A Scalable Content-Addressable Network

In Proceedings of ACM SIGCOMM 2001

S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker

Presented by L.G. Alex Sung

9th March 2005 for CS856

Outline

- CAN basics
- Improving the basic CAN
- Evaluations
- Extensions
- Comment
- Discussion

CAN Basics

- Virtual d-dimensional coordinate space
- Each node holds a zone
- Key is hashed to a point P located in a zone hold by a node
  - Insertion
  - Lookup
  - Deletion

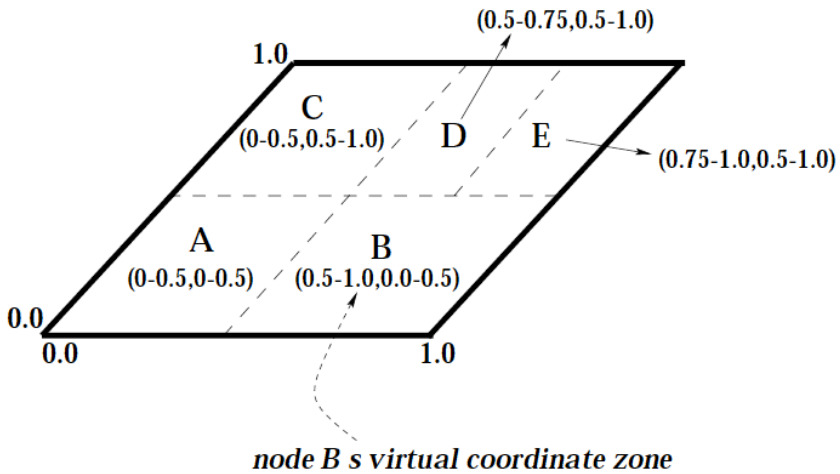


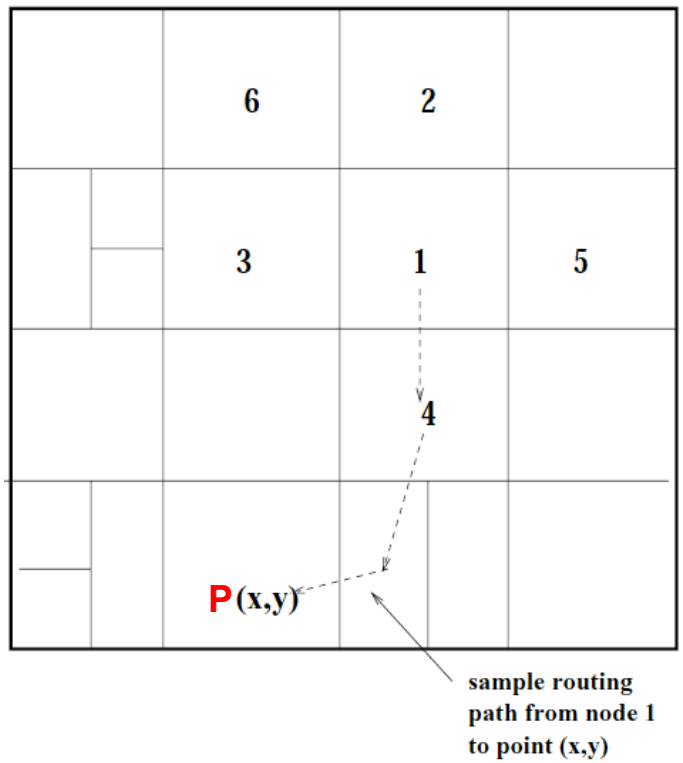
Figure 1: ***Example 2-d space with 5 nodes***

Finding the key

- Location of Key K → Hash(K) → Point P
- By looking at the routing table of neighbours'
  - IP addresses
  - virtual coordinate zones
- Determine the neighbour with the closest coordinate to P
- Greedily forward the msg[P(K), dst_coordinates] through that neighbour

