Distributed Systems

Distributed File System

Ryan Kane – 1332 5208

Computer Science and Business

<https://github.com/kaner94/distroDFS>

Table of Contents

INTRODUCTION 3

Authentication System 4

# INTRODUCTION

In this report I will be detailing how I completed the assignment to create a Distributed File System in Haskell. From attending lectures and through the advice of our lecturer it was highly recommended to split the system into a number of sub-systems. As such, for each sub-system I will explain my idea for how the system would work, how I implemented the system and how the system works. If I feel any system can be improved I will also include that information.

The system is made up of the following sub-systems:

1. Authentication System
2. Caching System
3. File System
4. Directory System

In the following pages you will find these sub-systems described as explained previously.

## Authentication System

The authentication system provides a large range of services for this system ranging from Signing Up to Retrieving a Token.

**Add User**

The user is most likely going to use this system first when they wish to sign up. This occurs by using the ‘addUser’ method. The user will pass a ‘name’ and ‘password’ when calling this method. These will both be used in the addUser method to create an instance of the ‘User’ datatype. This instance of User will then be posted to the MongoDB by calling the ‘postUser’ method. The password will be encrypted before it is posted to the database. All data must be converted to BSON before it is posted to the MongoDB to ensure it works correctly.

**Encryption and Decryption**

The encryption in this system is based off a simple Caesar cipher. The encryption function relies on the Token. This is simply a datatype that contains an integer which is determined by the Authentication system. To encrypt a string, the system uses the ‘Map’ function combined with ‘Ord’ to convert each character in the string to a hexadecimal value. ‘Map’ is then used again to augment each hexadecimal value by the predetermined value of the Token. The Map function is used one last time with the ‘Chr’ function to convert the array of hexadecimal values back into characters.

Decryption is performed by performing very similar steps. The encrypted string is converted to hexadecimal values, the Token is reapplied to the values and then the hexadecimal is converted back into a string.

**Get Token**

Before a user can access files, they are required to retrieve a token from the Authentication system. This is performed by using the ‘getToken’ method.

When making this call, the user is required to pass a Username and Password. The system will check the MongoDB to ensure that the username is in the system and the password does indeed match. If this check is successful, a token will be returned to the user. Otherwise, this request will fail.