A Scalable Content-Addressable Network

SIGCOMM’01

Introduction

- Locating queried files is essential to emerging peer-to-peer file sharing systems.
  - Napster, Gnutella, Ezpeer, Kara, …
- The Content-Addressable Network (CAN) proposed herein is a scalable indexing mechanism with additional robustness and low-latency properties.

Applications

- Large scale storage systems
  - OceanStore, Farsite, Publius
- Name resolution
  - DNS
- Grass-roots Content Distribution
  - "RAID meets the Web"

Features

- Operations with hash tables
  - Insertion, lookup, deletion of (key, value) pairs
- Zone of node
  - Each CAN node holds a chunk(zone) of the entire hash table and information of adjacent zones.
- Routing of requests
  - Requests for a particular key are routed to the CAN node whose zone contains that key.
- CAN is completely distributed, scalable, and fault-tolerant.

Infrastructure

- A *d*-dimensional Cartesian coordinate space dynamically partitioned into zones

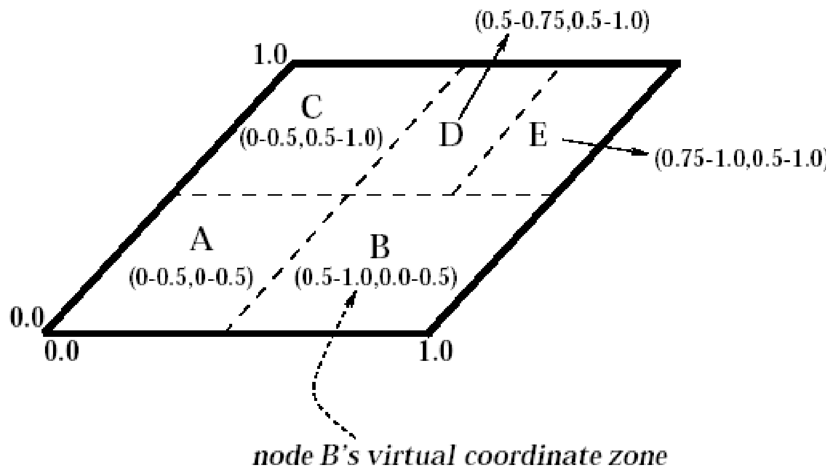


Figure 1: ***Example 2-d space with 5 nodes***

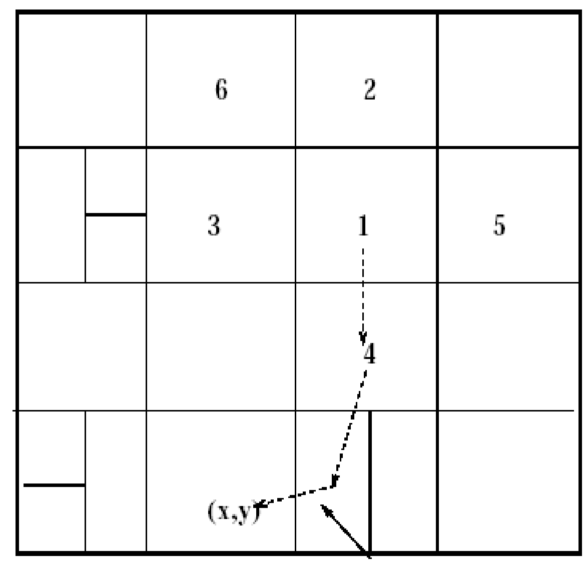
Infrastructure

- For a (key, value) pair (K, V):
  - K is mapped onto point P by hash function
  - (K, V) is stored in the node whose zone contains P
- To retrieve the (K, V) pair:
  - Node i applies same hash function to map K onto P
  - Retrieve (K, V) from node i if P belongs to zone(i)
  - Else, route K to the node whose zone contains P

Routing

- Straight line from source to destination
- Coordinate routing table
  - IP addresses and coordinate zone of neighbors

Neighbors in d-dimensional coordinate space:
•Overlap along d-1 D
•Abut along one D