Routing illustrated

- Routing the request until it reaches the node in which zone P lies

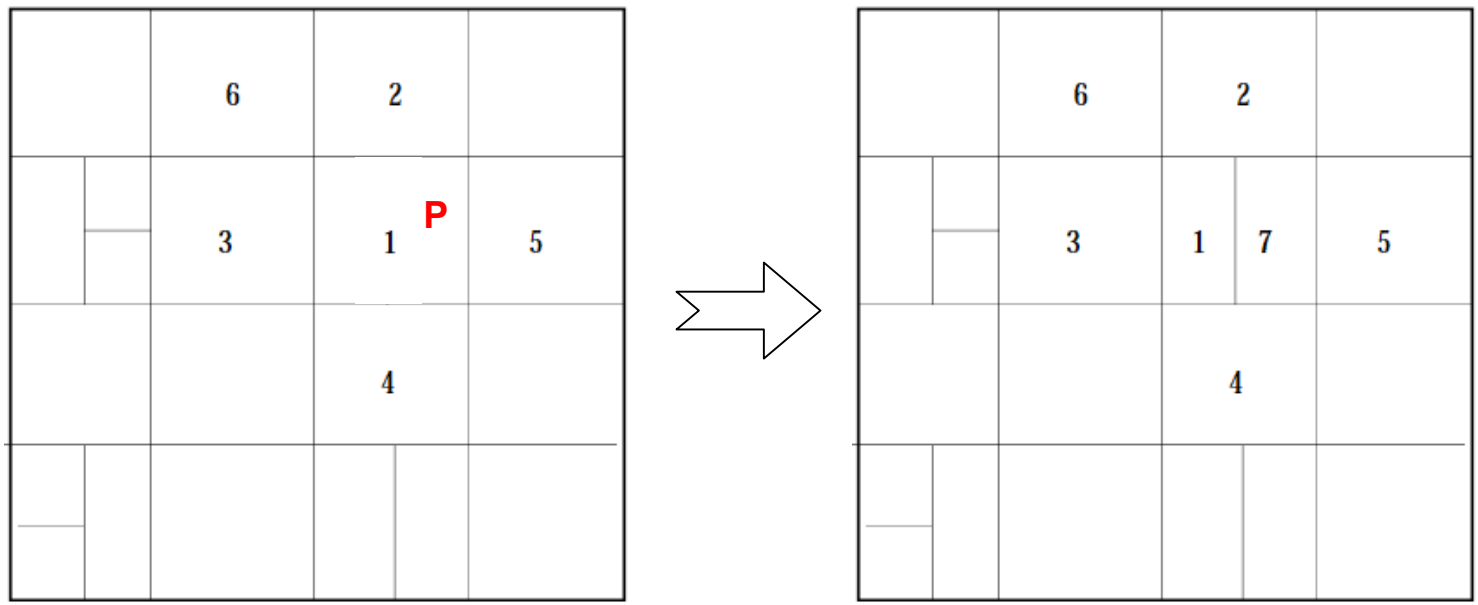


Joining CAN

- Find a node currently in the system (by DNS)
  - Ask for IPs of some other CAN nodes
- Randomly choose a point P in the space
- Send a JOIN request to the CAN node at point P through any CAN node
- Current occupant of point P splits its zone
- Being handed over
  - the key-value pairs of that zone
  - IP addresses and coordinates of neighbours
- Inform all old node's neighbours

Joining illustrated

- Randomly find a point; split that zone; get the keys and inform the neighbours



Leaving CAN

- Departure
  - Goal: give the keys to a neighbour
  - Combine the zone with a neighbour to form a valid single zone
  - OR temporarily hand over to the neighbour with the smallest zone
- Node Failure
  - Identified by prolonged absence of update message
  - The zone will be take over by the neighbour with the smallest zone volume

Improving the basic CAN

- Goal: reduce lookup latency
  - Nodes can be physically far away

The tradeoff

- (+) higher routing performance
- (+) system more robust
- (–) higher per-node states
- (–) higher system complexity

