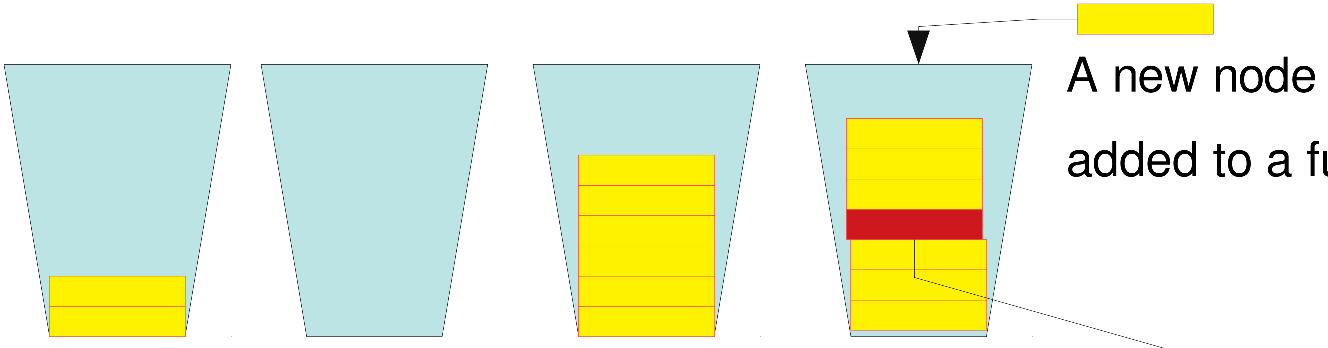
A node in the network has k-buckets which contain up to k IDs of other nodes

There are O(log(N)) k-buckets where N is the size of the network



A new node ID is added to a full bucket.

An offline node ID or the LRU node ID is removed.

Each bucket holds nodes of distance $[2^i, 2^{i+1}]$

Some implementations divide full buckets when necessary.

