CS559: ALGORITHMIC ASPECTS OF COMPUTER NETWORKSFall 2007

Lecture 15 — October 31

Lecturer: John Byers *BOSTON UNIVERSITY* *Scribe: Georgios Smaragdakis*

In today's lecture, we elaborate more on structured peer-to-peer systems based on Distributed Hashing Tables (DHTs). We present two of them, (1) the Chord system [3], and (2) the Content-Addressable Network (CAN) [2]. Both systems are motivated by very large storage applications, where a large universe of items has to be placed and retrieved in a large network.

15.1 DHT Basics

Main functions of DHTs are:

- Lookup; given a *key*, the key is mapped onto a node, resulting a $< key, value >$ pair. The metric that is used to evaluate the performance of lookup is the number of messages to perform a lookup. DHTs eliminate the need for bootstrapping.
- Maintenance, *i.e.* repairs and updates.
  - Correctness, *i.e.* maintain the invariance of the system under churn, is a requirement.
  - The performance metric is the number of messages needed to perform a restoration due to the arrival of a new node in the system, departure or failure of a node (the restoration complexity is identical).

15.2 Chord

In Chord, both the nodes and the items are hashed on $[0, R]$, where R is the maximum value in the hashing space (the hash space in not any more a line, but a ring). Consider a system with N nodes and I items or keys (henceforth, we will use the terms items and keys interchangeably). A node in position p stores its successors (the finger table): $\bigcup_i succ\left(p + \frac{R}{2^i} \ \ mod\ R\right)$, $\forall i \in [1, 2, ..., \log R]$ (see Figure 15.1).

For routing purposes, a node needs to store its successor and the intervals that each successor is responsible for. Thus, the space requirements for the routing table is $\approx \log R$ pointers and intervals. Each node is also connected to the immediate neighbor (to guarantee that a request always progresses). Note that the aforementioned system achieves load balancing of storage, if hashing is also used to place objects. On average, I/N items are stored in each node. It is also proved that the distance to an object is halved after each hop, with high probability.

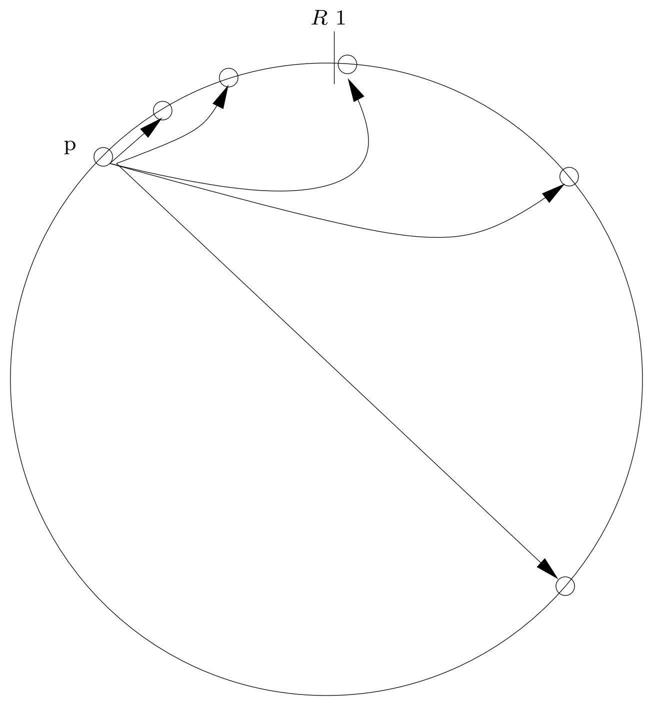

Figure 15.1. Graphical illustration of the overlay neighbors that node with id p maintains.

example: $p = 50,\ R = 100$
Routing table:

3	[3 49]
78	[78 2]
63	[63 77]
...	

15.2.1 An alternative view of the Chord structure

- Diagrammatically looks different than random trees [1].
- Nevertheless, Chord structure is similar to an unbalanced k-ary tree, like random tree (see Figure 15.2). Ideas like threshold-replication can be also applied in Chord. Notice that different paths traverse the same links before reaching a target node.

