Defining the semantics of `putAll` and `getAll` operations with concurrent clients and servers.
- Re-configuring the set of servers, to cope with additional load and server failure.
- Scaling up the performance with the amount of requests.
- Scaling up the performance with the amount of data stored.