Kyle Summers, Anish Katukam

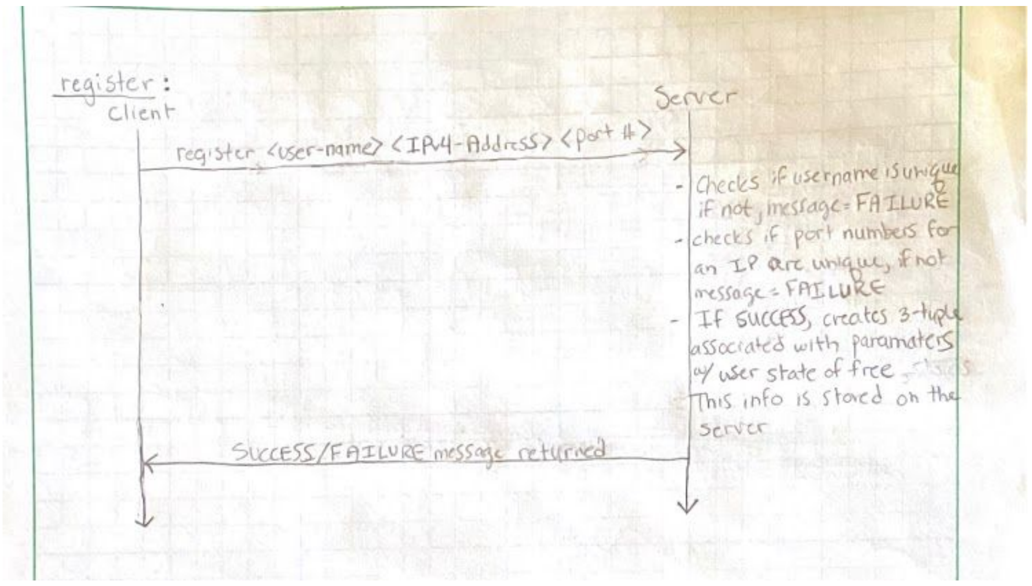
15 March, 2020

Socket Project Design Document

Professor Syrotiuk

The following described program uses sockets, provided by the Java library, in order to utilize communicating processes to build a Distributed Hash Table and then query this structure. This is done through the use of an "always-on server" that maintains an open communication channel with each of the *n* nodes that are registered, and then uses a socket for each one in order to send or receive commands/acknowledgements. This is all done through a series of commands and responses described as follows.

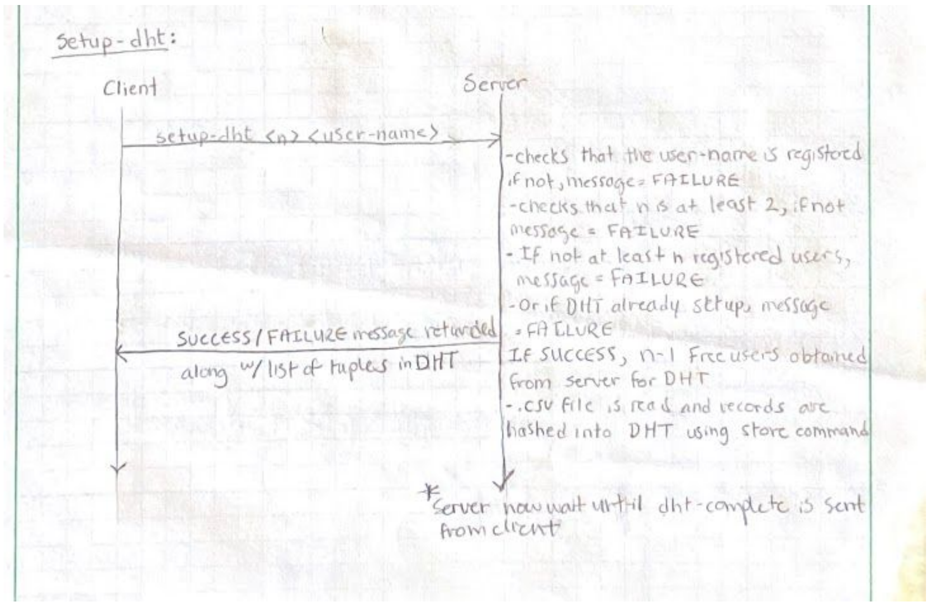
Ther server contains information about the registered users in this system. These users can be in one of three states: Free (able to participate in any capacity), Leader (user leading the construction of the DHT), and inDHT (user is one of the members of the DHT). Register is a command sent to the server by a client. A register message begins with the keyword "register", a username string (must be unique), an IPv4 address, and a port. This data is parsed and stored in an array stored on the server, and is immediately checked for appropriate uniqueness. If these are verified, then the server sends a SUCCESS code back to the location and port specified in the register command, also the state of the user is set to Free.


