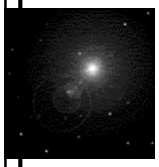


A Course on Foundations of Peer-to-Peer Systems & Applications

CS 6/75995
Foundation of Peer-to-Peer

Applications & Systems

Kent State University

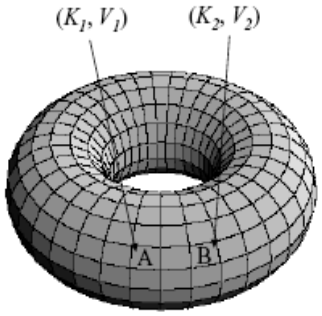
Dept. of Computer Science
www.cs.kent.edu/~javed/class-P2P08/

Scalable Content Addressable Network (CAN)

- Overview:
  - PhD Thesis (Sylvia Paul Rathnasamy, Scott Shenker, Ion Stoica, Barkeley, December 2002)
- Topology
  - N-dimensional Torus
  - Partitioning of Space
- Routing
- Node Arrival
  - Bootstrapping
  - Finding a Zone
  - Joining the Routing (Route Table Updates)
- Node Departure
  - Identification of Takeover Node
  - Recovery Algorithm
- Performance Analysis
- Evaluation
  - Stability
  - Robustness
  - Load balancing

CAN
Topology

Torus

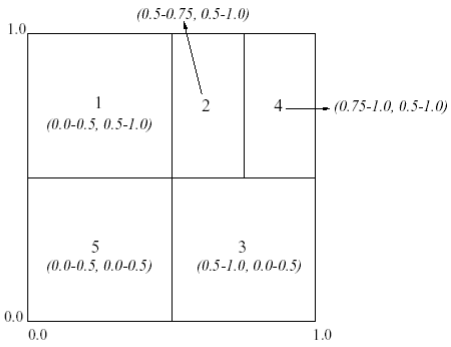


2-torus

- The topology of CAN centers around a virtual d-dimensional Cartesian coordinate space on a ***d-torus***. At any point in time, the entire coordinate space is dynamically partitioned among all the nodes in the system such that every node "owns" its individual distinct zone within the overall space.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Topology: Example

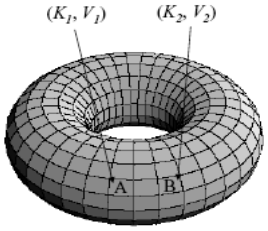


- A 2-dimensional [0,1]x[0,1] coordinate space with 5 partitions owned by five nodes.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Storage

- To store key value pair (K_1, V_1),
  - K_1 mapped to point P_1 using a deterministic hash function
  - (K_1, V_1) stored at the node N that owns the zone containing P_1

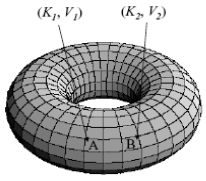


2-torus

FOUNDATION OF PEER-TO-PEER SYSTEMS

Search & Retrieval

- To retrieve a key K_1 mapped to point P_1 a requesting node R uses the same deterministic hash function to determine point P_1 and retrieve the (K_1, V_1) from node N that owns the zone containing P_1.
- If R or one of its neighbor own the point then it is retrieved immediately, otherwise it needs to be obtained by routing the request through the CAN topology.



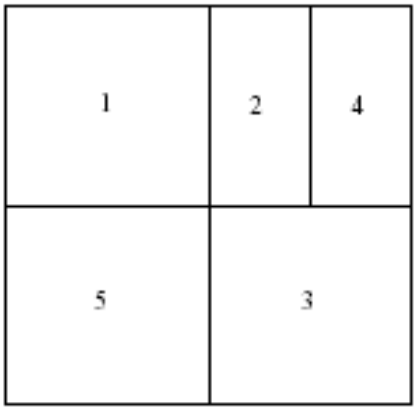
2-torus

FOUNDATION OF PEER-TO-PEER SYSTEMS

Topology

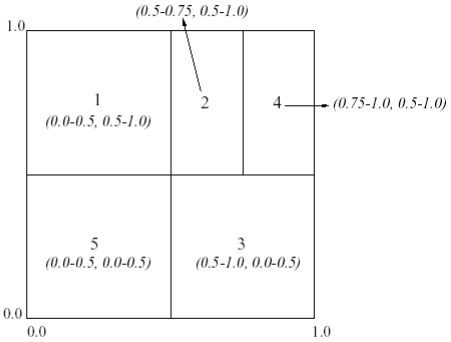
- Initially the entire space is one zone and is owned by one node.
- Each time a new node joins, an existing zone is split into two equal halves, one of which is assigned to the new node, the other retained by the previous owner of the whole zone.
- The split is done by following a well-known ordering of the dimension in deciding along which dimension a zone is to be split, so that zones can be remerged.
- As nodes join the zones are split along X_1, X_2, X_3, …X_d, and then again along X_1, X_2… dimensions.

