

A brief introduction to Content-Addressable Networks

Alireza Ghasemi

January 2008

1 INTRODUCTION

Content-Addressable Networks (CANs) are indeed distributed large-scale hash tables. The key goal of these structures is achieving a scalable indexing system for large-scale decentralized storage applications on the web. Basic operations on CANs are inserting a $(key, value)$ pair, deleting a $(key, value)$ pair, and looking up the value associated with a key. Hash table entries are divided across nodes and each node contains a portion of the entire hash table with respect to the rules we will see later.

2 DESIGN

2.1 Core idea

The main idea of CAN is having a d -dimensional space and assigning a portion of it to each node. We must also have a hash function which maps keys to a point in d -dimensional space. Then, by storing in each node those $(key, value)$ pairs whose keys are mapped to that node's portion of space, We can find the location of each hash table entry.

2.2 Routing in CAN

Routing in CAN is performed through the concept of neighbours. In d -dimensional space, Two nodes are called neighbours if their coordinate zones span overlap in $d-1$ dimensions and abut 1 dimension. Recall the familiar concept of neighbourhood in 2 and 3-dimensional space. In average, A node will have $2d$ neighbours and it must store the IP address and coordinate zone of these nodes.

To find the node whose zone contains a special point, In each hop, the nearest neighbour (or one of the nearest points) to the destination point is chosen until the destination is found, Using this algorithm, The average path length will be $\frac{d}{4}n^{\frac{1}{d}}$ hops with n being the number of nodes. So, the average routing time will grow as $\mathcal{O}(n^{1/d})$. It may seem worse than $\mathcal{O}(\lg n)$ of some other networks, But the advantage of CAN is that per-node data size ($2d$ neighbours' information) is independent of network size (n) which allows for high scalability of Content-Addressable Networks. On the other hand, however, we can achieve the same $\mathcal{O}(\lg n)$ routing time by choosing d as an appropriate function of n . But as said, It will limit scalability and so we prefer constant d .

2.3 CAN construction

Whenever a node wants to join the CAN, it must be given a portion of the entire CAN space and all $(key, value)$ pairs in it.

The process of joining a CAN takes three steps; First, The new node must find a node already in the CAN. How this is done may vary and is not important for us. e.g., We may have *bootstrap* nodes which always contain the IP address of some nodes which are surely in CAN and choose one of them whenever they are requested to give address of a CAN node.

Now, having the address of a node, our new node chooses a random point in the coordinate space of CAN and requests the very existing node of CAN to find the node whose coordinate zone contains that point. Then the new node tells the found node that it wants to have half of its coordinate zone. The found node then splits its coordinate zone in two portions and associates one of them to new node (according to a predefined convention) . Also the $(key, value)$ pairs belonging to the portion allocated to the new node are transferred to it. New node gets the information of its neighbours from the old occupant nodes and the occupant node also alters its neighbours list and erases those nodes that are no longer its neighbours.

At last phase of joining process, neighbours of the newly joined node and the split must be informed of the new node so that it can join the routing mechanism. This is done by sending an update message to all neighbours and giving them new information. The joining mechanism only affects $\mathcal{O}(d)$ nodes which is independent of network size. This fact becomes important when the number of nodes grows rapidly and becomes extremely large, Another advantage of constant d .

2.4 Node departure in CAN

When a node want to leave the network it must deliver its associated coordinate zone as well as $(key, value)$ pairs to one of its neighbours. If any neighbour exists such that its zone can be properly merged with zone of the leaving node and construct a new coordinate zone, then that node is chosen. If no such node exists the node with smallest zone is selected and it will temporarily handle both its zone and the leaving node's zone.

CANs must also be robust to network failures, i.e. when one node suddenly vanishes. In this case one of the dead node's neighbours must takeover its zone. But, however, the $(key, value)$ pairs stored in that node will be lost until the node becomes alive again. To prevent this loss, nodes inserting $(key, value)$ pairs in CAN may periodically refresh these entries.

To achieve robustness against network failures, Nodes should periodically to their neighbours, giving them their zone coordinates along with the list of their neighbours and their coordinate zones. Prolonged absence of update messages from a node signals its failure. When a node detects that one of its neighbours is dead, It starts a TAKEOVER. The length of this timer depends on the volume of the zone associated with the node. This mechanism allows efficiently selection of node to takeover the unhandled zone.

3 DESIGN IMPROVEMENTS

CAN's performance may be improved in various ways. We list some of them here:

- *Multi – dimensional spaces*: The more the dimension of the coordinate space, The less the average path length (Recall the $\mathcal{O}(dn^{1/d})$ formula.).
- *Better routing metrics*: We may take RTT as well as closeness to destination into account when choosing a neighbour to progress and so reducing the routing time.
- *Overloading the coordinate zone*: we may allow more than one node to share the same zone. This may increase availability and reduce path length in cost of redundancy.
- *Topologically – sensitiveness*: Nodes which are geographically close to each other may be assigned nearby portions of the coordinate space. RTT can be a good measure for geographic closeness. This policy reduces per-hop latency.
- *More uniform partitioning*: Nodes with large volume may be selected to be split when a new node joins the network. This balances the volume of nodes.
- ...

The reference contains more detailed improvements.

4 PROBLEMS AND RESEARCH AREAS

The design of CAN addresses two key problems: scalable indexing and routing. But the security problem still exists in CANs. Also denial-of-service attacks are hard to encounter because a malicious node can act as clients and server as well as taking part in routing mechanism.

Also researches may be done to handle mutable contents. Search algorithms can also be changed to allow keyword search in Content-Addressable networks.

5 REFERENCE(S)

Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A Scalable Content-Addressable Network. In *Proceedings of ACM SIGCOMM* 2001.