cs 591B1: ALGORITHMIC ASPECTS OF COMPUTER NETWORKS **Spring 2002**

Lecture 4 — February 11, 2002

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Mina Guirguis*

In today's lecture, we discussed CAN, a distributed infrastructure that provides hash table-like functionality on Internet-like scales. For a d-dimensional space partitioned into n equal zones, we derived the average routing path length between two zones as well as how routing can be achieved in such design. In the second half, we discussed CAN's design and its performance.

4.1 CAN as a Hash-Table

Main Idea : CAN divides a virtual d-dimensional Cartesian coordinate space among n individual nodes (peers), where each node stores a zone of the hash table in the form of (key,value) pairs. To store or retrieve a pair $(K1, V1)$, key $K1$ is mapped onto a point P in the coordinate space using a deterministic uniform hash function. Then the node who owns this zone where P lies, will be the one responsible for this pair $(K1, V1)$. Requests for a particular key are routed by intermediate nodes towards the CAN node whose zone contains that key.

4.1.1 Example

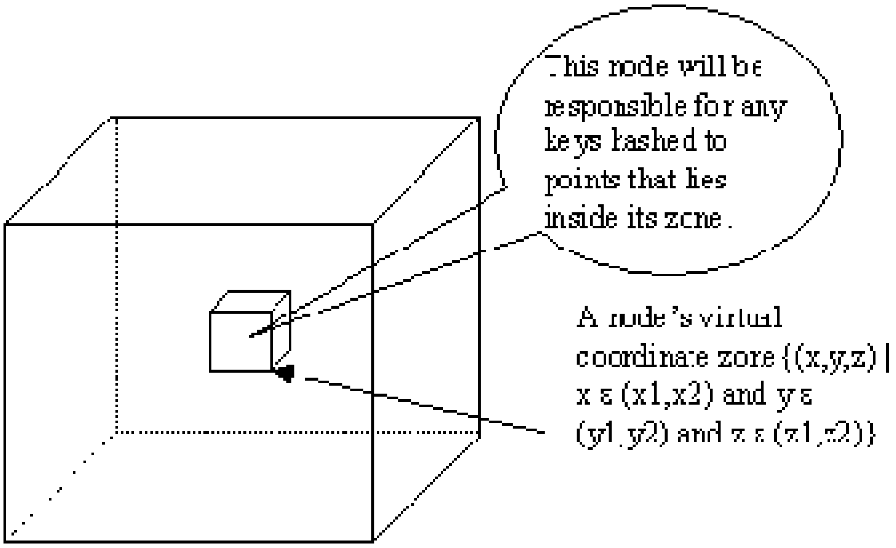


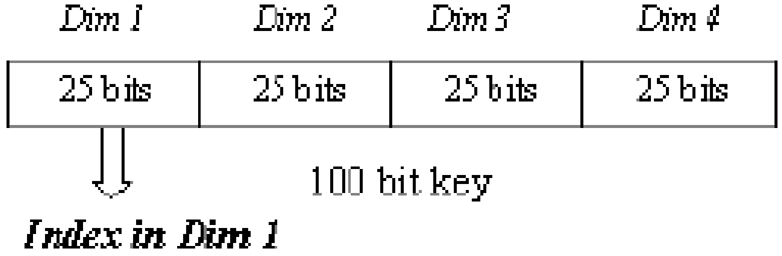
Table 4.1. A CAN node in a 3-dimensional space

4.2 Routing in CAN

It is clear for any d-dimensional space on a d-torus, each node will have $2d$ neighbors (one on each side per dimension). Each CAN node will only maintain state for its $2d$ neighbors. The higher value of d, the shorter is the path between two nodes but with correspondingly the more information state needed to be saved at each node.

4.2.1 Example

Let's assume we have 100 bit keys and a 4-dimensional space. The total number of keys that can be represented and mapped into the space is equal to 2^{100}.This will give us the volume of our universe $U(volume) = 2^{100}$. Let's choose the number of nodes equal to 625, this will give us 5 degrees of freedom (5 nodes can be projected on any given dimension), as $5^4 = 625$. Since we have 100 bit keys and 4 dimensions, then each 25 bits will give us a position in a given dimension indicating which node of the 5 nodes is responsible for this key.