CAN Construction

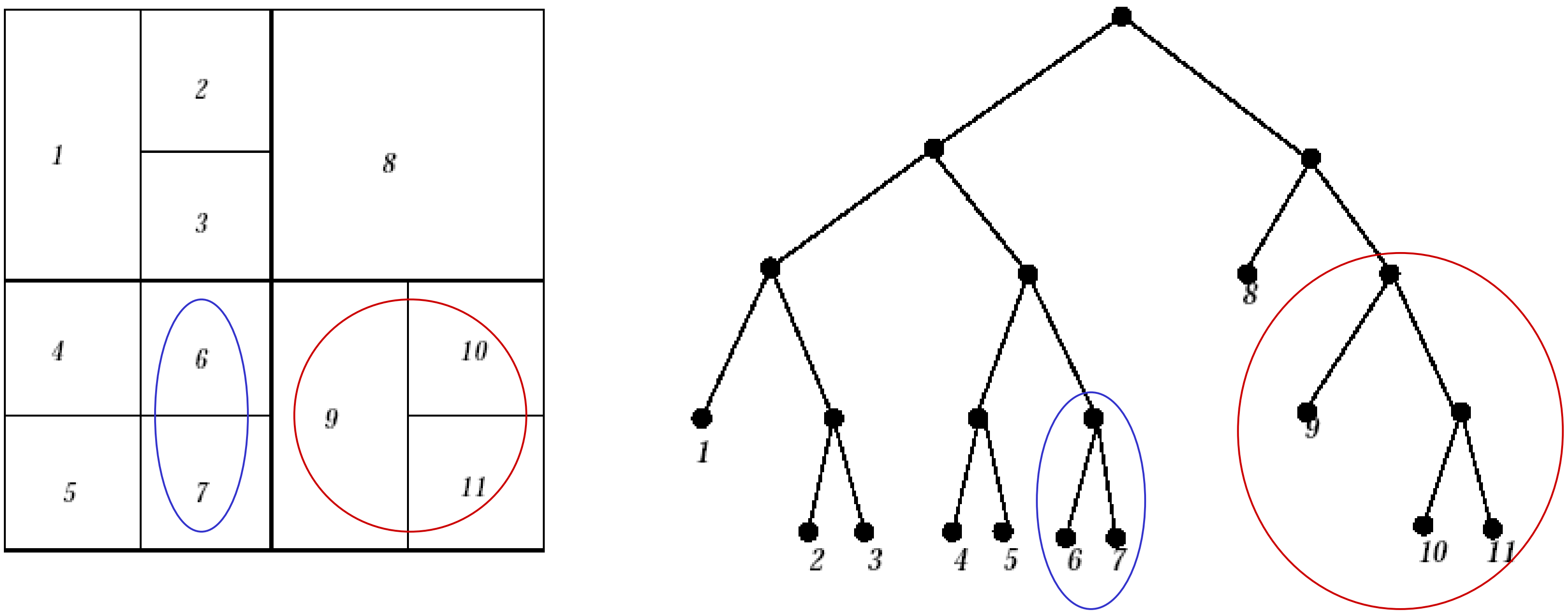
- When a new node joins:
  - – Bootstrap
    - DNS->bootstrap node->CAN nodes
  - – Find a zone
    - random point->JOIN->zone splitting->pairs transfer
  - – Join the routing
    - Neighbor adjustment->UPDATE
  - – O(d) regardless of N