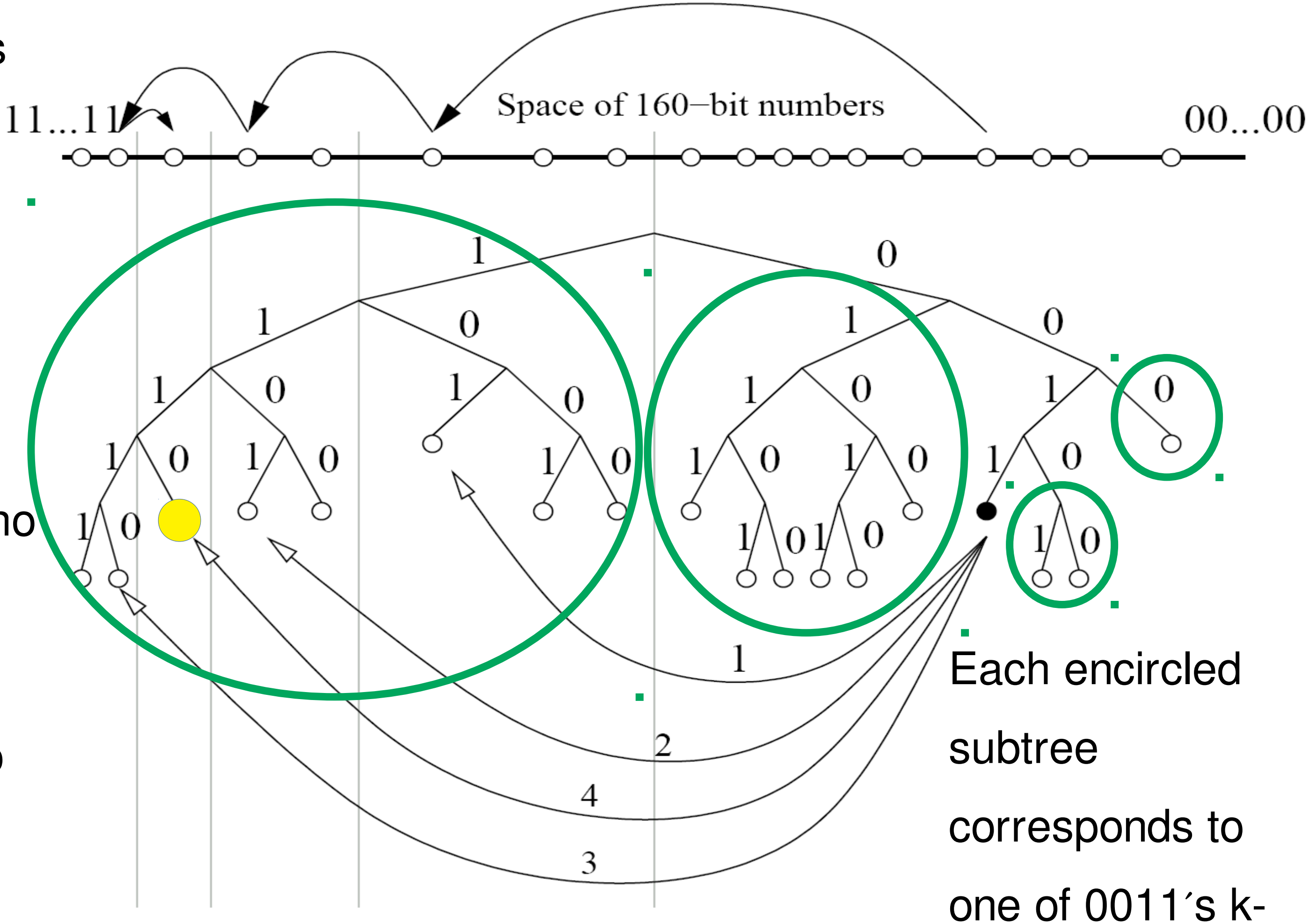
0011 queries 101 who returns 1101

0011 then queries 1101 who returns 11110

0011 thirdly queries 11110 who returns 1110

0011 finally queries 1110 who returns the desired value

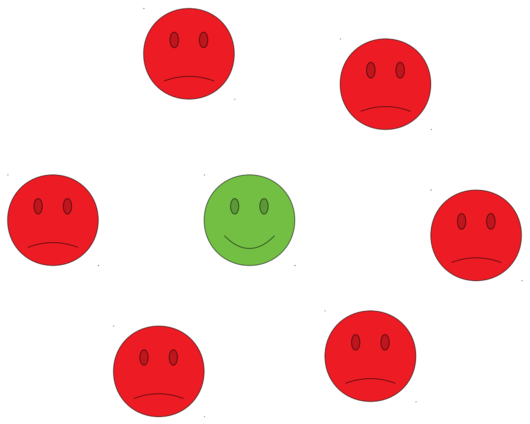
The cost of lookup is O(log(N))



Each encircled subtree corresponds to one of 0011´s k-buckets

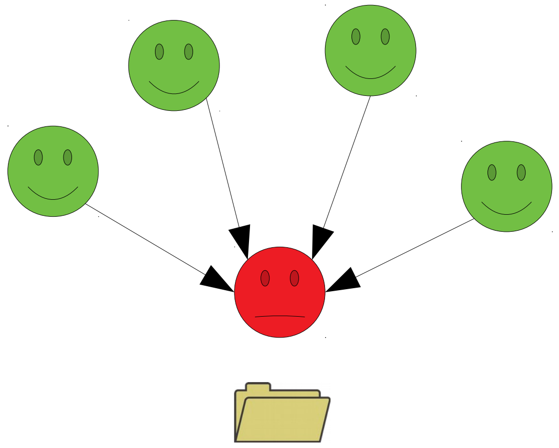
Eclipse Attack

- Malicious nodes represent themselves with node IDs close to their target
- The malicious nodes take over the target´s routing table
- This attack takes advantage of the fact that many areas of the ID space are largely empty
- The query of the attacked node can be altered or the return value can be manipulated



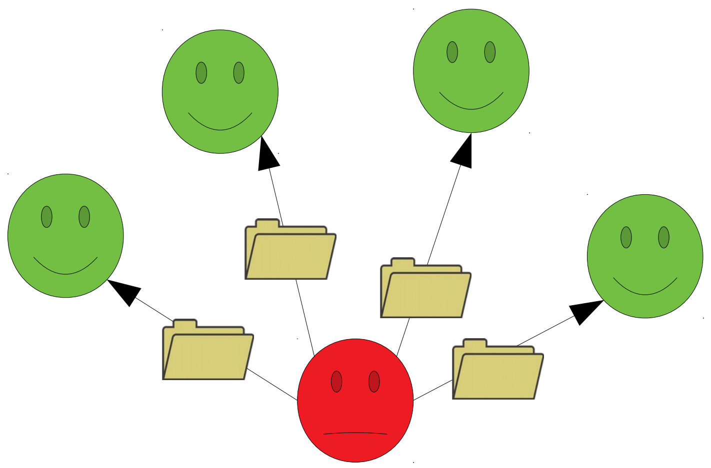
Node Insertion Attack

- A malicious node chooses an ID close to a target key
- Having multiple malicious nodes close to the key increases the effectiveness of the attack
- The malicious node can return false results to a node looking up the target key



Publish Attack

- A malicious node fills the routing tables of nodes close to the target key
- The attack needs to happen consistently to keep the misinformation from expiring and to constantly infect new peers
- This is rarely the most viable attack as peers tend to leave and join the network at high rates



Kademlia vs. Chord and Pastry

- Chord

  - when $\alpha = 1$, Chord has similar performance $O(\log(N))$ lookup

  - cost of join/leave is $O(\log^2(N))$ like Kademlia

  - routing table is not as flexible as Kademlia´s

  - chord uses an asymmetric metric where Kademlia´s XOR metric is symmetric

- Pastry

  - flexible routing table

  - Kademlia is simpler to analyze