Zone Partitioning of a 2-D Space



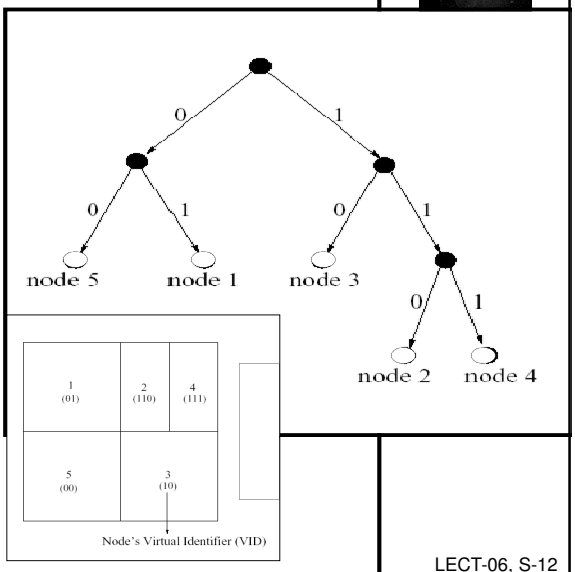
- In the example for a 2-d space, a zone would first split along X dimension, then the Y, then X again.

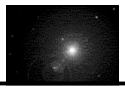
The Zones Ranges After Partition



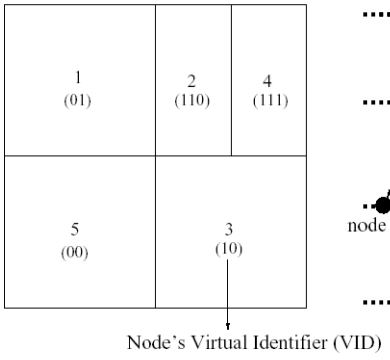
Partition Tree

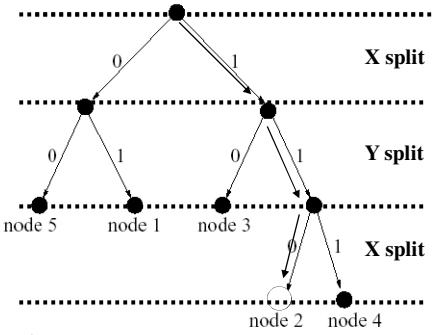
- Each existing zone can also be thought as a leaf of a "binary partition" tree.
- The leaf nodes are the exiting zones. The internal nodes are the zones those previously existed but were split.
- The edges are labeled. An edge connecting a parent and child zone is labeled "0" if the child occupies the lower half of the dimension along which the split occurred. It is labeled "1" for the upper half.
- Each node in CAN is addressed with a **virtual identifier** (VID)- the binary string representing the path from root to leaf node.





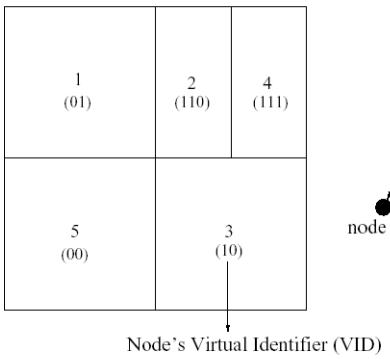
A Walk in a Partition Tree

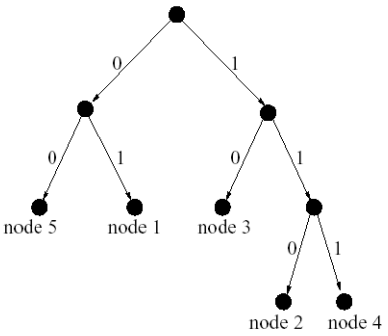




- The path from root to zone 2 is (110). So the first edge label tells is zone 2's zone is inside range [.5,1] along X dimension, [.5,1] along Y dimension, and then [.5,.75] along X dimension.

5- Zone and Partition Tree





- QUIZ1: can you show the partitioning if the dimension orders are reversed?