CAN Maintenance

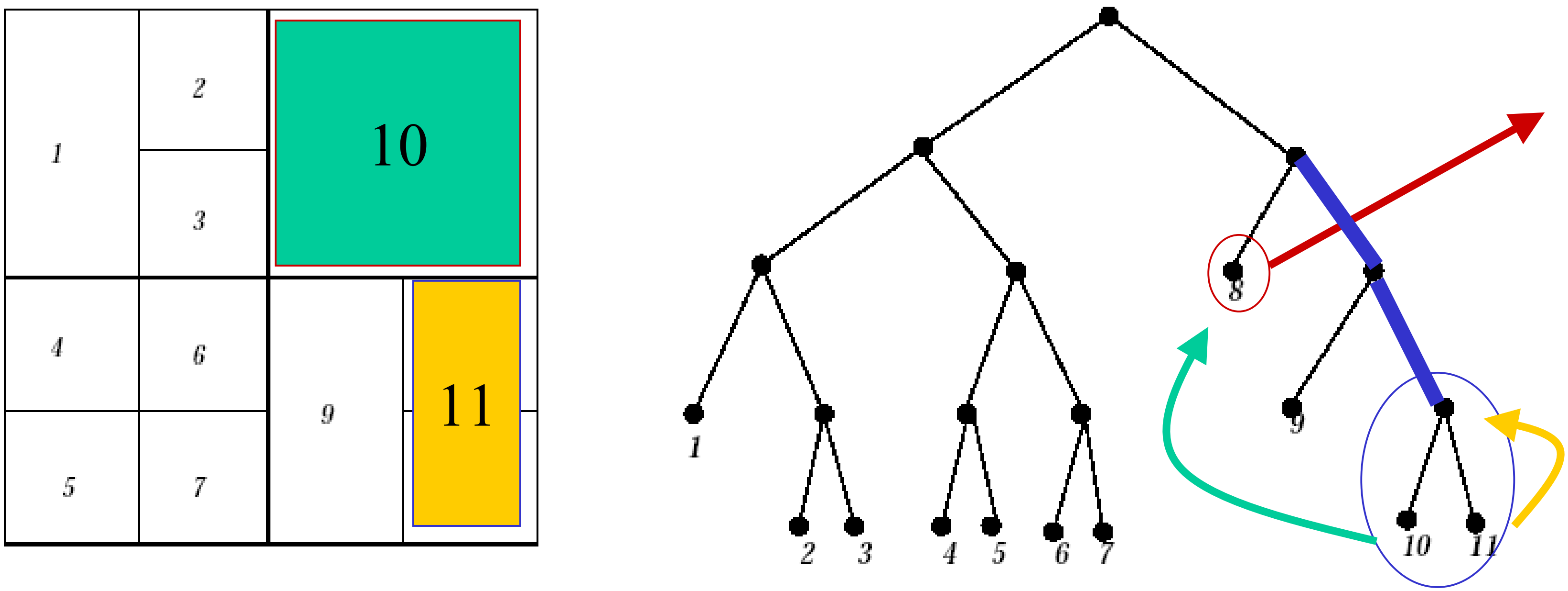
- When a node leaves:
  - Either neighboring node or node with least-size zone takes over the left zone.
- When failures occur:
  - Periodic UPDATE messages
  - Takeover algorithm
    - TAKEOVER message
    - Takeover timer

