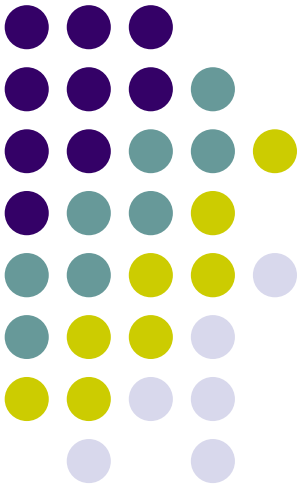
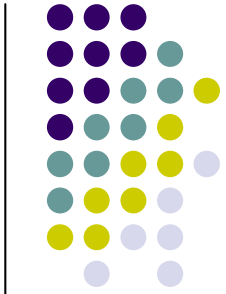
Kademlia: A Peer-to-peer Information System Based on the XOR Metric

Review

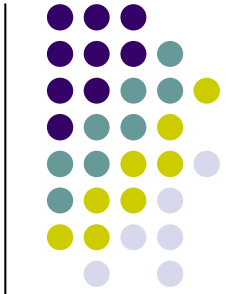


Overview



- Kademlia
  - a peer-to-peer, DHT, ⟨key,value⟩ storage and lookup system
- Features
  - Configuration information (routing maintenance) spreads automatically as a side-effect of key lookups
  - Uses parallel, asynchronous queries to avoid timeout delays from failed nodes

Description: nodes, and keys

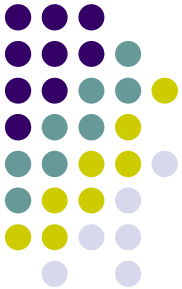


- Each Kademlia node has a 160-bit node ID
- A node chooses a random, 160-bit identifier when joining the system
- Every message a node transmits includes its node ID, permitting the recipient to record the sender's existence
- Keys of items: ⟨key,value⟩ are also mapped to 160-bits identifiers

Hashing and distance

- Nodes and Keys are mapped to m-bit binary strings
- Given two 160-bit identifiers, x and y,
  - The distance between them is defined as their bitwise exclusive or (XOR) interpreted as an integer:
  -
  - d: Id×Id $\Rightarrow$ I
  - d(x,y) = x $\oplus$ y

Hashing and distance (Example)



- XOR: $\oplus$
  - 1 if two bits are different 0 if two bits are the same
- x = 0 1 0 1 1 0 (22)
- y = 0 1 1 0 1 1 (27)
- x $\oplus$ y = 0 0 1 1 0 1 (13)
- d(x,y) = 13
- If x and y agree on most significant (n – i) bits, then, $2^i \leq d(x,y) \leq 2^{i+1}-1$, i=(0..n-1),
- i=3, $2^3 = 8 \leq 13 \leq 2^4-1 = 15$

x = 0 1 0 1 1 0
y = 0 1 1 1 1 0

x$\oplus$y = 0 0 1 0 0 0

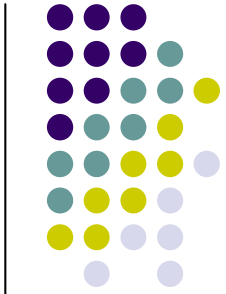
d(x,y) = 8

x = 0 1 0 1 1 0
y = 0 1 1 0 0 1

x$\oplus$y = 0 0 1 1 1 1

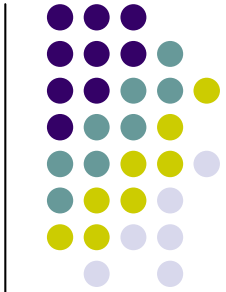
d(x,y) = 15

XOR is metric space



- $d(x,x) = 0$
- $d(x,y) > 0$ if $x \neq y$
- $\forall x,y : d(x,y) = d(y,x)$ (symmetry)
- $\forall x,y,z: d(x,z) \leq d(x,y) + d(y,z)$ (Triangular inequality)
  - $d(x, z) = d(x,y) \oplus d(y,z)$
  - $\forall a \geq 0, b \geq 0 : a + b \geq a \oplus b$

