Design Report

Distributed Systems CS - 4262

Content Searching In a Distributed Overlay Network

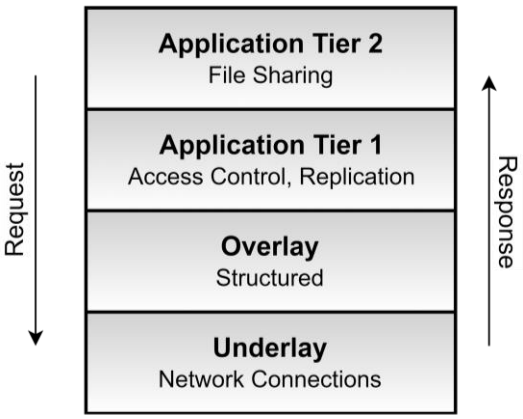
- Imesha Sudasinghe (130580E)
- Vithusha Aarabhi (130625A)
- Jayan Vidanapathirana (130613K)
- Keet Sugathadasa (130581H)

Proposed Topology

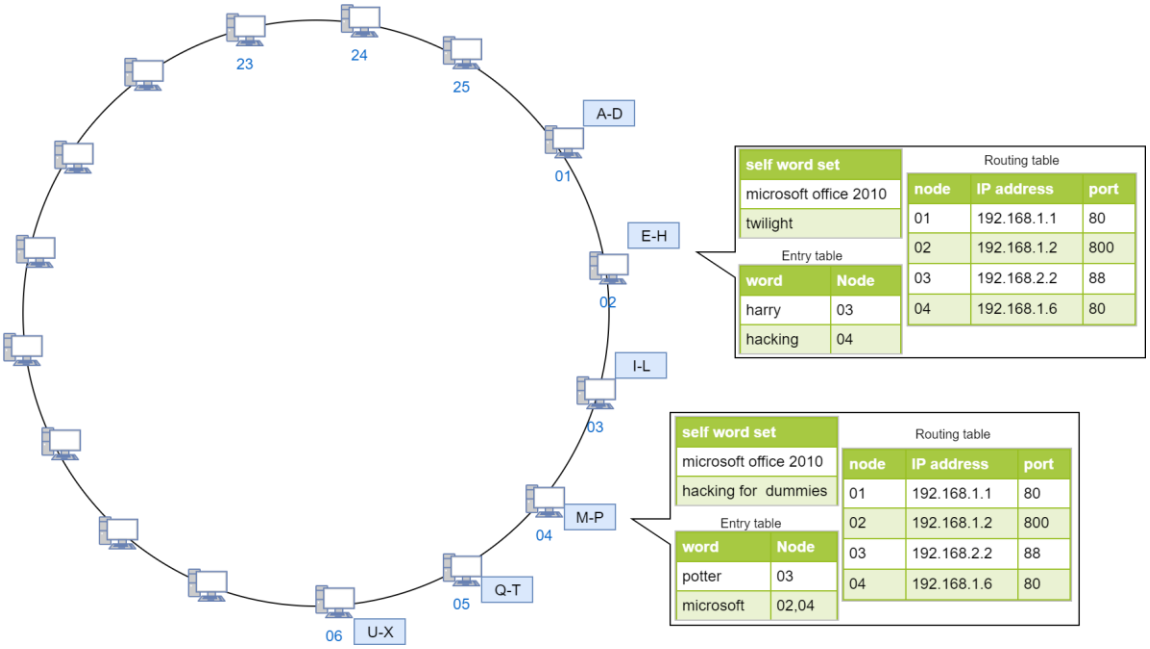
Architectural Style: Layered Architecture

Each layer in the architecture is well defined. Each of these will be built separately.

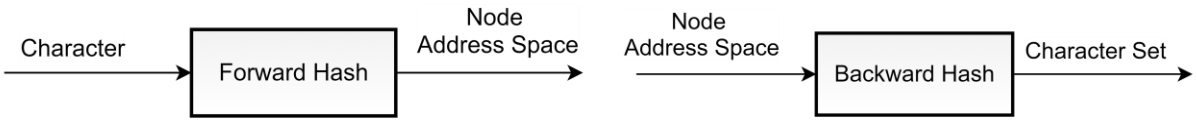
- **Application Tier 2**: Provides the interface that interacts with the user, for file sharing
- **Application Tier 1:** Provides Access Control Replication and other functionalities required to interact with the overlay network
- **Overlay**: A structured network where the nodes connect in an overlay
- **Underlay**: The layer which contacts with the hardware underneath



System Level Architecture: Peer to Peer - Structured (Overlay)



- Expected number of nodes = 25, address space size = 50.
- Define a naming system for each address space slot
- Each node has two hash functions



Communication among Nodes

Communication between the nodes, mainly happens through the routing table. Each node communication will happen via UDP. In this proposed system, only the following instances will require communication.