CAN
Node Arrival

Peer Joining

- A new peer must find a peer already in CAN.
  - The IP address can be found by some external bootstrap mechanism. Such as a named CAN can be associated with a DNS which can resolve into a server with a partial list of CAN nodes currently alive in the named CAN.
- New peer be allocated a unique zone on Torus.
- New peer should find its neighbors.
- Neighboring peers of the split zone must update their routing table that should now include the new peer.

Finding a Zone

FOUNDATION OF PEER-TO-PEER SYSTEMS

- The new peer ***n*** randomly chooses a point P in the space and sends a JOIN request destined for point P.
- The message is sent to an existing CAN peer ***s*** which is then routed to the peer node ***m*** currently responsible for the point P.
- On receiving the JOIN message, ***m*** the owner of the zone splits its own zone with the new node (basic scheme).
- Alternately (in uniform partition scheme) ***m*** knows its neighbors so it compares the zone volumes of its neighboring nodes and finds the node ***m'*** with the largest volume.

(continued…)

- QUIZ: if it is strictly necessary to split m?

Finding a Zone

FOUNDATION OF PEER-TO-PEER SYSTEMS

(continued…)

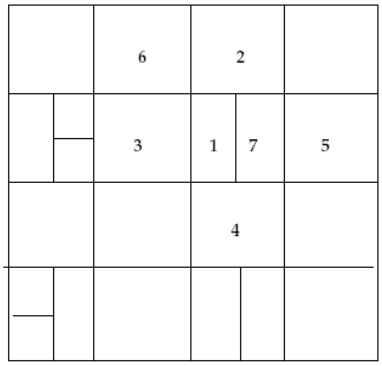
- Peer ***m*** splits its zone into half; the occupant retains the lower half zone (along the dimension of split) and assigns the other half zone to the new node ***n***.
- The occupant node ***m*** appends a'0' to its original VID and the new node is given a VID of occupant node's original VID appended with a'1' at the end.
- The new node ***n*** is informed of its new VID.
- The stored data (key,value) pairs from the half zone is transferred from ***m*** to new node ***n***.

- QUIZ: Is it strictly necessary for m to retain lower half?

Definition: Coordinate Neighbor

FOUNDATION OF PEER-TO-PEER SYSTEMS

- In a d-dimensional coordinate space, two nodes are **coordinate neighbors** if their coordinate spans overlaps along d-1 dimensions and abut along one and only one dimension.



- Example1: Is zone 6 is neighbor of 3? Yes because its coordinate zones overlap with 6 along x axis and abuts along y axis.
- Example2: Is zone 2 is neighbor of 3? No because their coordinate zones abut along both X and Y axes.
- QUIZ: Argue why 3-7 are not neighbors?

Finding Neighbors

FOUNDATION OF PEER-TO-PEER SYSTEMS

- Having obtaining a zone the new node **n** must learn the IP addresses of its **coordinate neighbor** set.
- A new node **n**'s zone is derived by splitting the previous occupant **m**'s zone; consequently the new nodes ***coordinate neighbor*** set is a subset of the previous occupants neighbors.
- Previous occupant **m** updates its ***coordinate neighbor*** set to eliminate those nodes that are no longer neighbors, plus the new node.
- New peer **n** copies the ***coordinate neighbor*** set of **m** and updates its ***coordinate neighbor*** set to eliminate those nodes that are no longer neighbors, plus the previous node.

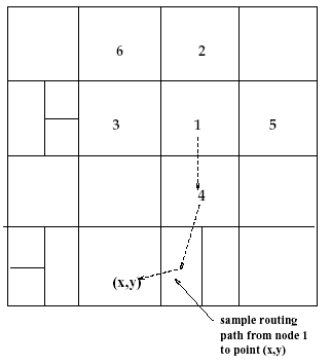
Updating Neighbors

- Both the old and new nodes inform its neighbors about the reallocation of zones.
- The message contains its currently assigned zones and is sent to all the neighbors.
- Neighbors upon receiving the message updates their table.
-
- Every node in the system also sends an immediate update message, followed by periodic refresh.

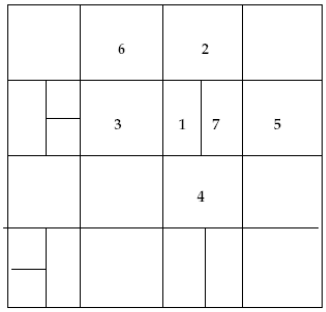
FOUNDATION OF PEER-TO-PEER SYSTEMS

Example of Join

- A New node 7 wants to join. Zone 1 is split. Now old node 1 and new node 7 both updates their neighbor set.