register :
Client
Server
register <user-name> <IPv4-Address> <port #>
- Checks if username is unique
if not, message = FAILURE
- checks if port numbers for an IP are unique, if not message = FAILURE
- If SUCCESS, creates 3-tuple associated with paramaters w/ user state of free
This info is stored on the server
SUCCESS/FAILURE message returned

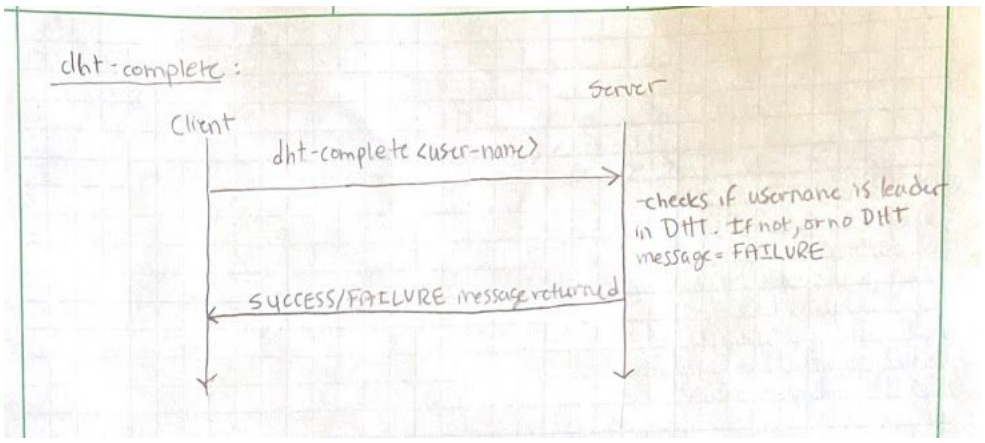
Setup-DHT is a command sent to the server by a client. It indicates intent to construct a DHT using the current existing list of nodes with a specified username as its leader. The server will check for a few conditions — that the username is registered and that certain basic checks (sufficient connected users, no existing DHT) are fulfilled. Once the server has ensured these conditions hold, the server will set the state of the specified username (each client has one of three states — free, leader, and InDHT) to leader and then selects a set of users to form the DHT, updating their states to inDHT. These processes store and receive/transmit messages in a logical ring, with each process knowing the address of it's left and right neighbor.

The DHT is then populated with records obtained from a .csv file using a provided hash function. To store these records, the file is read from input and each record (which is represented as a 9-tuple of data) is held as an array of strings. These arrays are all then placed inside of an arrayList which holds the records before they are hashed and stored on the logical ring. To store these on the rings, we first compute the sum of the ASCII values of the long-name held in each 9-tuple. That sum is modded with 353 to obtain its position on the hash table (will be referred to as *pos*) of it’s node on the logical ring, though we don’t know which node it will be stored on. To obtain the node we take hash table position value obtained above and mod it with n, where n is the number of nodes in the ring, this value will be referred to as *id*.

