Content-Addressable-Network (CAN)

- CAN: Internet Scale Hash table
- Interface
  - insert(key, value)
  - value = retrieve(key)
- Idea: associate to each node and item a unique coordinate in an d-dimensional Cartesian space.
- Properties
  - scalable
  - operationally simple
  - good performance
CAN: basic idea (cont.)

- retrieve \( K \)

CAN (Solution)

- virtual Cartesian coordinate space
- entire space is partitioned amongst all the nodes
  - every node "owns" a zone in the overall space
- abstraction
  - can store data at "points" in the space
  - can route from one "point" to another
- point = node that owns the enclosing zone

CAN (Simple example 2-dimension)

1

CAN (Simple example cont.)

1 2

CAN (Simple example cont.)

1 3
2

CAN (Simple example cont.)

1 3
2 4
node I::insert(K,V)
(1) a = h_x(K)
b = h_y(K)
y = b
x = a

route(K,V) -> (a,b)
node I::insert(K,V)
(1) a = h_x(K)
   b = h_y(K)
(2) route(K,V) -> (a,b)
(3) (a,b) stores (K,V)

node J::retrieve(K)
(1) a = h_x(K)
   b = h_y(K)
(2) route "retrieve(K)" to (a,b)

Data stored in the CAN is addressed by name (i.e. key), not location (i.e. IP address)

Question: What is missing in the procedure?

Routing!
3) I routes to (p,q), discovers node J

4) Split J’s zone in half... new owns one half

**CAN (Node Failure)**

- Need to repair the space
  - recover database
  - soft-state updates
  - use replication, rebuild database from replicas
- repair routing
  - takeover algorithm
  - when a node fails, one of its neighbors takes over its zone

*Only the failed node’s immediate neighbors are required for recovery*

**CAN (Evaluation)**

- **Scalability**
  - For a uniformly partitioned space with n nodes and d dimensions
    - per node, number of neighbors is 2d
  - average routing path is \((d^{n/d})/3\) hops (due to Manhattan distance routing, expected hops in each dimension is \(\text{dimension length} \times 1/3\))
  - Can scale the network without increasing per-node state
  - Chord/Plaxton/Tapestry/Buzz
    - \(\log(n)\) nbrs with \(\log(n)\) hops
  - Load balancing
    - overloaded node replicates popular entries at neighbors
  - Robustness
    - no single point of failure
    - Can route around trouble

**CAN (Improvements)**

- Topologically-sensitive CAN construction
  - distributed binning
  - Goal
    - bin nodes such that co-located nodes land in same bin
- Idea
  - well known set of landmark machines
    - each CAN node, measures its RTT to each landmark
    - orders the landmarks in order of increasing RTT
- CAN construction
  - place nodes from the same bin close together on the CAN