# **Interplanetary Filing System (IPFS)**

Group Members:

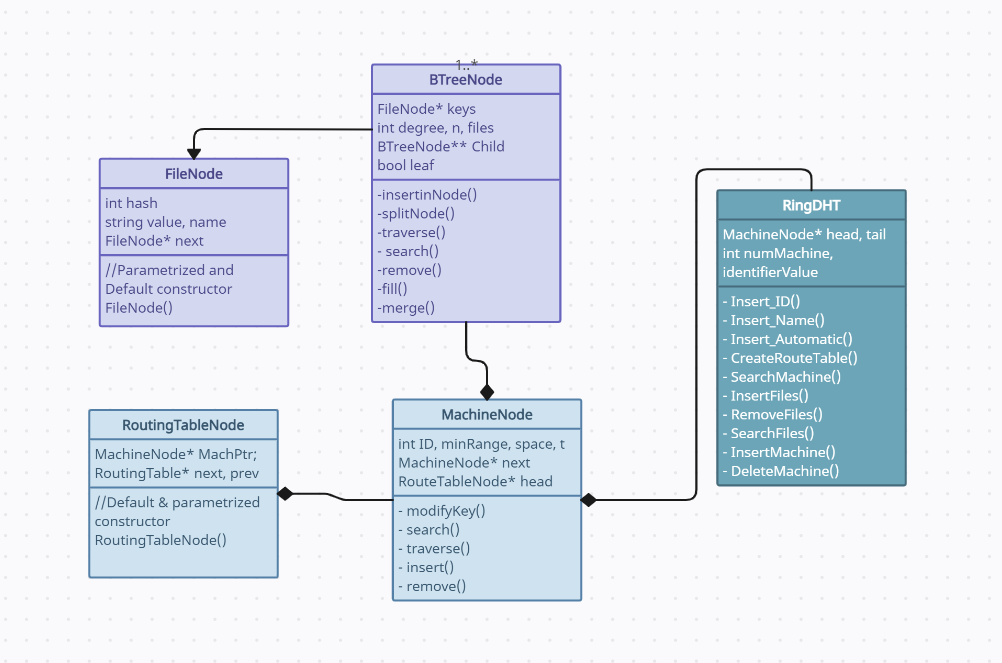
* Amna Aamir 22i-1142
* Hadiya Tanveer 22i-1113
* Sahrish Mustafa 22i-0977

Section A

## ***Introduction***

IPFS allows file storage across a network of machines. This particular IPFS is implemented in the form of a distributed hash table, which allows file access and storage. Every file is addressed using a hash calculated using its content. This hash lies between the identifier space, which is specified by the user. The hash value decides the machine onto which the file must be stored, as the machines, too, have a unique hash value. The following Class Diagram expresses the system.

### ***Class Diagram***



Therefore, the IPFS we implemented has the following components:

1. The Machine Node
2. The Routing Table Node
3. The B-Tree Node
4. The File Node
5. The Ring DHT
6. The Hash Function (SHA-1)

### ***Machine Node***

This forms the Circular Singly Linked List. It is identified by a unique ID, which is calculated using the Hash Function. As mentioned previously, the machines store files. This storage is implemented in the form of a B-Tree, a root of which lies in each Machine Node. Additionally, the machine node also includes a number which specifies the range of hash IDs it is responsible for. Each Machine Node also consists of a routing table, which is implemented in the form of a doubly linked list. Therefore, the root to this doubly linked list will be found in the Machine Node.

Machine Nodes are found using a searching algorithm designed specifically for DHTs. The routing table allows each node to have a reference to *N* more nodes ahead of it in the linked list, where *N* is the number of bits specified by the user. This searching algorithm had been provided to us beforehand in structured English.

### ***The Routing Table Node***

This forms the Routing Table doubly linked list. It contains a reference to the machine node it points to.

### ***The B-Tree Node***

This is what forms each B-Tree in a machine Node. It contains an array of keys, which are file nodes, and an array of pointers B-Tree type, to point to its children. The degree of this B-Tree is specified by the user.

### ***The File Node***

Each File Node corresponds to a unique file, as it has a unique hash and a path, denoted by “value”.

### ***The Hash Function (SHA-1)***

This function is used to calculate the hash for the machine name (insertion with machine name) or for the name of files to be inserted. By using a widely available function SHA-1, the hash of the string was first calculated and later on converted into hexadecimal. Using this hexadecimal value, depending on the identifier value the hexadecimal number was converted to simple decimal value. After that, in order to avoid large values, the converted integer was masked with the identifier space and thus, that number of bits were extracted.

### ***The Ring DHT***

Ring DHT is the main class of this whole system which brings together all the functionalities of the interplanetary file system (IPFS). It is used to first specify the identifier value and the number of machines (these are must values). After that, it is used for inserting machine nodes with user-given identification numbers, with machine name, or automatically. Then, it consists of creating a routing table for each machine node and a search function that is used to find the specified machine starting from any position. Moreover, it contains all the functions to provide functionality for files (insertion, deletion, search, traverse). Lastly, it contains insert and removal of machines on the fly without disrupting the functionality of the whole Ring DHT.

The following are the main functionalities performed by the ring DHT:

1. File Insertion
2. File Deletion
3. Machine Insertion
4. Machine Deletion

File related functionalities require two search algorithms; searching the circular linked list of machines, and searching through the B-Tree once the respective machine has been found.

Machine Insertion means that a new machine is inserted into the Circular Linked List. For this, the routing table of all nodes in the Linked List has to be updated, as well as the B-Tree files- the hash ID range that the succeeding node is responsible for has to be divided. In our implementation, all routing tables are deleted, and then remade after insertion. The file nodes required in the newly inserted node’s B-Tree are deleted from the succeeding node’s B-Tree, and copied into the new node, one by one.

Machine Deletion mirrors the same process. Once a machine is deleted, the hash IDs it is responsible for are added into the succeeding nodes, and the B-Tree of this node is broken down, and files are inserted into the succeeding machine node’s B-Tree.