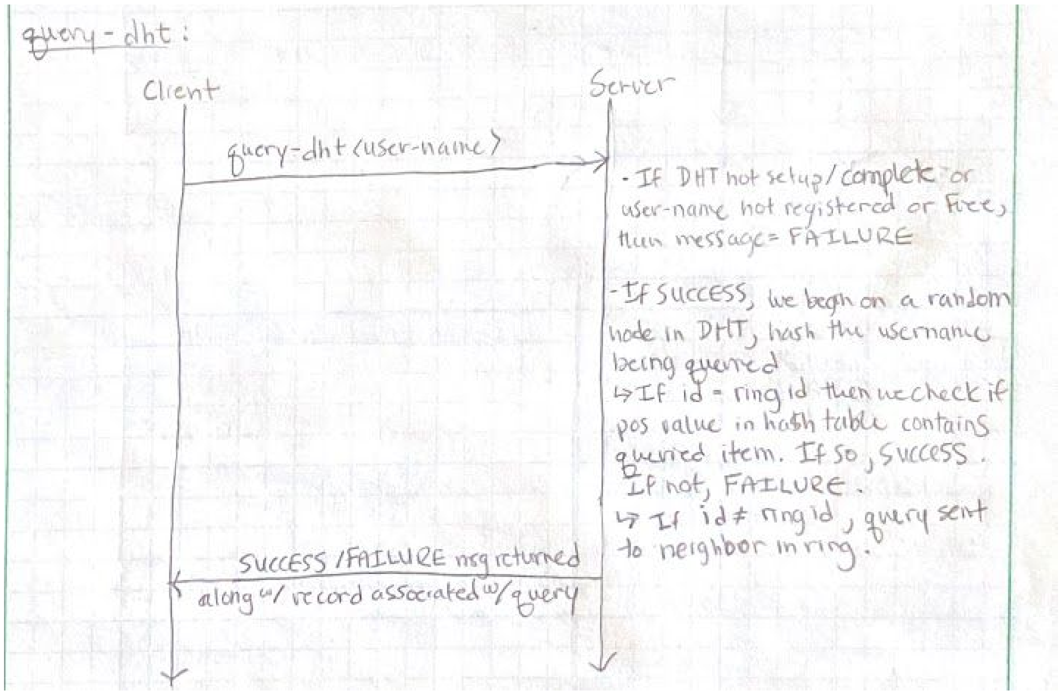
If all the conditions checked by the server are satisfied, it returns a message of SUCCESS along with the list of n users that will construct the DHT. Otherwise, the server responds with a message of FAILURE. The format of the command is setup-dht <n> <username> where n is the amount of users that the DHT’s records will be distributed over, and username is the client whose state will be changed to leader.


Setup-dht:
Client
Server
setup-dht <n> <user-name>
-checks that the user-name is registred if not, message = FAILURE
-checks that n is at least 2, if not message = FAILURE
-If not at least n registered users, message = FAILURE
-Or if DHT already setup, message = FAILURE
If SUCCESS, n-1 Free users obtained from server for DHT
-.csv file is read and records are hashed into DHT using store command
SUCCESS/FAILURE message returned along w/ list of tuples in DHT
*Server now wait until dht-complete is sent from client

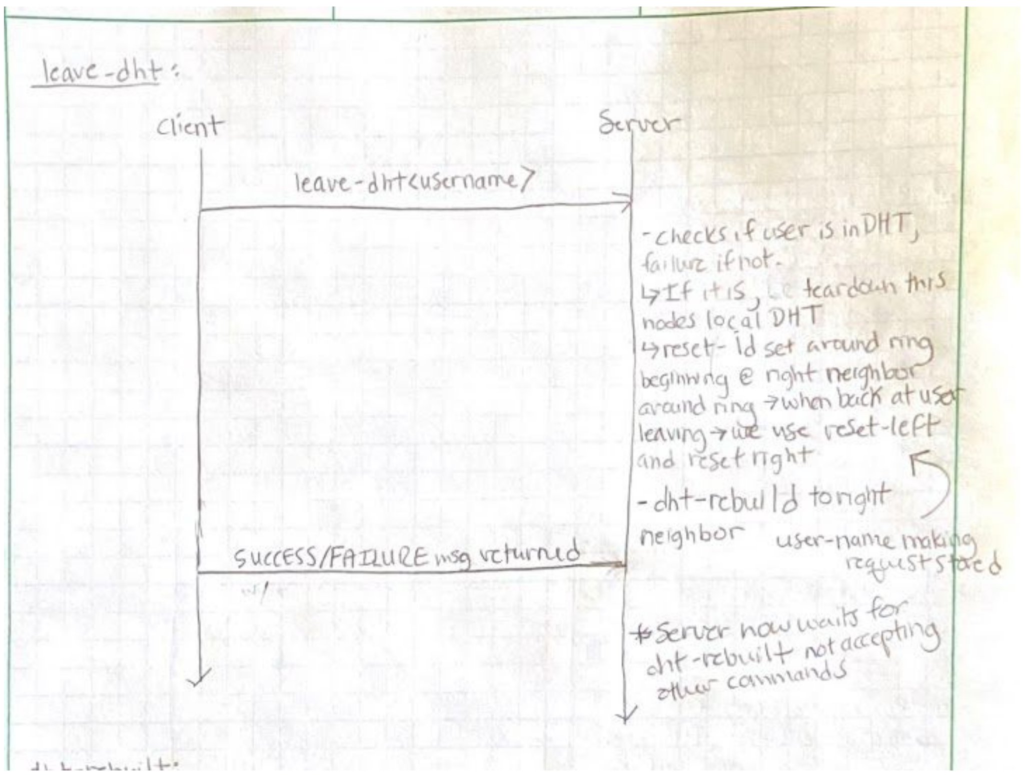
After the setup-dht command is issued, assuming the server responds with a message of SUCCESS, the server now waits for a dht-complete command to be issued. This command is used to indicate that the leader had completed all necessary steps in setting up the DHT. The format of this command is: dht-complete <user-name> where the user-name is expected to be the name of the leader of the DHT. The server responds with a FAILURE message if this user-name is not the name of the leader’s and otherwise returns a message of SUCCESS.


