Problem Set #4

**Problem 1 Hashing (20 pts)**

The hash table has 13 slots, and integer keys are hashed into the table with the following hash function H

int H (int key)

{

x = ( key + 5 ) \* ( key - 3 );

x = int( x / 7 ) + key;

x = x % 13;

return x;

}

1. Fill in the final hash table with the following keys: 17, 22, 73, 56, 310, 100, 230, 12, 42, 18, 19, 24, 49.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Slot** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** |
| **Contents** | **49** |  | **18** |  | **22,42** |  | **310, 12,24** | **73,100,19** | **17** | **56** |  | **230** |  |

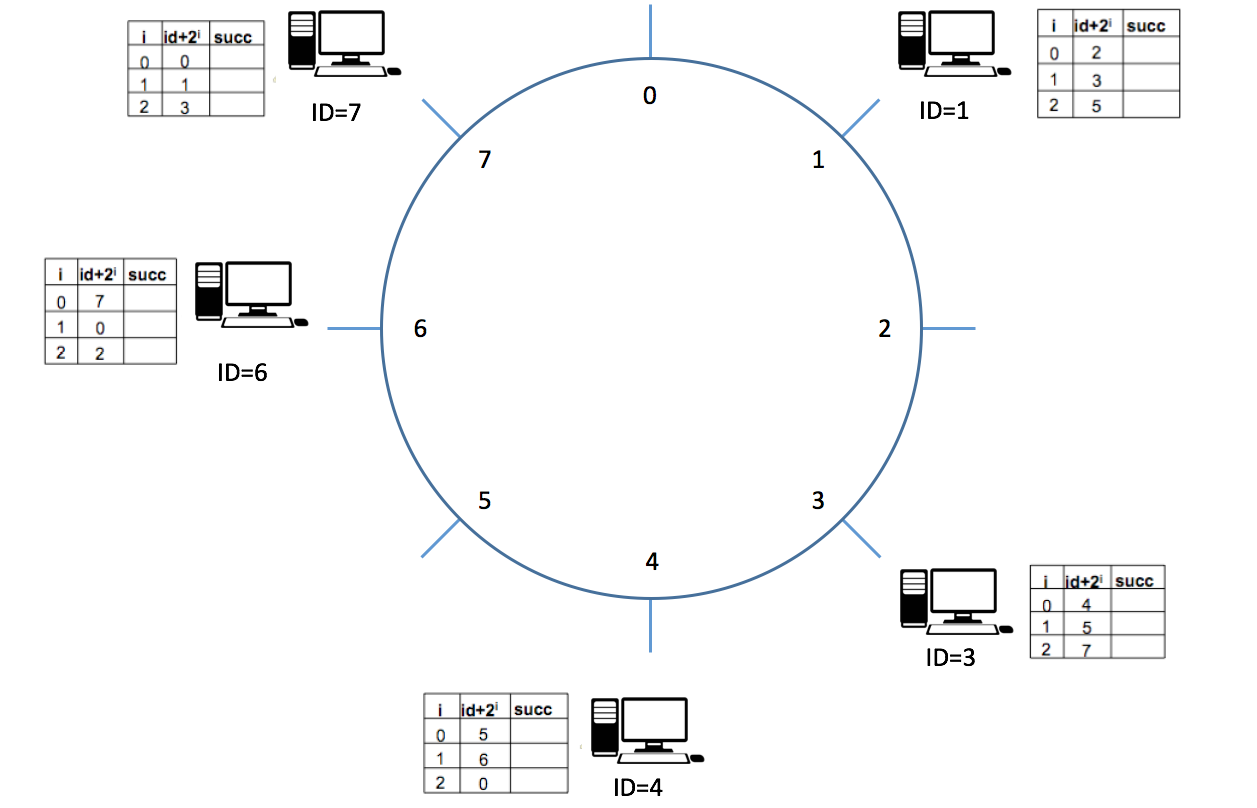
1. List one or two methods that can handle collision in hashing.

One method commonly employed to handle collisions in hashing is known as chaining. In this approach, each bucket in the hash table is associated with a linked list. When a collision occurs, meaning two keys hash to the same index, the collided elements are appended to the linked list at that index. This method allows for multiple elements to coexist at the same index, forming a chain of elements. Chaining provides a straightforward and efficient way to manage collisions without requiring extensive rehashing or probing. Another widely used method to address collisions is open addressing. In open addressing, when a collision happens, the algorithm searches for the next available slot in the hash table. There are various open addressing techniques,  
including linear probing, quadratic probing, and double hashing. Open  
addressing techniques aim to minimize clustering and utilize the available slots in the hash table more effectively. The choice between chaining and open addressing depends on the specific characteristics and requirements of the application at hand.

**Problem 2 Distributed Hash Tables (20 pts)**

There is a Chord DHT in Figure 1. with 5 nodes. The finger tables are listed beside the nodes. Each node may be storing some items according to the Chord rules (Chord assigns keys to nodes in the same way as consistent hashing)

Figure 1. Chord DHT for Problem 2

****

1. Fill in the table for node id=1 and 7

|  |  |  |
| --- | --- | --- |
|  | ID＋2i | successor |
| 0 | 2 | 3 |
| 1 | 3 | 3 |
| 2 | 5 | 6 |

Table for ID=1

|  |  |  |
| --- | --- | --- |
|  | ID＋2i | successor |
| 0 | 0 | 1 |
| 1 | 1 | 1 |
| 2 | 3 | 3 |

Table for ID=7

1. List the node(s) that will receive a query from node 1 for item 5 (item named by key 5)

Node 1 -> Node 6  
Node 1 has “5” as its finger table with the Node 6. Hence, node 6 will receive the query from node1

**Problem 3 Bloom Filters (10 pts)**

**Derive** the probability of false positive rate after 10 keys (or elements) are inserted into a table of size 100. Assume that 5 hash functions are used to setup bit positions in the table for the keys (elements).