1) When a new node is joining (Pseudo Code given below)

a) Contact the initial two nodes sent by the Bootstrap server and request for their routing tables.
b) Once the new node establishes itself in the address space, it will do a broadcast to all nodes, informing about its address, IP and port.

2) When a node is gracefully departing
   a) Pass your entry table records to the immediate successor
   b) Send a broadcast to all the nodes, asking them to remove it from their routing tables and unicast to remove the file indices from the relevant hashed node.
3) When querying a file name (Pseudo Code given below)
   a) Check the file name, and according to the hash function, locate the address space, and check whether that address space is occupied. If so, check the routing table and go query from that node N. Else, get the immediate successor, by incrementing address space.
   b) If N has an index to that file, it will return all the nodes that contain the file.
4) Periodic Routing Table Update
   a) Each node will send a heartbeat to each entry in their routing tables
5) Addressing an ungraceful departure:
   a) Successor and predecessor of each node will have a copy of indexed entries of the subjected node (replication).
   b) When a node departures ungracefully, predecessor and successor will identify that the neighbour is gone now (through periodic heartbeat). Once identified, successor of those nodes will take over providing for indices related to the lost node.
   c) Routing tables will be updated accordingly.

Format of Routing Tables

Node Name	IP Address	Port
01	192.168.1.1	80
02	192.168.1.2	800
04	192.168.2.2	88
10	192.168.1.5	80

The routing tables, will have the following structure.

- A routing table will be maintained at every node in the system. Initially, a new node will be getting the IP address and Port number of a few selected nodes, via the bootstrap server. Then, this new node will request for the other nodes' routing tables and update its own.

- Once the new node gets established within the system, it will send every node a message with the new node's "name, ip, port".

- A periodic routing table update will happen for each node separately, where each node will contact the nodes in its own routing table, update. A random walk will can be used to generate the routing table at the initial stages.
- At a given point in time, each node will have a common routing table, in the above format. This is all based on the assumption that network will be available at all times.

Performance Parameters and How to Capture Them

- Query Hop Count
  - Each query will be having a counter which keeps track of the number of nodes it visits
- Our approach should provide the search results in nearly O(1) time.
  - Can be directly calculated/measured

- Latency of each search - Using a local timestamp in the requesting node and getting the difference between the query trigger time and the result delivery time.
    - Can be measured directly
- Rate of successful search - Number of queries that were successful / Total number of queries issued.
    - Can be calculated using the test queries provided
- Network utilization at query time - How much of network has been utilized when the search query was issued. (Should be very low in our case since the access is supposed to be O(1) in most of the cases)
    - Requires monitoring in and out data transfers and the capacity of the link. We may need to assume that all the data transfers during that particular time is corresponding to the query requests.
- Network utilization at routing table synchronization phase - How much of the bandwidth has been used for resolving node name - ip pairs.
    - Can be measured similarly as in network utilization in query time.
- Time for successor to take over a node once a node is gone ungracefully - This is similar to the time to response for failures of neighbors.
    - Can be directly measured by randomly disconnecting nodes from the network (by unplugging LAN cables) and measuring time to respond and etc.

Pseudo Codes

Joining a New Node (general scenario)

1. The new node (Z) will assign a node name (A) randomly to itself, from the address space.
2. It will get the routing table of the first node, and update its own routing table.
3. Get the IP addresses and Ports of 2 nodes (X & Y)
4. Update the new node's routing table, from the routing tables of X and Y
5. If address space A, is not empty (there is already another node)
    a. Then randomly assign different names from the address space, until it is available
6. The Z will send msg to every other node (broadcast) in its routing table → (node name A, IP, Port)
7. From the list of predefined file names: randomly assign 3 to 5 file names to a self-list
8. For each filename: Split it by a Space (gets one or two words)
    a. For each word (H): take the first character, and enter it to the hash function.
    b. Go to the returned address space; and look for the node (K) in that space
        i. If no node, go to the immediate successor (K)
    c. Update that nodes entry table, saying, Z has H.
9. Do a replication of that entry on the entry tables of, the successor of K, and the predecessor of K

Query a file name (general scenario)

1. Check whether the searching node (P) has the file
2. Split the filename F; by a space (gets one or two words)
3. For each word: (Do steps 4-6)
4. Get the first letter and insert it to the Hash Function
5. Go to the returned address space; and look for the node (K) in that space
    a. If no node, go to the immediate successor (K)
6. K will return the list of nodes, that contain the file
7. Get the list of returned nodes, for each word in filename:
    a. And take the intersection of node names, and return it to P