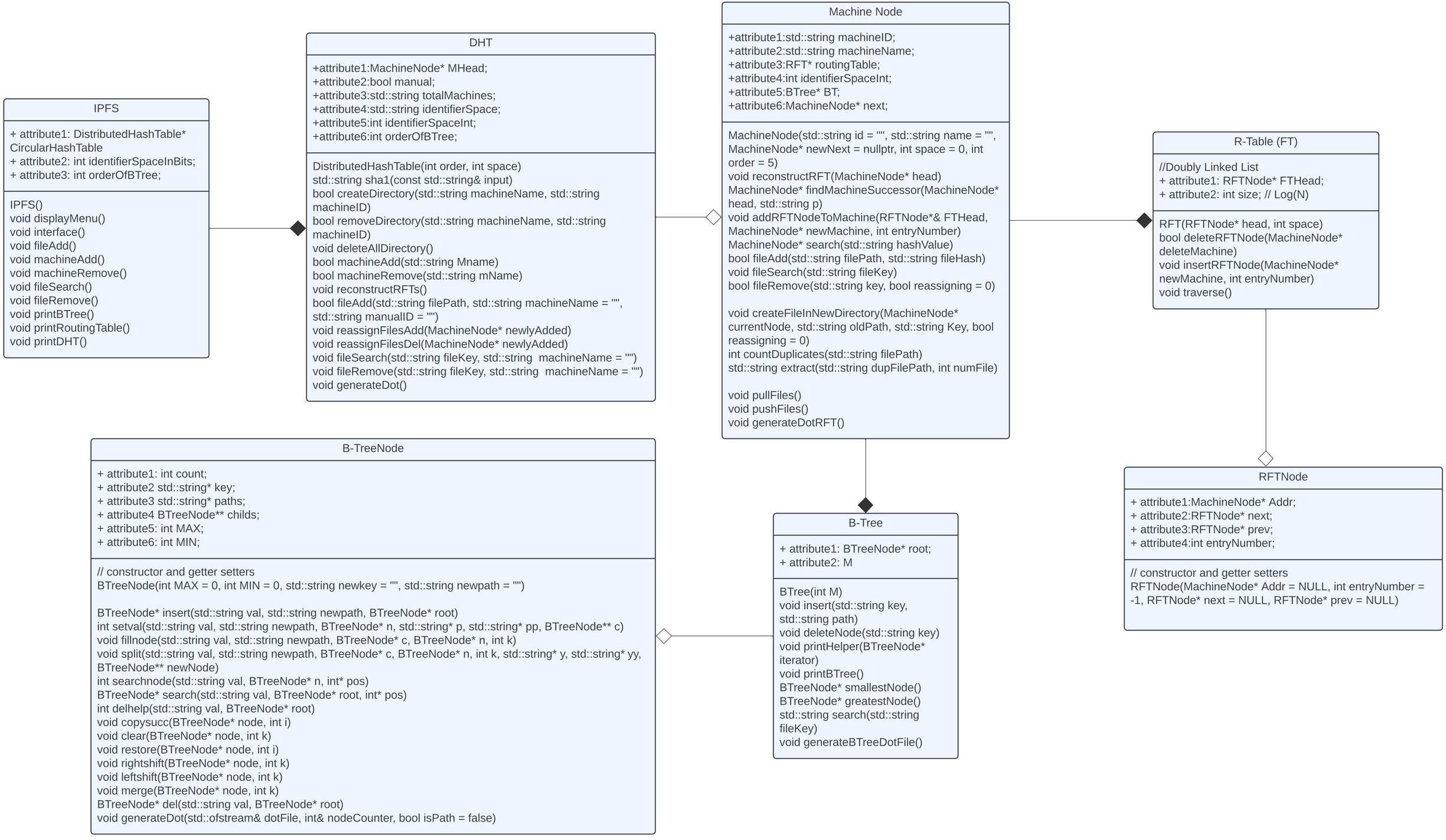
IPFS
+ attribute1: DistributedHashTable* CircularHashTable
+ attribute2: int identifierSpaceInBits;
+ attribute3: int orderOfBTree;
IPFS()
void displayMenu()
void interface()
void fileAdd()
void machineAdd()
void machineRemove()
void fileSearch()
void fileRemove()
void printBTree()
void printRoutingTable()
void printDHT()
DHT
+attribute1:MachineNode* MHead;
+attribute2:bool manual;
+attribute3:std::string totalMachines;
+attribute4:std::string identifierSpace;
+attribute5:int identifierSpaceInt;
+attribute6:int orderOfBTree;
DistributedHashTable(int order, int space)
std::string sha1(const std::string& input)
bool createDirectory(std::string machineName, std::string machineID)
bool removeDirectory(std::string machineName, std::string machineID)
void deleteAllDirectory()
bool machineAdd(std::string Mname)
bool machineRemove(std::string mName)
void reconstructRFTs()
bool fileAdd(std::string filePath, std::string machineName = "", std::string manualID = "")
void reassignFilesAdd(MachineNode* newlyAdded)
void reassignFilesDel(MachineNode* newlyAdded)
void fileSearch(std::string fileKey, std::string machineName = "")
void fileRemove(std::string fileKey, std::string machineName = "")
void generateDot()
Machine Node
+attribute1:std::string machineID;
+attribute2:std::string machineName;
+attribute3:RFT* routingTable;
+attribute4:int identifierSpaceInt;
+attribute5:BTree* BT;
+attribute6:MachineNode* next;
MachineNode(std::string id = "", std::string name = "", MachineNode* newNext = nullptr, int space = 0, int order = 5)
void reconstructRFT(MachineNode* head)
MachineNode* findMachineSuccessor(MachineNode* head, std::string p)
void addRFTNodeToMachine(RFTNode*& FTHead, MachineNode* newMachine, int entryNumber)
MachineNode* search(std::string hashValue)
bool fileAdd(std::string filePath, std::string fileHash)
void fileSearch(std::string fileKey)
bool fileRemove(std::string key, bool reassigning = 0)
void createFileInNewDirectory(MachineNode* currentNode, std::string oldPath, std::string Key, bool reassigning = 0)
int countDuplicates(std::string filePath)