Multiple independent coordinate spaces (Realities)

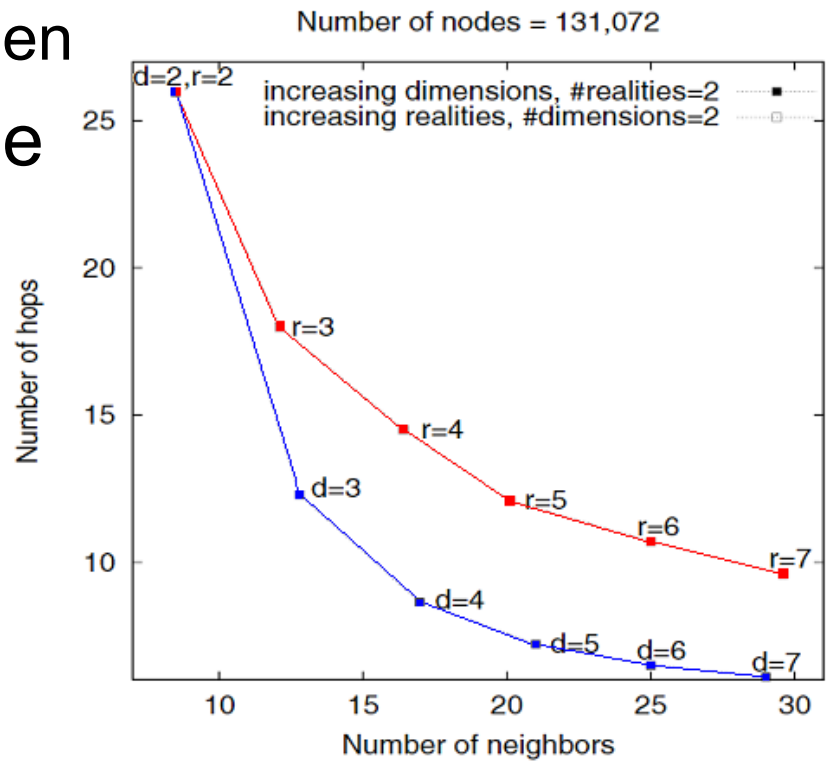
- Allocation of multiple zones per node
  → each zone in a different reality
  (Hash tables are replicated on every reality)
- Route to the neighbor closest to destination in all realities.
- (+) lower path length and path latency
- (+) higher data availability
- (+) routing fault tolerance
- (–) more states per node

Multi-dimensioned coordinate spaces

More dimensions → more neighbours per node

- (+) routing fault tolerance
  → more paths can be chosen
- (–) more states per node
  → routing table

Multi-dimension is better.



Refinement of CAN routing metrics

- Goal: reduction of per-hop latency
- When selecting the next hop
  - take into account the RTT
  - and not just closer coordinate
- Simulation results: 24-40% improvement

Overloading coordinate zones

- When joining, zone sharing (if < MAXPEERS) instead of splitting
- More state info: neighbour list + peer list
- Neighbour selection by lowest measured RTT
- Hash tables: replication vs. partitioning
  - (+) higher data availability
  - (–) need consistency mechanism
  - (–) larger size of data stored
- (+) lower path latency
- (+) higher fault tolerance
- (–) higher system complexity
- (–) additional control traffic

Multiple hash functions

- Mapping a single key to multiple nodes (replication) → parallel queries
- (+) lower query latency
- (+) higher data availability
- (–) larger size of the <key, value> database
- (–) higher query traffic

Topologically-sensitive CAN construction

- Node insertion based on RTT from landmarks (instead of random insertion)

- (+) lower path latency
- (–) uneven load distribution

  → load balancing needed

Uniform partitioning

- Volume-based zone splitting
- (+) some form of load balancing
  → each zone holding similar # of keys
- (–) "Hot spot" problem: some <key, value> pairs are more popular
  → network congestion