5 Hashing functions means: For each key (element) there will be 5 bits to be set to 1 in the table (it can also be less than 5 if the hash functions generate same outputs).

Hint:

Assume **m** is the number of bits in the filter, **n** is the number of elements and k is the number of hash functions used.

After inserting one key, the probability of a particular bit being 0 is . This is because k hash functions are independent, and each hash function will have probability for a particular bit remain 0. Then after inserting n keys, the probability for a particular bit remain 0 is

Now you have to apply this to the case of false positive.

False positive : The probability of 5 hash functions point to the bits that are set to ‘1’ when the keys have not been inserted yet (‘0’).  
P(False Positive) = (1 − (1 – 1 𝑚)𝑘𝑛)𝑘 = (1 − (1 − 100)(5)(10))5 = 0.0096

**Problem 4 P2P system (20 pts)**

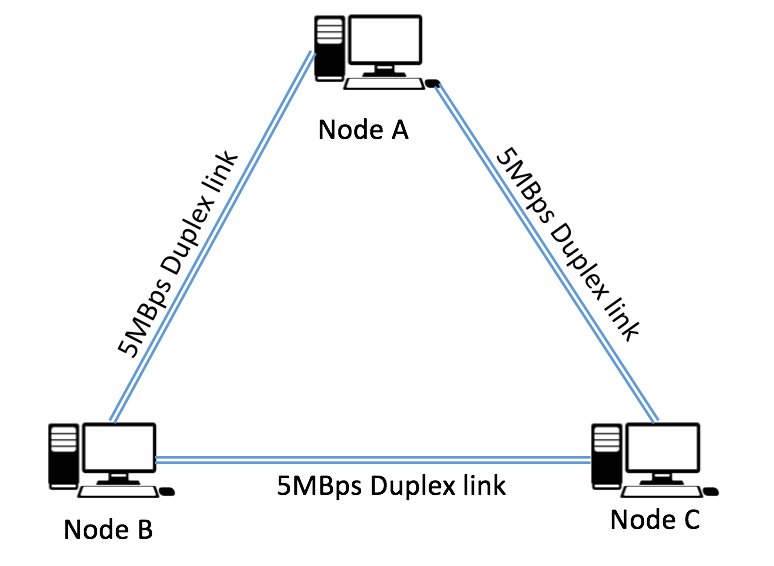


Figure 2. Network topology for Problem 4

3 nodes A, B and C are connected with each other via 5MBps duplex link as is now in Figure 2. Node A want to share a 2500MB file to Node B and C. During the actual transmission 2.5MB piece of the file can be send on the links each time. In the problem we ignore the RTT delay.

1. What is the time of the sharing process using centralized approach? (The process ends when B and C all received the file, and the centralized approach means B and C are communicating with A independently and there is no communication between B and C)

The time of the sharing process = max(2500MB/5MBps, 2500MB/5MBps) = 500s

1. What is the ideal minimum time of the sharing process using P2P approach? (P2P approach means after B and C received a piece from A, they immediately share the their piece to other party)

Ideal minimum time is when both B and C receiving different half of the data from A. Additionally, while the data is transferring from A, B and C are also sharing the data received between each other. So, the time that B and C received half of the data from A is at 250s, and the last piece of the data that will be transfer to each other (B <-> C) will be at 250s + 2.5MB/5MBps = 250.5s

**Problem 5 File Distribution (10 pts)**

Consider distributing a file of F = 20 Gbits to N peers. The server has an upload rate of us = 30 Mbps, and each peer has a download rate of di = 2 Mbps and an upload rate of u. For N = 10 and 100 and u = 300 Kbps and 2 Mbps, prepare a chart giving the minimum distribution time for each of the combination of N and u for both client-server distribution and P2P distribution.

**Client Server**

|  |  |  |
| --- | --- | --- |
| U | N= 10 | N= 100 |
| 300 Kbps | Max(204800/30 , 20480/2) = 10240 | Max(2048000/30 , 20480/2) = 68266.67 |
| 2 Mbps | Max(204800/30 , 20480/2) = 10240 | Max(2048000/30 , 20480/2) = 68266.67 |

**Peer to Peer**

|  |  |  |
| --- | --- | --- |
| U | N= 10 | N= 100 |
| 300 Kbps | Max(20480/30 , 20480/2 , 204800/(30+0.293)) = 10240 | Max(20480/30 , 20480/2 , 2048000/(30+2.93)) = 62192.53 |
| 2 Mbps | Max(20480/30 , 20480/2 , 204800/(30+20)) = 10240 | Max(20480/30 , 20480/2 , 2048000/(30+200)) = 10240 |

**Problem 6 Security (20 pts)**

Diffie-Hellman Symmetric Key Exchange solves key the distribution issue of symmetric keys. Fill in the brackets below. Assume that Alice and Bob know p-ordered group G and a generator g, and Alice’s and Bob’s random seeds are ***a*** and ***b*** and their public keys are ***A*** and ***B***, respectively, and computes a shared key **s**. Please use ***p*** = 23, ***g*** = 11, ***a***= 13, and ***b*** = 8. Note that Eve is an eavesdropper and can see any communication between Alice and Bob.

A white sheet with black text

Description automatically generated