15.2.2 Node insertions and departures

Node Insertion
Initially, a newcomer node should have an entry point, *i.e.* connect to at least one node that already participates in the network. Then it proceeds to the following steps:

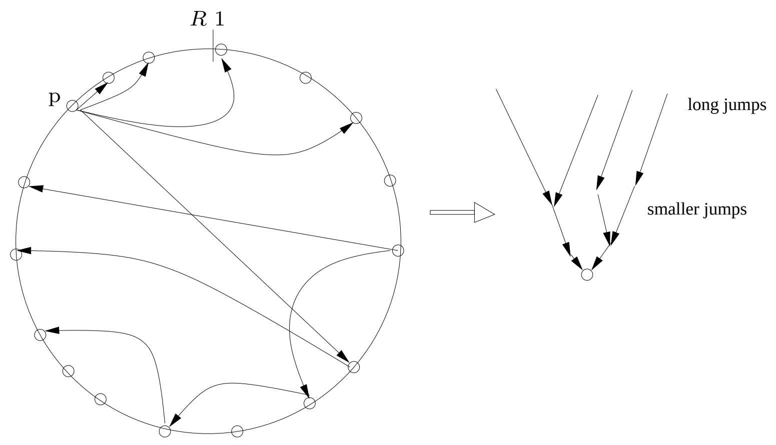

Figure 15.2. Reduction of Chord to Random Tree structure.

	Chord	CAN
Lookup	$O(\log N)$	$O(d \cdot N^{1/d})$ for $d = \log N,\ O(\log N)$
Restoration	$O(\log^2 N)$ ($\log N$ pointers have to be adjusted and each one is located in $\log N$ steps)	$O(d \cdot N^{1/d})$ for $d = \log N,\ O(\log N)$

Table 15.1. Complexity (probabilistic bounds) of lookup and restoration in Chord and CAN. N is the number of nodes in the system and d is the dimensionality of space in CAN.

1. Find the successor and predecessor, in $O(\log N)$ steps. The newcomer lookups the hash value for key and node id that are mapped to the same space, and selects the responsible node id as successor (or predecessor accordingly). The selection is strictly based on the has value and not any network or geographical distance metric.

2. Migrate $< key, value >$ pairs.

3. Update finger table (entries that point to the interval $[x, p]$). There are $\log N$ intervals each of size $dist(x, p)$ that have to be corrected. Locating the first node in each of these intervals amounts a lookup that takes $O(\log N)$ time. Interval width is expected to be a small (see also Figure 15.3).

Note that each node has a constant out-degree and in-degree.

Node Departure

When a node departs (or fails), the aforementioned steps have to be undone. Note that now the successor and the predecessor of the departing node are known.

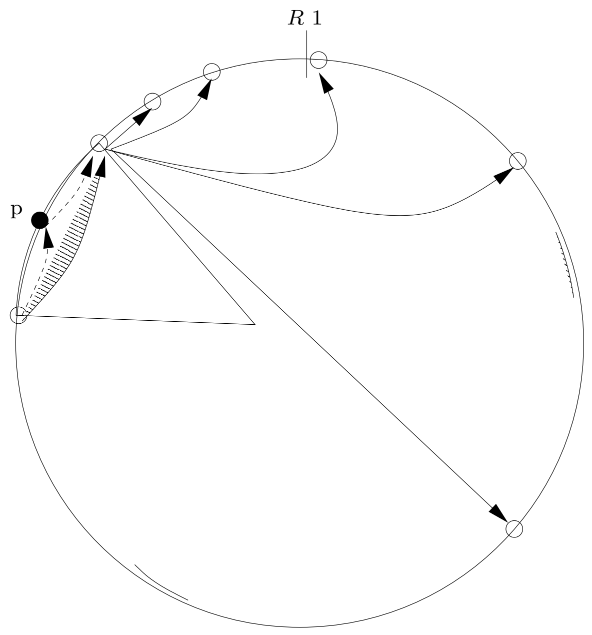

Figure 15.3. Node p is a newcomer in the system. Shaded area denotes that hash area that has to be reassigned.