dht-complete:
Server
Client
dht-complete <user-name>
-checks if username is leader in DHT. If not, or no DHT message = FAILURE
SUCCESS/FAILURE message returned

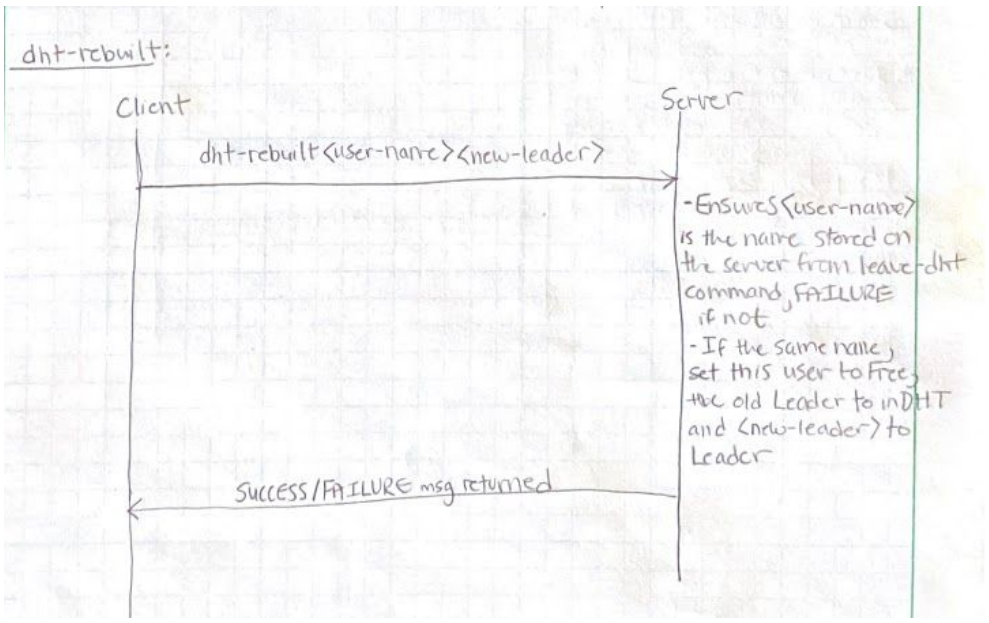
Once the DHT has been completed, a user that has a state of Free is able to query the DHT through the command format: query-dht <user-name>. If the DHT has not been fully set up, or the user sending the query is not Free or not registered, then the server will respond with a receipt of FAILURE. Otherwise, the server begins at a random node in the logical ring and the server will return a message indicating where to initiate the query. The user sends a query which has a 'long-name' to an IP-address and port number from the 3-tuple associated with that node in the logical ring. A user receiving the query computes a hash function of the long-name provided and checks to see if it's *id* value equals the user's identifier on the ring. If it does equal the identifier, then it checks to see if the *pos* value calculated by the hash function in the local DHT holds the record associated with the long-name being queried. If it does, that user returns SUCCESS along with the entire record associated with long-name back to the user who originally issued the query-dht command. If the *pos* value does not hold the record being queried but the *id* did match, then a return message of FAILURE is sent. If the *id* values do not match however, then the query message is forwarded to it's right neighbor for processing. This will continue until the *id* value does match, at which point will check the *pos* values. FAILURE messages will include an output stating that the record associated with long-name was not found in the DHT.


