Surname : Pendyala
Firstname : Praveen Kumar
Matrikel # : 2919474

Some of the answers were arrived at while working in group with:

Lokesh Kumar Jamjoor Ramachandran - 2596208
Ankush Chikhale : 2973449

Problem 10.1

A. Maximum routing state info kept by any node = 160 * k * 26 Bytes

B. From the given routing table for node 101010, k value could be 3 or 4

C. k-bucket range for 101010
   1. 101001
   2. 101000

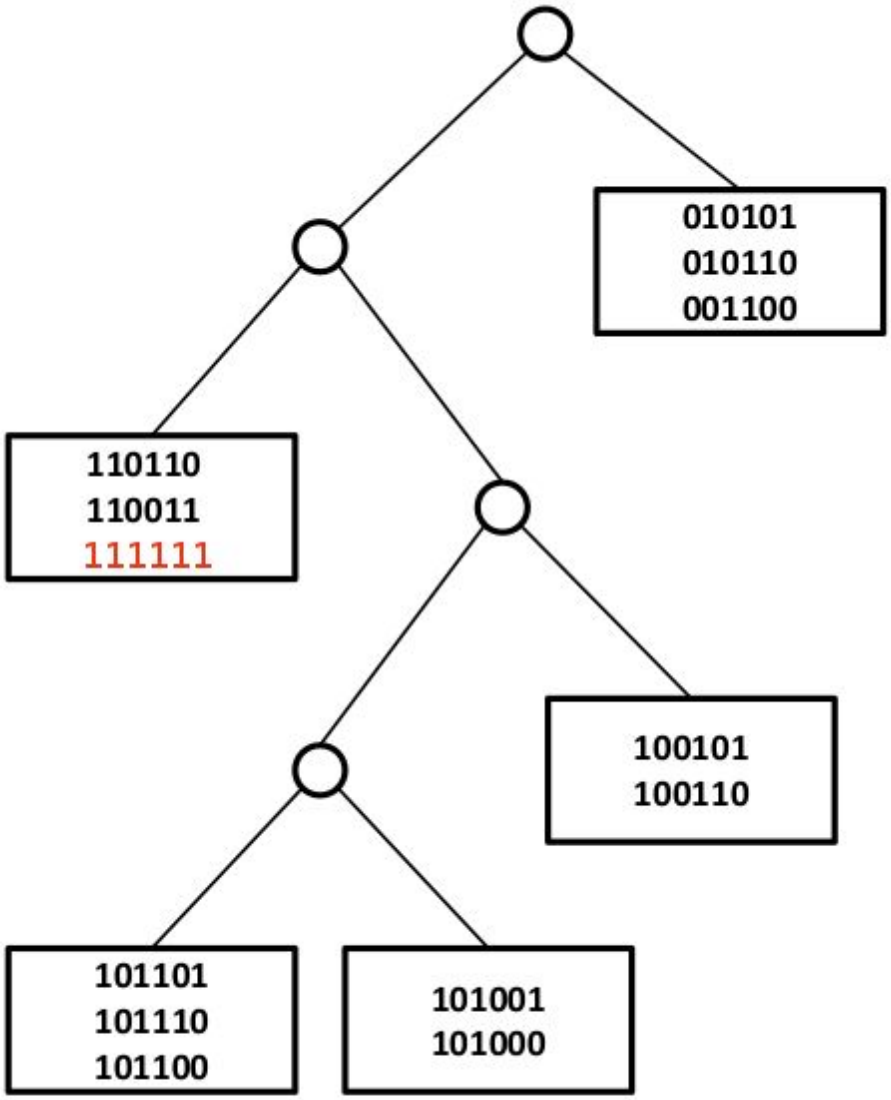
D. New nodes joining the network. (assuming k value as 3)

   1. Node 001110 joined

      - Routing table unaffected. Nodeid discarded without adding

   2. Node 111111 joined

      - New entry added to routing table as shown below



3. Node 110101 joins

   - Network remains unchanged

E. Routing from node 101010 to node 111100 with α = 2,

   - Nodes selected in 1st step : 111111, 110110

Problem 10.2

A. **Chord over CAN**

Reason : Low latency long range communications

Scenario : A particular scenario where nodes often communicate with other nodes that are far from them in the DHT address space. Example. #0000 to #FFFFF. Chord outperforms CAN in such a scenario. Chord maintains a finger for each distance range in the address space so, message reaches destination in few hops. In CAN, each request has to traverse all the neighbours on the connecting straight line.

B. **Kademlia over Chord**

Reason : For higher reliability and fault tolerance.

Scenario : Both Chord and Kademlia saves information about nodes which are in the distance range of $[2^i - 2^{i+1})$. Chord saves only one finger corresponding to each distance range while Kademlia saves “k” of them in each range and thus has a higher reliability in case the finger node in that range fail.

C. **CAN over Kademlia**

Reason : For low latency lookups

Scenario : Kademlia uses iterative looks for all routing operations. So, in situations which need an quick routing to the destination node, CAN can outperform since it doesn’t perform iterative looks and waste more round-trip times.