1's coordinate neighbor set = {2,3,4,5}
7's coordinate neighbor set = { }



1's coordinate neighbor set = {2,3,4,7}
7's coordinate neighbor set = {1,2,4,5}

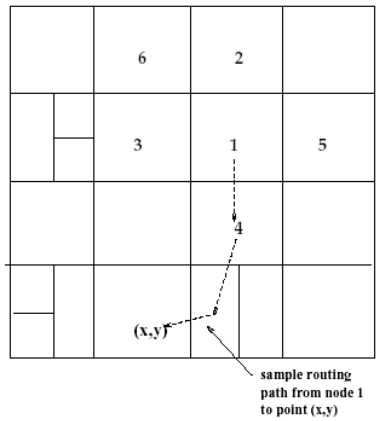
FOUNDATION OF PEER-TO-PEER SYSTEMS

CAN
Routing

Routing

- A CAN peer maintains a coordinate routing table that holds the IP address and VID of each of its coordinate neighbors.
- An intermediate node can uniquely determine the zone of its neighbors from VID.
- A CAN message includes the destination coordinate point P.
- It uses ***geometric greedy forwarding***. It forwards the message to one of the neighbors with coordinates closest to P.

Example Routing



1's coordinate neighbor set = {2,3,4,5}
7's coordinate neighbor set = { }

- From node-1 of [2,3,5,4] 4 is closest to (x,y).

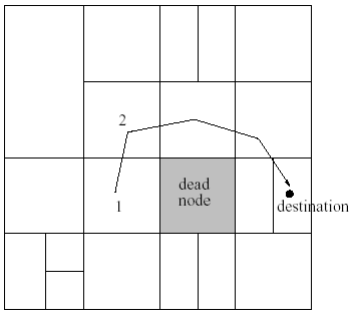
FOUNDATION OF PEER-TO-PEER SYSTEMS

Routing Distance

FOUNDATION OF PEER-TO-PEER SYSTEMS

Routing Redundancy

- There are many paths from any node to any point in a Cartesian space. So Even if one or more neighbors were to crush, a node routes along the next best available path.



CAN
Node Departure

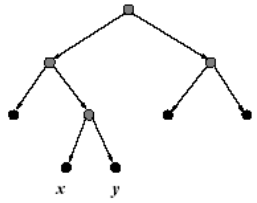
Peer Departure

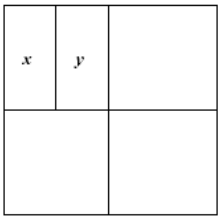
- A departing peer hands over its zone to a ***takeover peer***.
- If the zone of the takeover peer can be merged with the departing peer's zone, a single valid zone is created.
- The takeover peer is handed over all the data (key,value) pairs.
- If, the zones cannot be merged, the takeover peer temporarily handles both the zones. (it can be assigned later to a new incoming node).
- A departing node notifies all its neighbors about is departure and the takeover node notifies all its neighbors about takeovers.

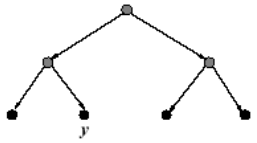
Concept: Takeover Peer

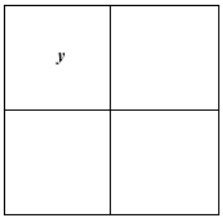
- In the binary partition tree, let the sibling of a departing peer x (a leaf node) is the peer y.
- If y is also a leaf node (its zone did not split), node y becomes the takeover node. It coalesce the two zones (case A)
- If y is not a leaf node anymore (its zone has been further split), perform a depth-first search in the sub-tree rooted at y until a leaf node z is found. Z becomes the takeover node. (case B)
- 'z' retains both the zones since, zones of x and z cannot be simply merged. (case B)

Example Case A



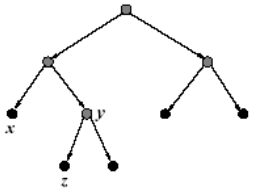


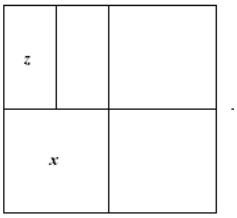


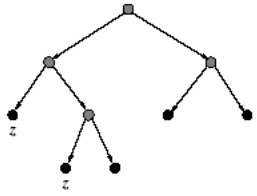


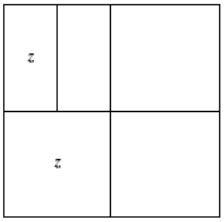
FOUNDATION OF PEER-TO-PEER SYSTEMS

Example Case B









FOUNDATION OF PEER-TO-PEER SYSTEMS

Computing Takeover Node

- A nodes VID is a *l*-bit string stored in a *k* bit identifier, where the leading *l* bit prefix represents the actual VID, remaining *(k-l)* bits are simply zero.
- The numerical value of this *k*-bit string is *val(VID(x)).*
- The *VID distance* (*numerical)* between two nodes is |*val(VID(z))- val(VID(z))*|
- The takeover node z is one of the nodes from neighbor set of departing node x which has numerically closest *VID distance*.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Node Failure