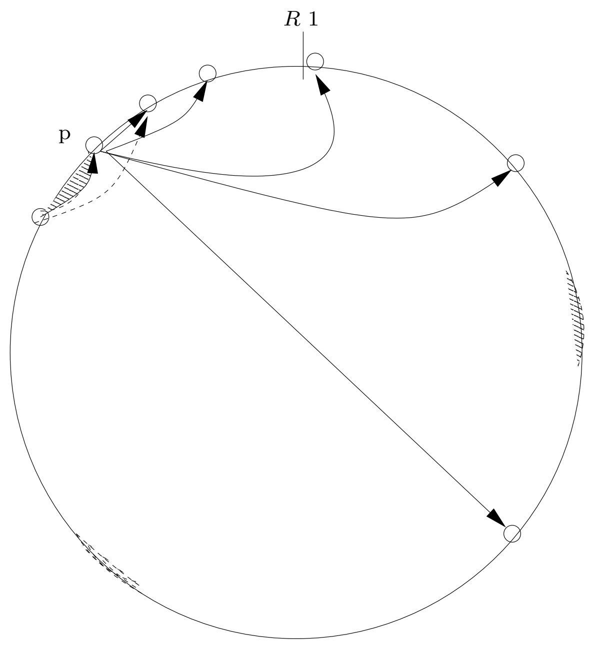

Figure 15.4. Node p is leaving the system. Shaded area denotes that hash area that has to be reassigned.

15.3 CAN

In CAN, items are placed in a d-dimensional lattice (coordinate system). Servers are responsible for zones (regions) in the d-dimensional space. Requests are routed to neighboring regions, resulting a worse routing time of $d \cdot N^{1/d}$ (product of dimensions and side length of dimension, see Figure 15.6). Insertion of a server causes a zone to be split, along with migration of $< key, value >$ pairs, and update of neighboring servers in the coordinate system. Departures of servers causes zones to be merged (along with migration of pairs and updates of neighboring servers).

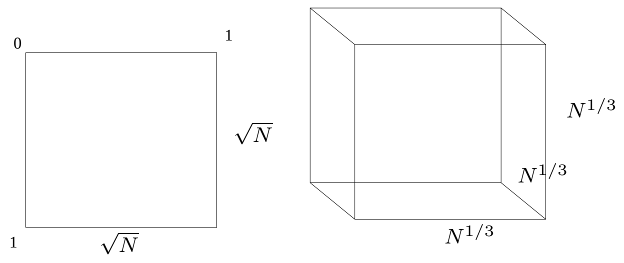

Figure 15.5. CAN defines a d-dimensional space. It can be abstracted as a d-grid with $N^{1/d}$ points in each dimension.

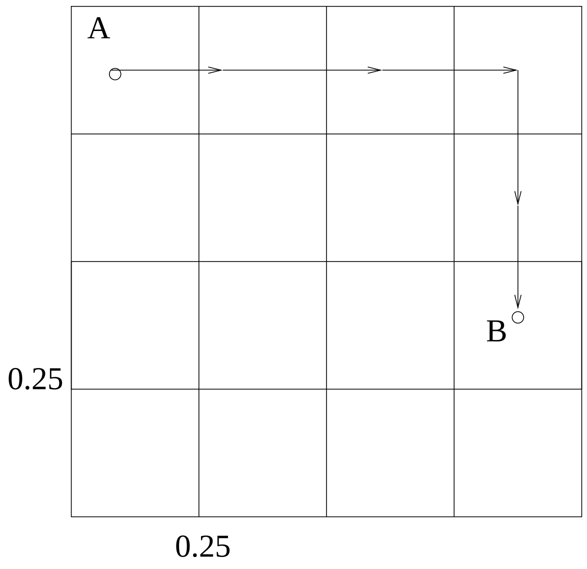

Figure 15.6. Forwarding in (2-dimensional) CAN.

Bibliography

[1] David Karger, Eric Lehman, Tom Leighton, Rina Panigrahy, Matthew Levine, and Daniel Lewin. Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web. In *STOC '97*, 1997.

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01*, 2001.

[3] Ion Stoica, Robert Morris, David Liben-Nowell, David R. Karger, M. Frans Kaashoek, Frank Dabek, and Hari Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Trans. Netw.*, 11(1):17–32, 2003.