ECEN 757 | Homework 5

Muhammed U. Ersoy
M.S. ECEN Student
Texas A&M University
mue@tamu.edu

Question 1

Consider a chord system with m=7. There are 42 servers. The IDs are: 3,6,9,....,126. A file has a hash value of 70. Where is it stored?

The file would be stored at the first peer that is greater than or equivalent to its hash value $key(\mod 2^m) \rightarrow 70(\mod 2^7) = 70$. In this case it would be stored at the first peer with id **72**.

Question 2

For the same above, List the finger table of node 30 and for node 15.

For node 30:

i	ft[i]
0	33
1	33
2	36
3	39
4	48
5	63
6	96

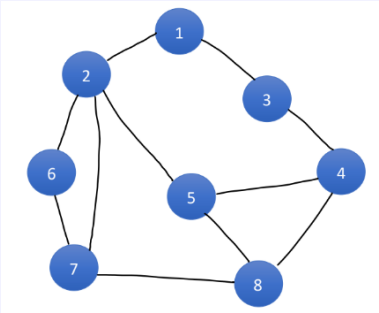
For node 15:

i	ft[i]
0	18
1	18
2	21
3	24
4	33
5	48
6	81

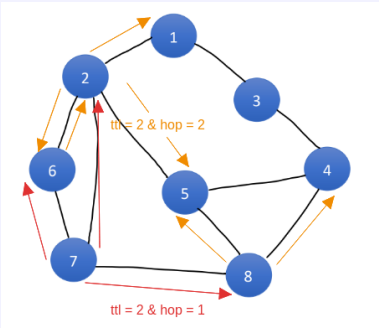
37th Conference on Neural Information Processing Systems (NeurIPS 2023).

Question 3

The following is an overlay of Gnutella. Suppose node 7 starts a query with TTL = 2. Which nodes receive the query? Explain your answer



As seen in the graph below node 7 would send the query message to its immediate neighbours $\{2, 6, 8\}$(red arrows). Which the receiving nodes would mark as $hop = 1$ and the header would have $ttl = 2$ since $hop < ttl$ they would forward the message to their immediate neighbours $2 \rightarrow \{1, 5, 6\}$, $6 \rightarrow \{2\}$, and $8 \rightarrow \{4, 5\}$. These are marked by the orange arrows. In this jump $hop = 2$ and since $hop = ttl$ the flooding would terminate at this step.



Question 4 (10.7)

Explain why using the secure hash of an object to identify and route messages to it is tamper-proof. What properties are required of the hash function? How can integrity be maintained, even if a substantial proportion of peer nodes are subverted?

- The use of a secure hash makes a resource 'self-certifying'. A requestor can validate the requested resource by comparing with its hash. This way an untrusted source cannot replace the *real* resource with a fake/malicious one.
- The function should be random to randomly distribute resources over the overlay network. It should not be a function to 'place' similar objects next to each other or give them similar hashes.

(1) Hash functions should have a reasonable uniqueness guarantee.
For SHA-1 to have a 50% chance of a hash collision, there would have to be 1.42×10^{24} records in the table. Linstedt and Olschimke [2016]

(2) Hash functions should be difficult to reverse meaning given $y = hash\_fn(x)$, x must be computationally infeasible to compute from y, e.g. $x = hash\_fn^{-1}(y)$. Preventing tampering and collision searching.

Requestors can validate the resource they receive by comparing their expected hashes with the calculated hash. This way if a subverted node sends a bad resource the requestor can identify it and ignore it.

Question 5

Consider a Pastry system and a node of ID – 48E175. The following is snap of its routing table.

P	GUID prefixes and corresponding node handles														
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
1	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E
2	480	481	482	483	484	485	486	487	488	489	48A	48B	48C	48D	48E
3	48E0	48E1	48E2	48E3	48E4	48E5	48E6	48E7	48E8	48E9	48EA	48EB	48EC	48ED	48EE

this node has to send packets to nodes,

(a) 48E322

(b) 54A666

Explain how the routing takes place to send packets to the nodes given above.

(a)
1. The first step is to look at the GUID prefix table and figure out the largest matching prefix. In this case it is 48E3XX $\rightarrow n$ where n is the handle of a node matching this pattern. then the current node (48E175) would route the message to n which would compare the delivery id with its own. If it's not the final destination it would look at its' own GUID table to find a closer destination. And the process would continue until the message reaches its' final destination.

(b)
Similar to the previous example the current node would first look at the ID in the message and compare to its own. Then it would look at the GUID table and find the largest matching GUID prefix. In this case it would be 5XXXXX $\rightarrow n$.

References

Daniel Linstedt and Michael Olschimke. Chapter 2 - scalable data warehouse architecture. In Daniel Linstedt and Michael Olschimke, editors, *Building a Scalable Data Warehouse with Data Vault 2.0*, pages 17–32. Morgan Kaufmann, Boston, 2016. ISBN 978-0-12-802510-9. doi: https://doi.org/10.1016/B978-0-12-802510-9.00002-7. URL https://www.sciencedirect.com/science/article/pii/B9780128025109000027.