Ron Searl

Ben Sommer

Jon Tedesco

MP 2 – Distributed Hash Table

Duplicate Key Mapping:

We decided to store the file info for each node in a map data structure where the filename was the key and the value was a FileData struct. The FileData struct contained the information about the file, including the ip address. This approach made duplicate hashing a non-issue, in terms of using the hash function that was provided. The hash function merely tells us at what node to store the file information. However, we don't use that hashed value to actually get the file information once you are at the correct node. The only possible problem with duplicates in this scheme is if you tried to enter 2 files with the same name because a map only stores one of each key so the older file data would just be overwritten. However, that error appears to be outside the scope of this mp because the lookup function, as defined in the spec, only does a lookup with the filename. That means you would never be able to differentiate files of the same name on a lookup anyways.

Test Measurements:

- file1.txt – 24
    - Yes, this makes sense because ADD_NODE took 2, 3, 4 and 5 messages. ADD_FILE took 1 message because f1 was at Node 0, and ADD_FILE f2 and f3 both took 2 messages because they were stored at Node 32. Finally, the quit

command takes N messages where N is the number of nodes in the ring. This gives a total of 24 messages.

- file2.txt – 29
    - Yes, this makes sense because all that is added are 3 FIND_FILE from the file1.txt run. F1 is stored at Node 0 (1 message) and F2 and F3 are stored at Node 32 (another 2+2 messages). This gives us 24 + 5 = 29 messages total.
- file3.txt – 34
    - Yes, this makes sense because all we added was two GET_TABLE commands where GET_TABLE 32 took 2 messages listener -> Node 0 -> Node 32 and GET_TABLE 11 took 3 messages listener -> Node 0 -> Node 10 -> Node 11, giving a total of 5 messages added on to the message count of file2.txt.

Work Distribution:

- Jon
    - Coded the class structure (All decomposition, .h, & skeleton .cpp files)
    - Everything in the Listener
    - Extensive documentation
    - Cleaned up the code
    - Debugged outputs, final testing, & turnin
    - Coded the QUIT, constructions, and helper functions in the Node class
- Ben

    - Coded the ADD_NODE and GET_TABLE functionality in the Node class
        - A constructor
        - addNode()
        - findBestFinger()
        - createFileMap()
        - getTable()
    - Helped with this write-up
- Ron
    - Wrote the Node initialization and ADD/DEL/FIND file functionality in the Node class
        - A constructor
        - Initialize()
        - listenForCommands()
        - acceptConnectionHelp()
        - handleCommand()
        - add/delete/find file()
    - Helped with this write-up
- All of us

- Helped with conceptual design
- Helped debug and combine functionality
- Made minor modifications to each other’s design/functions as necessary