**Classes and Functionality Explanation For DHT**

**DS Fall 2023 Project**

**BigHex**

This class is designed to help handle very large numbers that can be possibly used in this project. Basically, it is used to store the hash values of the machines and files in the DHT, in the form of C++ standard strings. Normal C++ integer datatypes, even unsigned long long int, are limited by current computer architectures, with a limited ability to handle numbers up till a maximum of 64 bits. As the DHT hash values can range from 4 bits to 160 bits, these datatypes fail, hence the need for this class. It contains all the necessary arithmetic and logical operators required in this project. Although the numbers are displayed in hexadecimal format, the addition and modulo operations are carried out by first converting them to binary. Comparison operations are done in hexadecimal though.

**File**

The objects of this class simply store the hash values and paths of each individual file stored in the DHT. These objects constitute the actual elements that are stored in the B-trees. The actual files are stored on the computers hard disk, at the paths specified by the B-tree file object. It also contains a pointer to help chain other file objects to it in the case of collisions in hash values.

**B\_tree**

As the name suggests, it implements the B-trees of the project. Unlike conventional B-trees used in industry, these are stored and modified entirely in the main memory, instead of read and write operations from secondary storage. The class makes B-trees according to the order specified by the user. It contains a simple default parametrized constructor, search, insert, delete, and print functions, along with functions that transfer the contents of one B-tree to the other.

**Routing\_Table**

This class implements a simple finger or routing table as a doubly linked list, containing the IDs and addresses of other machines in the DHT. It helps improve file search complexity from O(n) to O(log n) by providing calculated shortcuts to other machines. The number of entries is equal to the identifier space, or hash bits, given by the user. Their initialized at the time of insertion of each machine, with all of them being updated after the insertion and removal of a machine.

**Machine**

A class that represents real computer machines. Each machine has a unique name, hash ID, and folder on the computer. The files are stored in the folders, with their file objects being stored in the B-tree of each machine. Each machine also contains its own routing table, to aid in searching in the chord DHT. IDs can be autogenerated or be given by the user.

**Ring**

It is the ring network of the DHT, which is implemented as circular linked list. It stores all the machines of the network. The ring can be traversed in a sequential manner, as well as with the use of finger table entries located at each machine. It has the functionality for the addition, removal, and searching of machines in the ring, as well as printing all of its elements, along with initialising and updating all of them routing tables of the machines.

**DHT**

This is the main user interface class. It allows the user to input the number of machines, the identifier space, and the order of the B-trees. It contains the starting functions for all the various operations of the system. It allows the user to add machines, remove machines, add files, remove files, print all the machines, print the B-tree of any machine, and print the routing table of any machine. It also creates a main folder, in which all the machine sub-folders are created, which is deleted at the end of the DHT’s lifetime.

**General Functionality**

A simple IPFS function has to be called inside the main C++ function of the current user program. Inside the IPFS function is a simple DHT object and while loop driven menu which is used to create the menu user interface. From here the user has to enter important details such as the identifier space, the order of the B-trees, and the number of machines in the DHT. After this they are given all of the functions of the DHT to choose from, with extra options to clear the console window, or exit the program. The addition of any machine or file also requires the user to enter the required information, such as the name of the machine and the path of the file. Deletion requires entering the specific hash ID. File operations also require the user the specify the starting machine of their search. The deletion of any file or machine objects also results in the deletion of the respective secondary storage file or machine folder. Furthermore, the addition of any new machine to the system copies the file, with its original name and contents into the respective machine folder, without deleting the original file from the user’s computers. The initial path also needs to be an absolute path from its original disk, while internal paths are relative to the Visual Studio project directory. Routing tables are updated after each machine insertion and removal operation. The deletion of any machine also makes sure that its files are not lost, as they are passed to the correct successor in the ring, as well as the correct machine folder.