- If a peer fails, the data (key,value) of a zone is lost. However, the remaining peers tries to reconstruct their neighbor table to a consistent state.
- Under normal condition, each nodes sends a periodic update message to its neighbors. The prolonged absence of an update message signals a failure.
- When it detects a failure of a node x, then it deletes the dead node x from its neighbor set and attempts to contact the dead neighbor's takeover peer z.
- (continued..)

FOUNDATION OF PEER-TO-PEER SYSTEMS

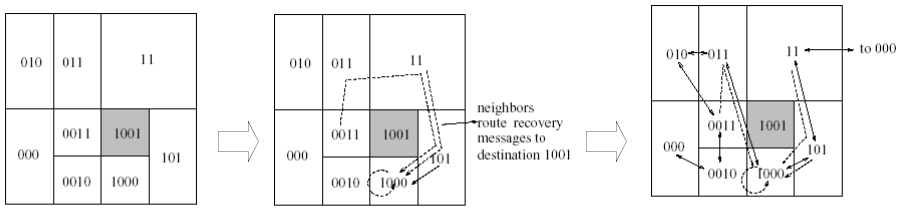
Node Failure

- (continued..)
- It does so by sending a RECOVERY message addressed to x to its neighbor closest to dead node x in terms of VID-distance.
- The message is incrementally routed close to the dead node's VID, and ideally it arrives at the takeover node from which it cannot be forwarded any closer.
- The takeover node is then alerted.
- Since all neighbors of dead node x initiates the RECOVERY message, the takeover node discovers all the neighbors of failed zone.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Example

FOUNDATION OF PEER-TO-PEER SYSTEMS



node 1001 dies

node 1001's neighbors route recovery messages towards VID=1001; recovery messages terminate at takeover node 1000

ordered link list of VIDs maintained in the face of node failures using Chord's stabilization algorithm.

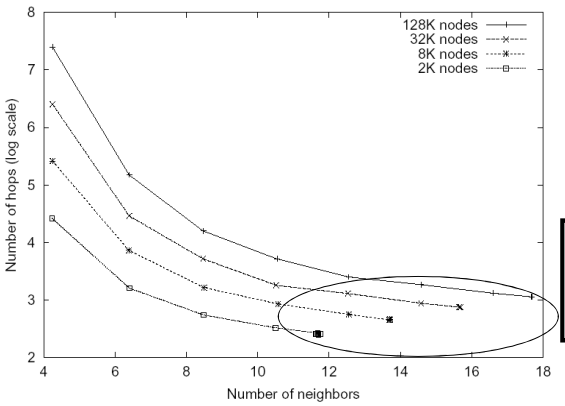
- QUIZ: When a RECOVERY message may not reach to takeover node?

CAN
Performance

Performance

- For a d dimensional Torus with n elements the average path length is $O(d. n^{-1/d}/4)$. However, for P2P Torus the partition is never ideal. Experiments were performed to verify the trend in P2P setup.

Dimensionality & Path Length

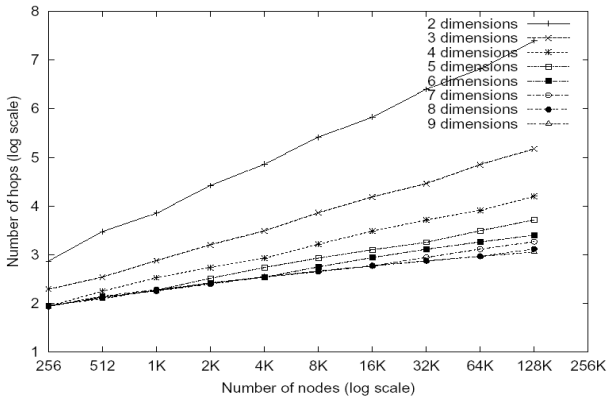


FOUNDATION OF PEER-TO-PEER SYSTEMS

- QUIZ: Can you explain the clustering of data points at the tail ends?

