System Design

Team member: Jia Li (li001011)

Overview
This Book Finder System uses a distributed hash table (DHT) based on the Chord protocol. Using this system, the client can set and get book's information (title and genre for simplicity) using the DHT.
Assumptions:
• The system is not persistent in this project.
• We do not deal with node failures or nodes leaving the DHT after they've joined.

There are 3 components in the system:

Client
A client can set book titles and genres or get a genre with a book title from the system. It connects to the SuperNode and get a random node from it, then send requests to this random node.
There are three usages:
Usage: set:<title>:<genre>[:-withlogs]
Usage: set:<filename>[:-withlogs]
Usage: get:<title>[:-withlogs]

Node
Nodes are multi-threaded servers where all the data (titles and genres) is saved, finger table is calculated and task(get/set) is processed.
Node interfaces including:

```
void setGenre(1: string title, 2: string genre, 3: bool withLogs),
// Set genre along the DHT, the node forwards the request recursively if it is not
responsible for the book title. The forward information is printed at every node
visited if –withLogs is set.

string getGenre(1: string title, 2: bool withLogs),
// Gen genre along the DHT, the node forwards the request recursively if it is not
responsible for the book title. The forward information is printed at every node
visited if –withLogs is set.

void updateDHT(1: TableItem nodeInfo, 2: i32 idx),
// Update finger table along the predecessors when a new node join.

void printDHT(),
// Print information of DHT for debugging.

TableItem getSucc(),
// Get the successor of current node.

TableItem getSuccOf(1: i64 nodeID),
// Get the successor of a node(or key) ID.

TableItem getPred(),
// Get the predecessor of current node.

TableItem getPredOf(1: i64 nodeID),
// Get the predecessor of a node(or key) ID.

void setPred(1: TableItem pred),
```

```
// This is used to update predessor of current node.

TableItem getClosestPredFinger(1: i64 nodeID),
// This is a helper function to get the closest predessor of a node ID in current
finger table.

map<string, string> removeDataBeforeNode(1: i64 nodeID),
// This is a helper function used to remove and return data before a ID, it is
needed when a new node join and some data need to be moved to this new node.
```

SuperNode

SuperNode is a supervisor node and only stores all ips and ports of nodes in DHT,
ID collisions is avoided by having SuperNode generate all the node IDs. It is the initial point of contact for the nodes to join the DHT. A client also needs to contact to the SuperNode to get node information to contact.

SuperNode interfaces including:

```
NodeInfo join(1: string ip, 2: i32 port),
// Used for a node to join DHT, the supernode will assign a node ID for it and also
save ip and port of this node. Both ID and a random node is returned to the node to
help it continue initializing. A flag noting some node is joining is set. Given
information, the node joining the DHT will (1) find a successor and a predecessor
(2) build a finger table, and (3) distribute update notice to predecessors to let
them update their finger tables.

void postJoin(1: string ip, 2: i32 port),
// When a node finish initialization, it will call postJoin to reset the joining
flag, so that other nodes can now join.

NodeInfo getNode()
// Get a random node from DHT. This is used when a client wants to get a random
node to contact and also used in join.
```