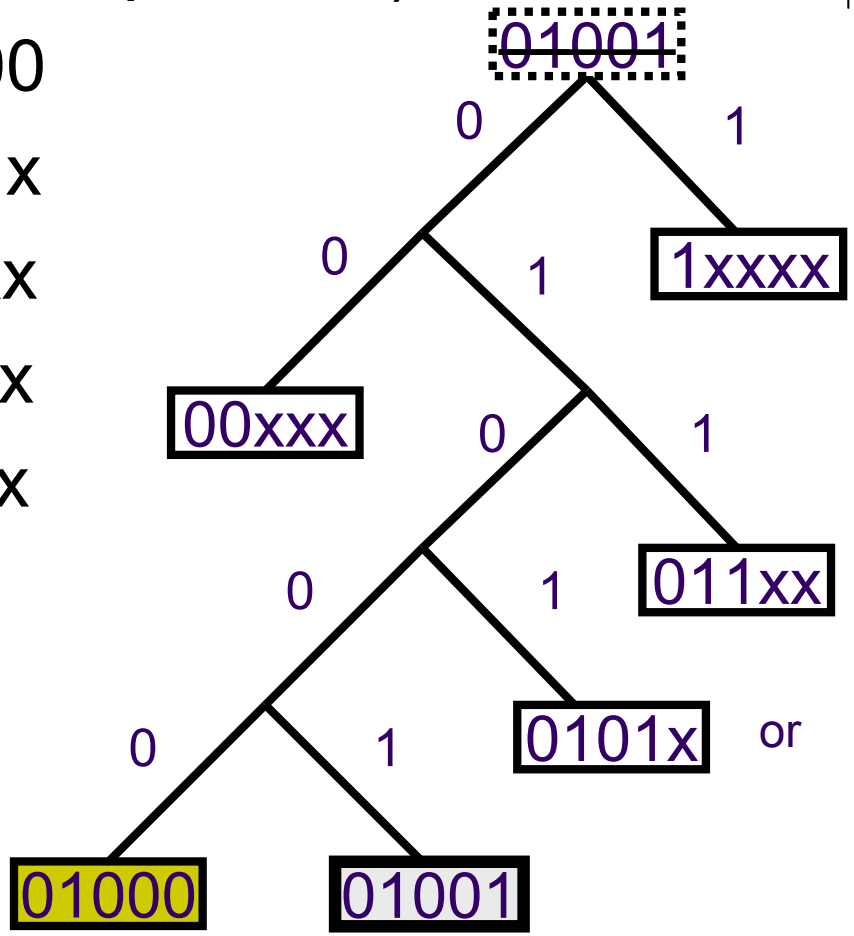
Kademlia – Routing table



- For each $0 \leq i < 160$, every node keeps a list of (IP-address; UDP port; Node ID) triples for nodes of distance between 2^i and 2^{i+1} from itself
- $2^0, 2^1, 2^2, 2^3, 2^4, 2^5, \ldots, 2^{159}$
- Each list is called a **bucket** and stores at most k triples
  - a k-bucket stores at most k nodes that are at distance $[2^i, 2^{i+1})$
    - Empty bucket if no nodes are known
    - Each k-bucket is kept sorted by time last seen

Kademlia – Routing table k = 1

- Node 01001 (identifier space 64)
- Distance $[2^0, 2^1)$:01000
- Distance $[2^1, 2^2)$:0101x
- Distance $[2^2, 2^3)$:011xx
- Distance $[2^3, 2^4)$:00xxx
- Distance $[2^4, 2^5)$:1xxxx



Kademlia – Updating buckets

- Whenever a node receives any message, it updates the appropriate k-bucket by sender's information
- If the bucket is full the least-recently node is removed if it is not live
- keeping the oldest live contacts around, k-buckets maximize the probability that the nodes they contain will remain online.