- With d-dimension, each node needs to store 2d states in neighbor set table. But, it also reduces number of hops.

Nodes & Path Length



FOUNDATION OF PEER-TO-PEER SYSTEMS

- With increase of n the hop count do tends to follow the expected $O(d. n^{-1/d})$ analysis.

Routing Resilience with Dimensionality

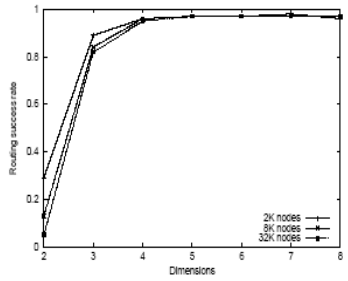


Figure 2.11: *Routing resilience improves with increasing dimensions*

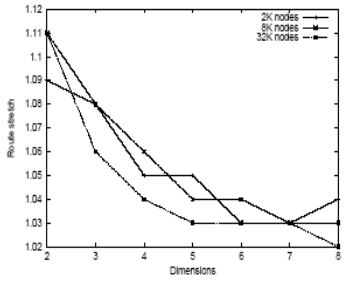


Figure 2.12: *Routing stretch decreases with increasing dimensions*

- Static resilience: node fails. No attempt is made to fix routing. Worst case!
- Increased dimension also increases paths. Thus improves route resilience (25% nodes failed)

FOUNDATION OF PEER-TO-PEER SYSTEMS

Routing Resilience with Node Failure Rate

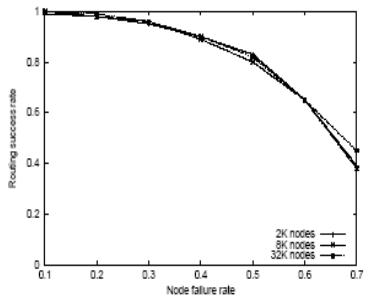


Figure 2.13: *Routing resilience with increasing failure rate*

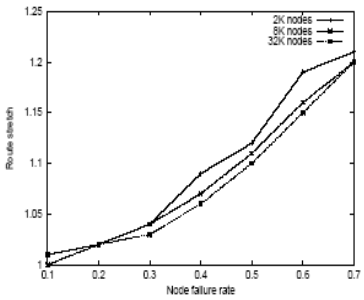


Figure 2.14: *Routing stretch increases with increasing failure rate*

- Even at high node failure nodes (at d=8) the resilience is high.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Load Distribution

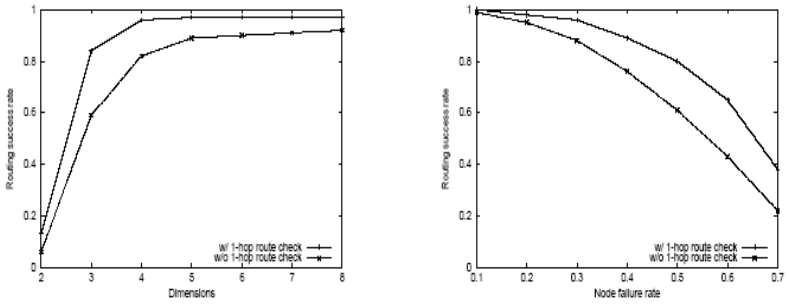


Figure 2.15: *Performance gain with the 1-hop route check for increasing dimensions and node failures*

- With the simple mechanism of 1-hop route check the routing resilience increases significantly.

FOUNDATION OF PEER-TO-PEER SYSTEMS

Load Distribution

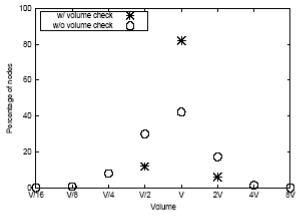


Figure 2.16: *Distribution of zone volumes with and without 1-hop volume check for a CAN with 32,768 nodes and 3 dimensions*

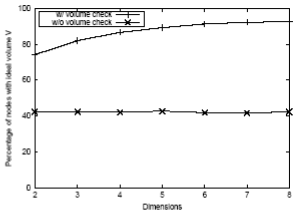


Figure 2.17: *Effect of increasing dimensions on the efficacy of the 1-hop volume check for a CAN with 32,768 nodes*

- Ideally the load per node should be V/n. But in practice it may not be. Shows the distributions with and without 1-hop volume check. (Quiz: why power of two?)

FOUNDATION OF PEER-TO-PEER SYSTEMS