std::string extract(std::string dupFilePath, int numFile)
void pullFiles()
void pushFiles()
void generateDotRFT()
R-Table (FT)
//Doubly Linked List
+ attribute1: RFTNode* FTHead;
+ attribute2: int size; // Log(N)
RFT(RFTNode* head, int space)
bool deleteRFTNode(MachineNode* deleteMachine)
void insertRFTNode(MachineNode* newMachine, int entryNumber)
void traverse()
RFTNode
+ attribute1:MachineNode* Addr;
+ attribute2:RFTNode* next;
+ attribute3:RFTNode* prev;
+ attribute4:int entryNumber;
// constructor and getter setters
RFTNode(MachineNode* Addr = NULL, int entryNumber = -1, RFTNode* next = NULL, RFTNode* prev = NULL)
B-Tree
+ attribute1: BTreeNode* root;
+ attribute2: M
BTree(int M)
void insert(std::string key, std::string path)
void deleteNode(std::string key)
void printHelper(BTreeNode* iterator)
void printBTree()
BTreeNode* smallestNode()
BTreeNode* greatestNode()
std::string search(std::string fileKey)
void generateBTreeDotFile()
B-TreeNode
+ attribute1: int count;
+ attribute2 std::string* key;
+ attribute3 std::string* paths;
+ attribute4 BTreeNode** childs;
+ attribute5: int MAX;
+ attribute6: int MIN;
// constructor and getter setters
BTreeNode(int MAX = 0, int MIN = 0, std::string newkey = "", std::string newpath = "")
BTreeNode* insert(std::string val, std::string newpath, BTreeNode* root)
int setval(std::string val, std::string newpath, BTreeNode* n, std::string* p, std::string* pp, BTreeNode** c)
void fillnode(std::string val, std::string newpath, BTreeNode* c, BTreeNode* n, int k)
void split(std::string val, std::string newpath, BTreeNode* c, BTreeNode* n, int k, std::string* y, std::string* yy, BTreeNode** newNode)
int searchnode(std::string val, BTreeNode* n, int* pos)
BTreeNode* search(std::string val, BTreeNode* root, int* pos)
int delhelp(std::string val, BTreeNode* root)
void copysucc(BTreeNode* node, int i)
void clear(BTreeNode* node, int k)
void restore(BTreeNode* node, int i)
void rightshift(BTreeNode* node, int k)
void leftshift(BTreeNode* node, int k)
void merge(BTreeNode* node, int k)
BTreeNode* del(std::string val, BTreeNode* root)
void generateDot(std::ofstream& dotFile, int& nodeCounter, bool isPath = false)