CAN Maintenance

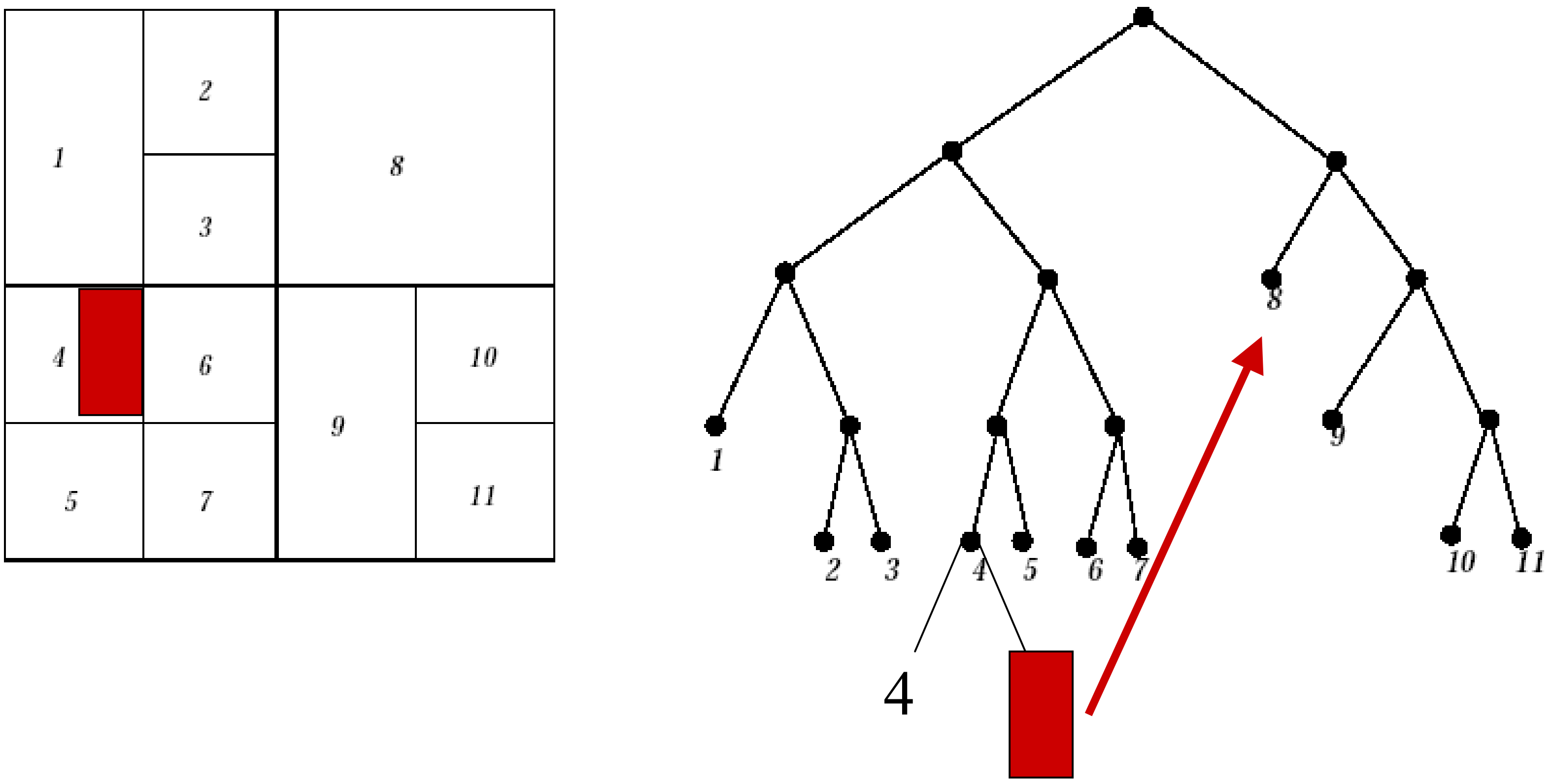
- Background zone reassignment



Zone Reassignment

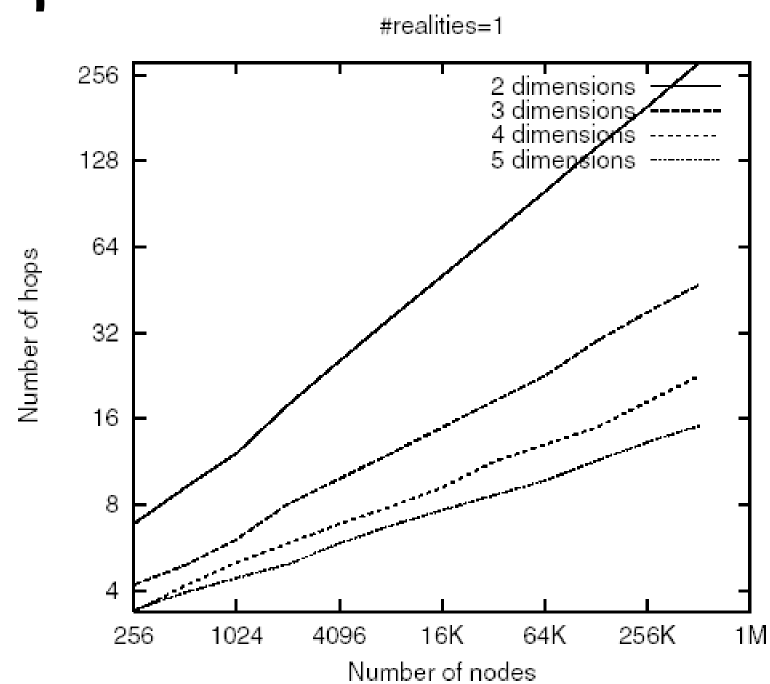


Zone Reassignment



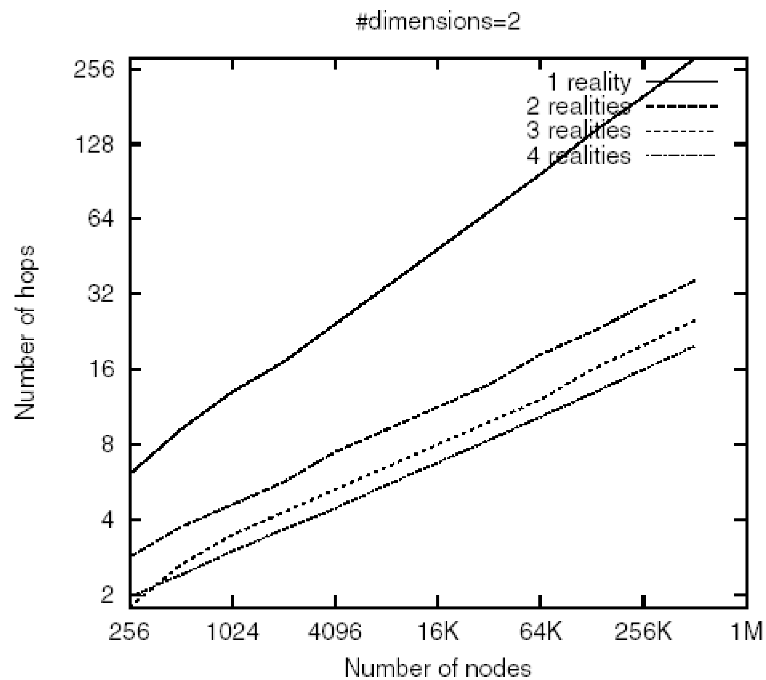
Multiple Dimensions

- Path length: $O(dN^{1/d})$
- Neighbor: O(d)
- Total latency: >
- Fault tolerance: <
- Routing table: <

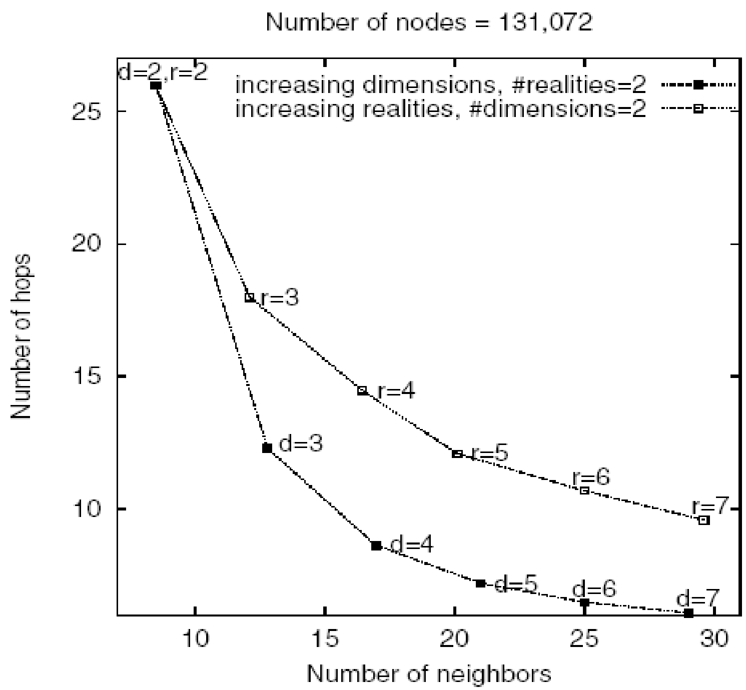


Realities

- Multiple coordinate spaces
- Path length: >
- Neighbor: O(rd)
- Total latency: >
- Fault tolerance: <
- Availability: O(r)



Dimensions v.s. Realities



Routing Metrics

- Round Trip Time (RTT)
- Total latency: >

Number of dimensions	Non-RTT-weighted routing (ms)	RTT-weighted routing (ms)
2	116.8	88.3
3	116.7	76.1
4	115.8	71.2
5	115.4	70.9