Note that each 25 bits in the key, give us an index inside this dimension which can be calculated by the decimal value associated with these bits.

Assume we start at Node A [2,1,3,0] and we want to route a request to Node B [3,2,1,4]. Since A is only aware of its neighbors, it routes the request to one of its neighbors along the direction to the destination. So in this case it routes to Node [3,1,3,0]. This process is repeated until the request reaches the destination as shown below. Each column represents a dimension and each row represents a node along the path from the source to the destination

Dimension	1	2	3	4
Source node A	2	1	3	0
CAN intermediate node	3	1	3	0
CAN intermediate node	3	2	3	0
CAN intermediate node	3	2	2	0
CAN intermediate node	3	2	1	0
Destination node B	3	2	1	4

Table 4.2. Intermediate CAN Node Routing

It is important to note here that there are other routes to the destination. These can be helpful when nodes go down.

4.2.2 CAN Distance

Recall that when comparing two bit strings, the Hamming distance is the count of bits different in the two patterns. In the same way we can define CAN distance as the minimum number of hops between 2 nodes.

4.2.3 Example

Consider the two patterns A and B, where A = 01101 and B = 11011, the hamming distance (A,B) = 3.

Now consider two nodes A and B, where A's coordinate is (2,1,3,0) and B's coordinate is (3,2,1,4), the CAN distance(A,B) = 5.

4.2.4 Average path length between 2 CAN Nodes

Theorem 4.1. *For a d-dimensional space partitioned into n equal zones, the average routing path length is* $d.(\frac{n^{\frac{1}{d}}}{4})$

Dimension	1	2	3	4
A	2	1	3	0
B	3	2	1	4
CAN Distance on dimension i	1	1	2	1

Table 4.3. CAN Distance (A,B) = 5

Proof: To calculate the average distance between two CAN nodes chosen at random, we first look at computing the average distance between two points chosen at random in the segment [0,1].

This value can be computed by the solving this double integral:

$$\int_0^1 \int_0^1 \mid y - x \mid \, dy \, dx$$

Since there is symmetry between the choices of the two points, we can re-write this integral as follows

$$= 2 * \int_0^1 \int_x^1 (y - x) \, dy \, dx$$

$$= 2 * \int_0^1 \left(\frac{x^2}{2} - x + \frac{1}{2} \right) \, dx$$

$$= 2 * \frac{1}{6} = \frac{1}{3}.$$

Now, let's extend this problem to calculating the average distance between two points chosen randomly in a torus i.e. the CAN case. This value can be computed by solving the following integral, where $\frac{\theta}{2\pi}$ represents the distance between the two points in the torus (the distance equals θr where $r = \frac{1}{2\pi}$ since the circumference $= 2\pi r = 1$)and we calculate this integral over half the torus (the two points can't be more than half the circumference of the torus apart):

$$\frac{1}{\pi} \int_0^\pi \frac{\theta}{2\pi} \, d\theta$$

$$= \frac{1}{\pi} \left(\frac{\pi^2}{4\pi} \right) = \frac{1}{4}.$$

Returning back to the d-dimensional space, and projecting the n nodes into the torus, we get that the average distance between two points is $\frac{k}{4}$ where k is the number of nodes on one side of the torus.

$$n = k^d$$

$$k = n^{\frac{1}{d}}$$

so the average distance between two nodes on any given dimension $= \frac{k}{4}$
Then the average path length between two nodes is
(Number of dimensions)(average distance between two nodes on any given dimension) $=(d)(\frac{n^{\frac{1}{d}}}{4})$

□

4.2.5 Observation

With n fixed and large, as d increases the term $dn^{\frac{1}{d}}$ decreases.
Lets see how the average path length changes as we double the dimension d

$$\frac{2d.n^{\frac{1}{2d}}}{d.n^{\frac{1}{d}}} = \frac{2.d.n^{\frac{1}{2d}}}{d.n^{\frac{1}{2d}}.n^{\frac{1}{2d}}} = \frac{2}{n^{\frac{1}{2d}}}.$$