Caching & Replication for “hot spot” management

- Caching recently accessed keys (which belongs to other nodes)
- Replication: actively pushing popular keys to neighbours
- (+) higher data availability
- (+) lower query latency
- (+) load balancing
- (–) cache management

Evaluation

- Critical factors:
- 1. increase in # of dimensions d
  → reduction of path length
- 2. Use of RTT-weighted routing
  → optimization of next-hop forwarding
  → reduction of path latency

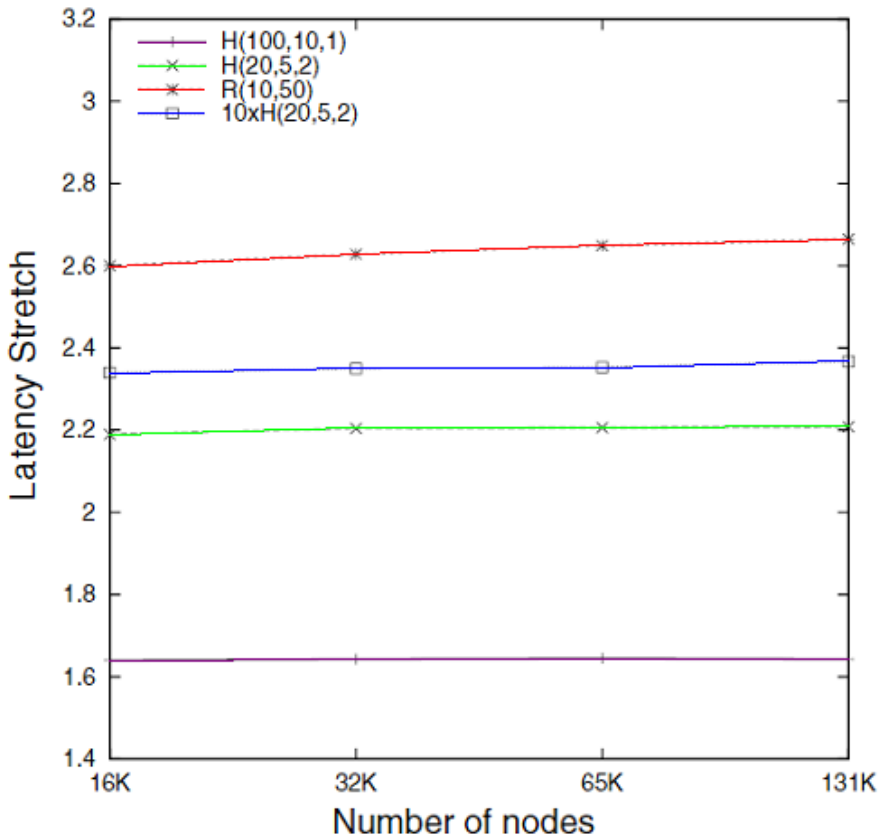
Metric	"bare bones" CAN	"knobs on full CAN"
path length	198.0	5.0
# neighbors	4.57	27.1
# peers	0	2.95
IP latency	115.9ms	82.4ms
CAN path latency	23,008ms	135.29ms

Parameter	"bare bones" CAN	"knobs on full" CAN
d	2	10
r	1	1
p	0	4
k	1	1
RTT weighted routing metric	OFF	ON
Uniform partitioning	OFF	ON
Landmark ordering	OFF	OFF

Evaluation (2)