query-dht :
Client
Server
query-dht <user-name>
- If DHT not setup/complete or user-name not registered or Free, then message= FAILURE
- If SUCCESS, we begin on a random node in DHT, hash the username being queried
↳ If id = ring id then we check if pos value in hash table contains queried item. If so, SUCCESS. If not, FAILURE.
↳ If id ≠ ring id, query sent to neighbor in ring.
SUCCESS /FAILURE msg returned along w/ record associated w/ query

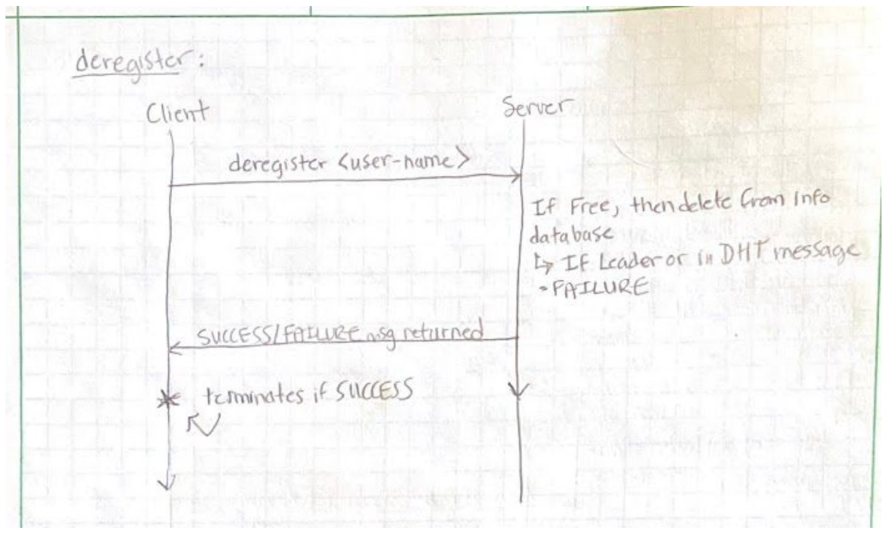
It is also possible for a user to leave the DHT through a command with the format: leave-dht <user-name>. To do this we rebuild a DHT with n-1 nodes by initiating a teardown and deleting its own DHT. then a reset-id command is sent by this user to its right neighbor which issues a renumber of all the rings with a new ring size of n-1. We also introduce two commands, reset-left and reset-right, in order to manipulate the user-to-be-removed's neighbors so that they do not recognize that user anymore as a neighbor. Then the user sends a rebuild-dht command to it's right neighbor, which is now the new leader of the ring. This neighbor uses the techniques from the setup-dht command to create the new DHT. After this is constructed, the user that was deleted from the original ring sends a dht-rebuilt command to the server, with the new-leader (discussed below) set as it's right neighbor. It should be noted that the server, after receiving a leave-dht command, does not take anymore commands until the dht-rebuilt message is received.


leave-dht:
Client
Server
leave-dht<username>
-checks if user is inDHT, failure if not.
↳If it is, teardown this nodes local DHT
↳reset-Id set around ring beginning @ right neighbor around ring → when back at user leaving → we use reset-left and reset right
-dht-rebuild to right neighbor
user-name making request stored
SUCCESS/FAILURE msg returned
*Server now waits for dht-rebuilt not accepting other commands