So this means that doubling the dimension d will only be better if $n^{\frac{1}{2d}} \geq 2$
What is a good choice for d?
Maximizing d subject to the constraint above would yield $\mathrm{O}(\log n)$

$$The\ average\ path\ length = \frac{\log n(n^{\frac{1}{\log n}})}{4}$$

$$= \frac{\log n(2^{\log n})^{\frac{1}{\log n}}}{4}$$

$$= \log n(\frac{2}{4}) = \frac{\log n}{2}.$$

As stated above, increasing the number of dimensions d results in shorter path lengths $(d\frac{n^{\frac{1}{d}}}{4})$, but higher per-node neighbor state $(2d)$.

4.3 CAN Basics

CAN basic operations can be summarized as follows:

- **Bootstrapping**
  1. Find a CAN node
- **Enter the CAN**
  1. Choose a random point P ϵ U
  2. Search for P using the hash function
  3. Inform the node owning P to split. Its recommended to split from left to right across dimensions.
  4. Inform old node's neighbors with this change
- **Leave the CAN**
  1. In case of graceful departure, inform your neighbors with explicit handing over your zone
  2. In case of dropping out, (key,value)pairs must be refreshed periodically in the background and a distributed algorithm (TAKEOVER) will redivide the CAN zones.

4.4 CAN Design Improvements

- **Multi-dimensioned coordinate space**
  Increasing the number of dimensions d not just results in shorter path lengths but also, implies that a node has more neighbors, hence it can find more next hop nodes so routing can be more fault tolerant.

- **Realities: multiple coordinate spaces**
  We can maintain multiple independent coordinate spaces (realities) with each node. So a node would be responsible for different zones in different realities. Such increase in realities r would improve data availability since data would be stored on every reality. Also, Multiple realities improve routing fault tolerance, because in the case of a routing breakdown on one reality, messages can continue to be routed using the remaining realities.
  One important point to know here is that if one were willing to incur an increase in the average per-node neighbor state for improving routing, then the right way to do so would be to increase the dimensionality d rather than the number of realities r.

- **Better CAN routing metrics**
  In CAN routing, a node routes messages to one of its neighbors along the direction to the destination. If this node measures the network-level round-trip-time RTT to each of its neighbors and then selects the neighbor with the maximum ratio of progress to RTT, this would reduce the network latency.

- **Overloading coordinate zones**
  With zone overloading, we allow multiple nodes to share the same zone (nodes that share the same zone are called peers). So rather than splitting the space, they share the space. This sharing have several advantages. First, it reduces the path length because by placing multiple nodes per zone, we reduce the number of nodes in the system. Second, it improves the fault tolerance because a zone is down if all the peers are down. Third, when combining the RTT measurements with the choice of peers, we can reduce the network latency. Overloading zones adds to system complexity as nodes must track a set of peers.

- **Multiple hash functions**
  Hashing a key using k hash functions would map this key into k points in the space. This would improve data availability because the pair (key, value) that is associated with this key is unavailable only when all the k replicas are simultaneously unavailable.

- **Topologically-sensitive construction of the CAN overlay network**
  CAN construction mechanism allocates nodes to zones in random, which doesn't reflect anything about the underlying IP network. What we want is to make neighbors in the coordinate space are likely to be topologically close on the Internet. This can be achieved by the idea of land-marking where nodes measure their RTT to different landmarks and order this list and when they join the CAN, they only join in that portion of the coordinate space associated with its landmark ordering.

- **More Uniform Partitioning**
  In order to achieve more uniform partitioning, when a node joins the CAN, rather than splitting the zone with the zone's owner, the zone owner checks its neighbors and then the node with the largest volume would split its zone with the incoming join requester.

CAN provides a scalable indexing mechanism in its core, not only for peer-to-peer file distribution but also can be used to construct a wide-area name resolution service. With DNS, the naming scheme is coupled with the name resolution process (like www.cs.bu.edu would be resolved to bu.edu). With CAN, we can hash any name on any location to a point and find its IP from the node associated with this zone, thus decoupling the naming from the name resolution process.

Bibliography

[1] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. ”A Scalable Content-Addressable Network,” in Proceedings of ACM SIGCOMM ’01.