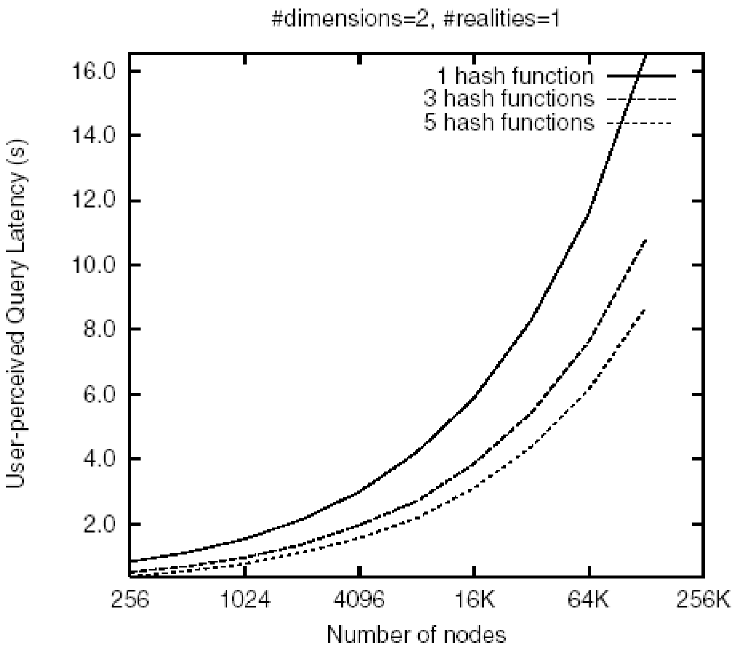
Zone Overloading

- Path length: O(1/p)
- Neighbor: O(pd)
- Total latency: >
- Fault tolerance: <
- Availability: O(p)

Number of nodes per zone	per-hop latency (ms)
1	116.4
2	92.8
3	72.9
4	64.4

Multiple Hash Functions

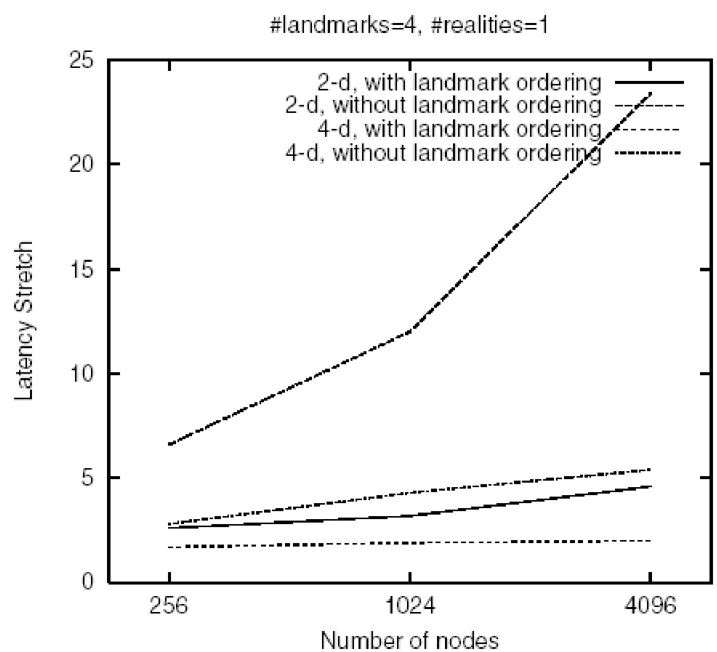
- Total latency: >
- Availability: O(k)



CAN over IP Topology

- m landmarks (DNS servers)
- m! ordering by RTT to m landmarks
- m! equal size portions
  - m at 1st D, m-1 at 2nd D, …
- A new node chooses random point P in the portion associated with its landmark ordering.

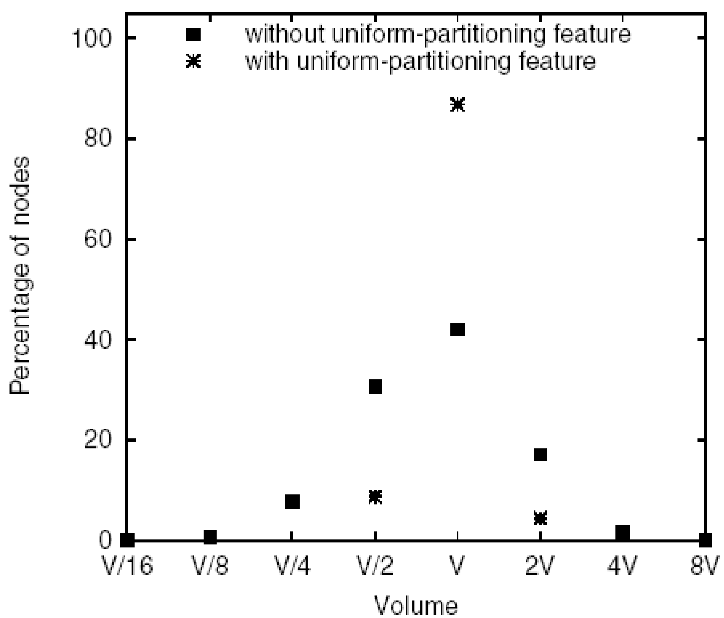
CAN over IP Topology



- Uneven distribution (interesting!)

Uniform Partitioning

- Volume balancing



Performances

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

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