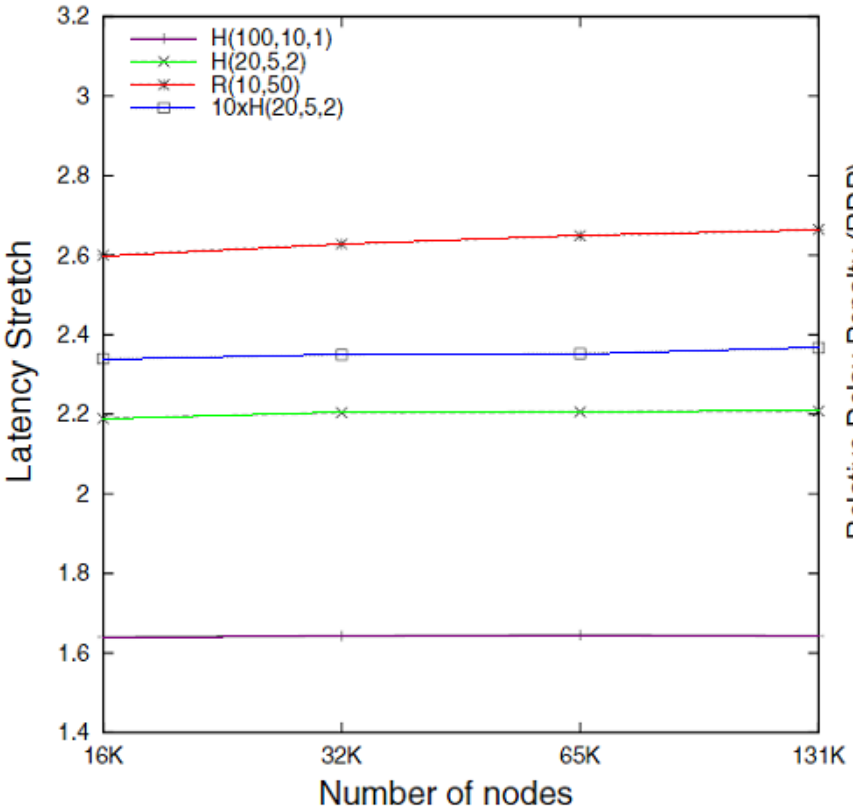
- Effect of link delay distribution on CAN latency
- latency stretch = CAN latency/IP latency

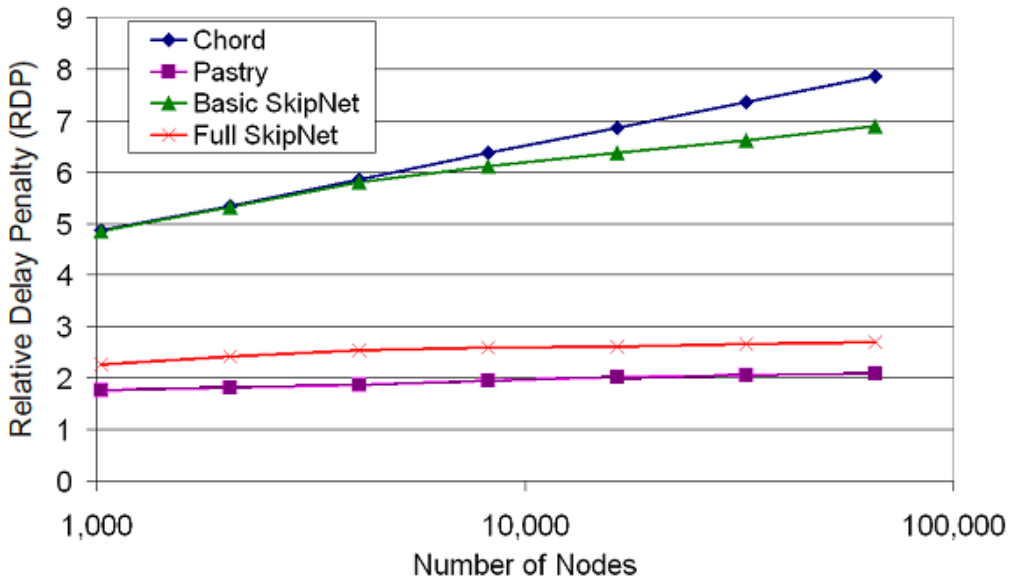
1. Increase in # of nodes
   → slow increase in latency stretch
2. Random delay
   → the largest latency stretch
3. Larger backbone
   → lower density of CAN nodes
   → less effect of RTT-weighted routing
   → degraded gains



