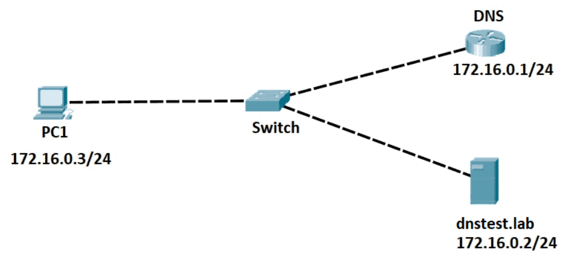
Cache lookup algorithm

Authors: Jeff Chow (34193189), Casey Yan Yu Chung (33112169)

Introduction/context

In a host/routers , cache, or temporary data of websites visited is stored to avoid frequent requests. Information such as DNS responses, routing table information are kept in the router to reduce the amount of time and resources consumed when requesting for routing information.

When we have an ip address, we need to either go to the DNS server to get the MAC address of the destination or look it up in router cache, which is a memory that stores the previous DNS response.



Routers generally use DRAM as the main memory component for its cheap cost[8], however at a cost of its speed. Hence, an efficient search algorithm must be implemented in order to make up for this disadvantage.

In the report, we are looking at the exact match lookup methods possible in cache lookup, as we want to find the MAC address corresponding to the given ip address.

Photo source: https://study-ccna.com/configure-cisco-device-as-dns-server/

Experiment

The purpose of this experiment is to find which of the 4 search algorithms uses the least time in cache lookup.

In this experiment, a simulation was performed by searching all the information in the data set one by one and averaging the time. The experiment only takes the searching efficiency into consideration, but it is important to know that data structures such as binary search and hash takes more time to build.

We will look at the following 4 algorithms:

Linear search:

Linear search is the simplest search algorithm. It works by going through the array one by one until it finds an exact match, having a time complexity of O(n).
Linear search has efficiency less prone to the size of the cache table compared to other search methods such as hash tables. [1]

Binary search tree:

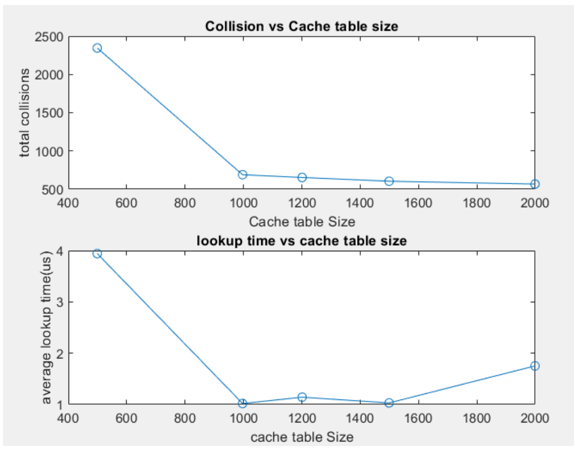
One of the first data structures for databases is implemented by using pointers and determining whether an object is bigger or smaller, creating a tree-like structure[8]. The structure reduces the number of variables to go through in a search.[2] it has the time complexity of O(log n).

Hash with linear probing:

Hash tables are one of the most used data structures. It uses a hash function to assign keys to the index. However, choosing the correct function is important, as it will affect the search time in general. The best case can be O(1). [3] [5]
Since linear probing is used, if collisions happen too often, the system will decide to linearly go through all the possible elements of the search table. Hence in the worst case scenario, the hash search will work like linear searchO(N)[3]. In this experiment, the multiplication hash function is used for its ability to control its parameter hence reducing collision rate.[4]

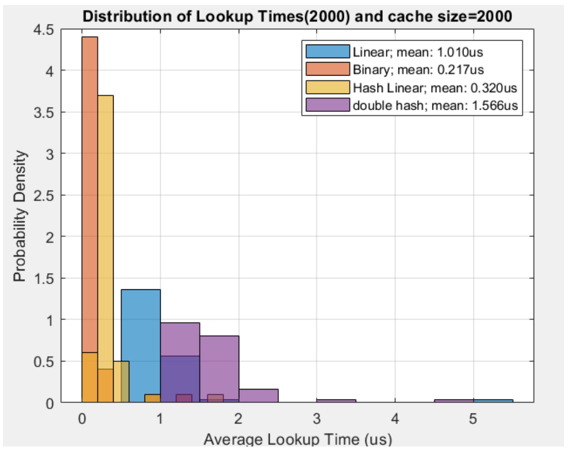
Double hash:

Double hash is based on a hash table , with modification that prevents collision by introducing a second hash function which determines the step size. However, there is a limit in table size having to be a prime number to prevent infinite loop cases.[6] Double hash has a smaller time complexity over hashing with linear probing, as it reduces the occurrence of collisions. So it has smaller lookup times, improving time efficiency.



From the plot, we can see that collision times decrease with the increase of cache table size, and the average lookup time also decreases with the cache size increase to a certain amount(1400~1600).

We will be testing the time difference between these exact lookup methods.



Results:
I found that the execution time varies with each time I run the code. Hence, I plot the distribution and calculate the mean.
In the graph, we can see that binary search is the quickest search algorithm out of the 4.

However, the time it takes for double hash to look up MAC addresses is even longer than linear search, this is because it involves the time for computing two hash functions during each look up. I also define hash function as another function in matlab, spending extra time for matlab to switch between function calls.

Conclusion: In theory, the time it takes to look up a corresponding MAC address on average should be Linear search > hash table with linear probing >= double hashing >= binary search. If the best case happened on both hash algorithms, the time complexity can be O(1), becoming the best choice in terms of searching efficiency. This highly depends on the choice of hash function and cache size. While the binary tree is the quickest search method as shown in the result plot, it is important to note that this experiment is limited to a certain setting with only time spent in cache search being evaluated. There are other realistic considerations such as the trade off between the cost of memory and the efficiency of the cache lookup.[7]

References

[1]Linear Search Algorithm Available: https://www.geeksforgeeks.org/linear-search/. .

[2]Binary Search Available: https://www.geeksforgeeks.org/binary-search/.

[3]Implementing own Hash Table with Open Addressing Linear Probing Available: https://www.geeksforgeeks.org/implementing-hash-table-open-addressing-linear-probing-cpp/

[4]Hash Tables in DSA Available: https://www.w3schools.com/dsa/dsa_theory_hashtables.php.

[5]Hash Functions and List Types of Hash Functions Available: https://www.geeksforgeeks.org/hash-functions-and-list-types-of-hash-functions/.

[6]Double Hashing Available: https://www.geeksforgeeks.org/double-hashing/.

[7] Network Algorithmics : An Interdisciplinary Approach to Designing Fast Networked Devices“Exact-match lookups” *book*, [book]. Available:https://ebookcentral.proquest.com/lib/monash/reader.action?c=UERG&docID=7135107&ppg=262 .