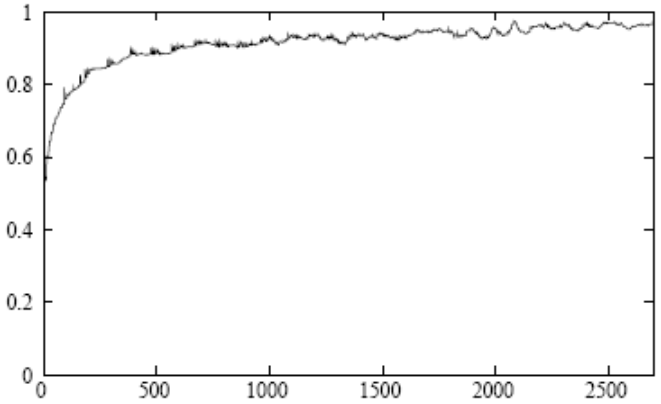
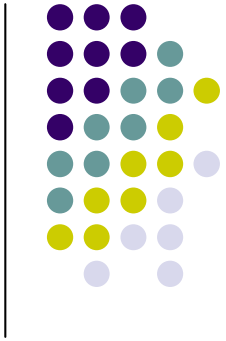


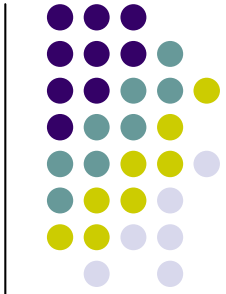
Figure 1: Probability of remaining online another hour as a function of uptime. The x axis represents minutes. The y axis shows the the fraction of nodes that stayed online at least x minutes that also stayed online at least $x + 60$ minutes.

Kademlia Protocol



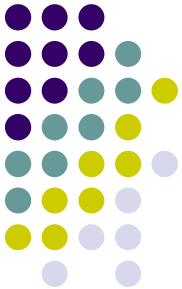
- Consists of four RPCs:
  - PING, STORE, FIND NODE, and FIND VALUE
- The PING probes a node to see if it is online
- STORE instructs a node to store a ⟨key; value⟩ pair for later retrieval
- FIND NODE(ID): The recipient of a the RPC returns triples for the k nodes it knows about closest to the target ID
- FIND VALUE(ID): similar to FIND NODE except that it returns the value if previously stored

Kademlia – Node lookup



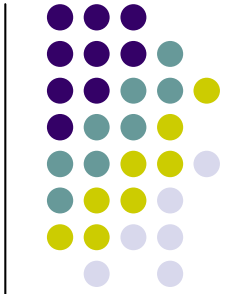
- The lookup process is iterative: everything is controlled by the initiator node
  1. query in parallel the α (α = 3) nodes closest to the query ID
  2. nodes return the k (k = 20) nodes closest to the query ID
  3. go back to step 1, and select the α nodes from the new set of nodes
  4. Terminate when you have the k closest nodes
- Key lookups are done in a similar fashion, but they terminate when the key is found
  - the requesting node caches the key locally.

Kademlia – Other operations



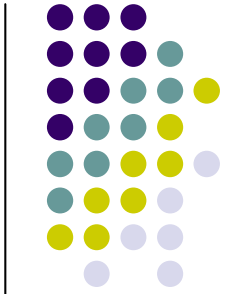
- Refresh
  - periodically, all k-buckets are refreshed by making a query for a value within the bucket
- Node Joins
  - contact a participating node and insert it in the appropriate bucket
  - perform a query for your own ID
  - refresh all buckets

Kademlia – Lookups



- Invariant: If there exists some node with ID within a specific range then the k-bucket is not empty
  - if the invariant is true, then the time is logarithmic
    - we move one bit closer each time
- Due to refreshes the invariant holds with high probability

Kademlia - Properties



- Easy table maintenance. Tables are updated when lookups are performed
  - due to XOR symmetry a node receives lookups from the nodes that are in its own table
- Fast lookup by making parallel searches
  - at the expense of increased traffic.