CAN vs other DHTs

- RDP = Overlay network latency / IP latency
- CAN seems to be better than Full SkipNet



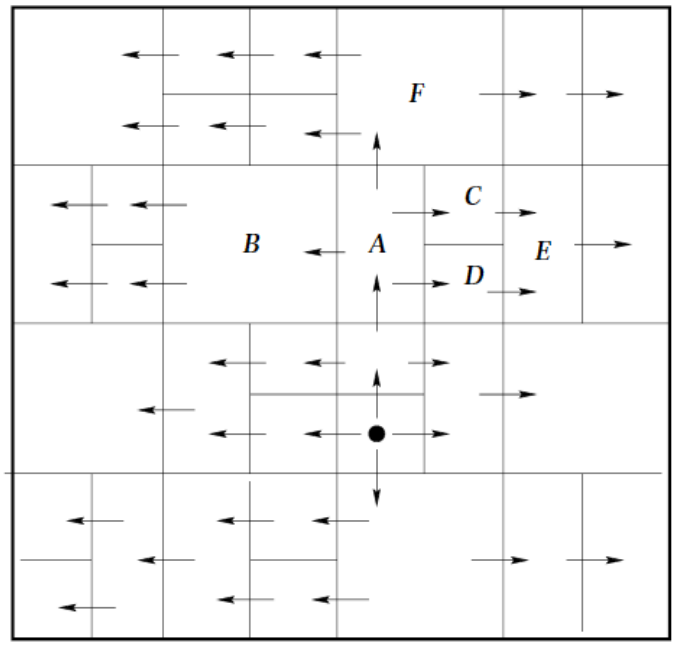


Source: the SkipNet paper

CAN Extensions

- Application level multicasting [3]
- Spatial Data Query Support [2]

Fig. 4. *Directed Flooding over the CAN.*



Summary

- CAN
  - an Internet-scale hash table
  - potential building block in Internet applications
- Scalability (basic CAN)
  - O(d) per-node state
  - $O(n^{1/d})$ average path length
- Low-latency routing
  - simple heuristics help a lot
- Robust
  - decentralized, can route around trouble

Comment

Strength

- Pioneer work in DHT (same as Chord)
- Intuitive presentation of formal concepts
- Taken into account the RTT in neighbour selection

Weakness

- High computational and memory requirement (it's a trade-off)
  - The CPU and memory usage statistics are not given

Discussion

- Network capacity is not taken into account when assigning keys. (hot-spot problem)
  - Can we divide the zone based on **networking capacity** (rather than zone size only)?
  - Can we **predict** the probability of congestion?
  - How many **levels of replication** is reasonable?
- Many optimizations involve replicating the (*K*,*V*) pairs (and require more CPU cycles)
  - **Replication limit** under reasonable assumptions?
  - How about **CPU limit**?
  - Which is cheaper: network delay or CPU/Memory?

Discussion - DHTs

- Internet users are **heterogeneous**. Memory and CPU power are relatively cheaper than routing cost.
  - Would it be better to **build CAN as a service** to lower the heterogeneity and select the best **balance point** for optimizations vs CPU power & memory requirement?
  - What can other DHT schemes do to reduce path latency? Which CAN optimization can be applied?
- Shall we give priority to maintenance or routing?
- What does O(log N) really mean? Would the **average IP network latency** be more important? What metrics shall we use to compare DHTs?

References

1. N.J.A. Harvey, M.B. Jones, S. Saroiu, M. Theimer, and A. Wolman, SkipNet: A Scalable Overlay Network with Practical Locality Properties, In *Proc. 4th USENIX Symp. on Internet Tech. and Syst.* (USITS), 2003.
2. Roger Zimmermann, Wei-Shinn Ku, and Haojun Wang. Spatial Data Query Support in Peer-to-Peer Systems. COMPSAC 2004.
3. Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-Level Multicast Using Content-Addressable Networks.
4. Zacharias Boufidis. http://www.srdc.metu.edu.tr/webpage/seminars/p2p/CAN.ppt