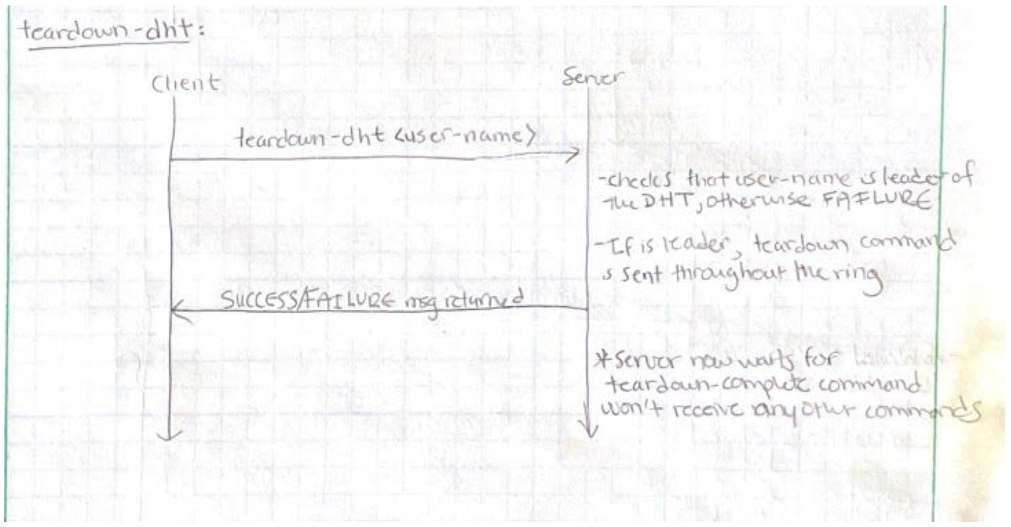
The dht-rebuilt command is a command used to indicate the proper steps have been taken to remove a user from maintaining the DHT. The format of this command is: dht-rebuilt <user-name> <new-leader>, if the user-name specified in the command is not the same as the user initiating the leave-dht command then a FAILURE message is sent. If it is the same user, then that user is set to Free. When rebuilding the DHT a new-leader will often not be the original leader of the DHT. If this is the case, then the state of the old leader is set to inDHT and new-leaders state is set to Leader.


dht-rebuilt:
Client
Server
dht-rebuilt <user-name> <new-leader>
-Ensures <user-name> is the name stored on the server from leave-dht command, FAILURE if not
-If the same name, set this user to Free, the old Leader to inDHT and <new-leader> to Leader
SUCCESS/FAILURE msg returned

A user may also be deregistered from the information base with a command in the form of: deregister <user-name>. If the user stated in the command is maintaining the DHT, then a response of FAILURE is sent. If the user is a Free user, however, then that user is deleted from the information base and the server responds with a SUCCESS message, the user process is also terminated.

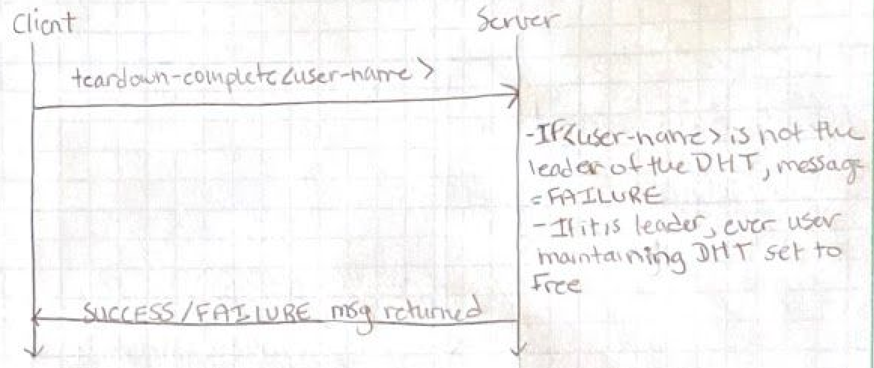

deregister:
Client
Server
deregister <user-name>
If Free, then delete from info database
↳ If Leader or in DHT message - FAILURE
SUCCESS/FAILURE msg returned
terminates if SUCCESS

The DHT can also be deleted using the teardown-dht command (formatted teardown-dht <usr-name>). This command requires the user-name in the command to be the leader of the current DHT otherwise the server returns FAILURE. If it is indeed the leader sends a teardown command throughout the ring by sending teardown to it's right neighbor, and that neighbor sending teardown to it's right neighbor and so on. This teardown command is the same used in leave-dht and simply deleted the local DHT in each ring node. Once it moves all the way around the ring (teardown comes back to the leader) a teardown-complete command is sent to the server.


teardown-dht:
Client
Server
teardown-dht <user-name>
-checks that user-name is leader of the DHT, otherwise FAILURE
-If is leader, teardown command is sent throughout the ring
SUCCESS/FAILURE msg returned
*Server now waits for teardown-complete command won't receive any other commands

The teardown-complete <user-name> command simply indicated that the DHT has been deleted and is only sent once a leader received a teardown sent back to it from throughout the ring. If the user in the command is not the leader of the DHT, then a FAILURE message is returned. Otherwise, the server changes the state of every user in the ring to Free and responds to the former leader with SUCCESS.

teardown-complete:



-If<user-name> is not the leader of the DHT, message = FAILURE

-If it is leader, ever user maintaining DHT set to Free