Lecture 21

1 Last time: Chord [3]

1.1 Properties of Chord

- insert/lookup of ojbect in $\Theta(\log^2 n)$
- join/leave of peers or "buckets" in $\Theta(\log^2 n)$
- storage is $\Theta(\log n)$ per peer + storage required for ring and fingers
- load-balanced since each bucket handles the same amount of the object space

1.2 Deterministic lookup

Chord features deterministic lookup, that is if an object is present, it is always found.

- non-trivial because of simultaneous changing of pointers, joining leaving of peers
- made easy by ring topology
- with finger list, time per operation can be reduced to $\Theta(\log n)$, where time corresponds to the number of hops taken

2 Awerbuch-Peleg [1], Thorup-Zwick [4]

Big idea: *static* scheme to locate the closest object.

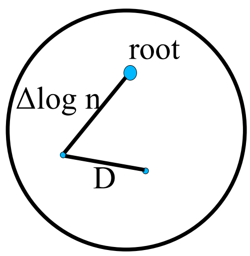

Figure 1: Distance to closest object under A.P.

3 CAN: Content Addressable Network [2]

Big idea: addresses ≡ points in d-dimensional space

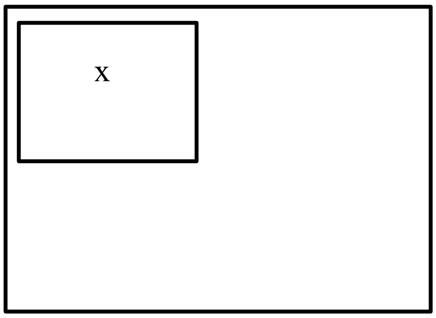

Figure 2: Object space

Adding a new node works as follows. When a new node joins, we split the largest box. The objects in the new node's object space come from a single other guy. Deleting a node requires merging of two smallest boxes (if the right node is deleted), but it can get more subtle and require rebalancing of the object space between nodes. We can use data structure tricks to amortize the tree.

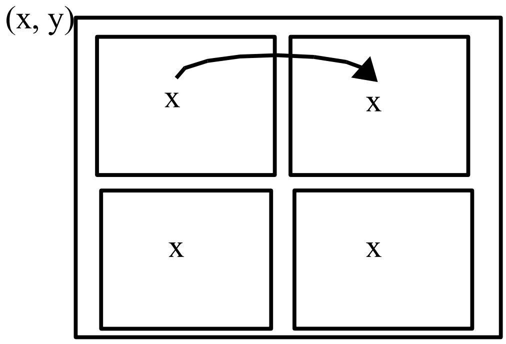

Figure 3: "Fixing" of dimensions in an object space

Routing time: $O(dn^{1/d})$ "fix one dimension at a time"
($d = \log n \Rightarrow n^{1/\log n} \log n \sim \Theta(\log n)$)
Space: $O(2d)$

4 Tapestry [6]/Pastry/PRR

Big idea: each item/node gets a random ID in base B

Example 1 Base 5

1234		123		12		1			
1234	addr	1230	addr	120	addr	10	addr	0	addr
		1231	addr	121	addr	11	addr	1	addr
		1232	addr	122	addr	12	addr	2	addr
		1233	addr	123	addr	13	addr	3	addr
		1234	addr	124	addr	14	addr	4	addr

Table 1: Routing table for base 5 Tapestry node

The table is constructed by pinging all the nodes with the same prefix and finding the address of the closest one. Each object is assigned to node with longest matching prefix.

EXAMPLE 2

$$4321 \mapsto \left|\begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array}\right| \; 4??? \mapsto \left|\begin{array}{c} 40 \\ 41 \\ 42 \\ 43 \\ 44 \end{array}\right| \; 43?? \mapsto \left|\begin{array}{c} 430 \\ 431 \\ 432 \\ 433 \\ 444 \end{array}\right| \ldots$$

Space: $O(b \log_b n)$
Features:

- distributed prefix tree (every ID is a root of a tree)
- complicated to maintain
- when you route, you may miss the root

In contrast to Chord where neighbors are half way across the network, Tapestry neighbors are close. The T.Z. algorithm cannot be implemented with Tapestry because nodes do not know about their grandparent(s).

5 Growth-restricted metrics

$|B(x, 2r) \leq c|B(x, r)|$

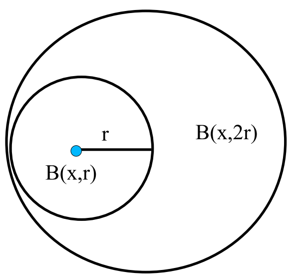

Figure 4: Growth-restricted metric

6 Domain Name Service [5]

Domain Name Service (DNS) is the distributed system for resolving domain names on the internet. It is hierarchically organized with higher-level servers answering queries for a particular suffix of the address and lower-level servers answering queries for a more qualified suffix of the address and so on until the full address is resolved. There is a single tree of the DNS servers, and everyone knows the root nodes.

7 Summary

Scheme	Big Idea	Publish	Stretch	Hops	Storage	Bal?	Dyn?
A.P.	sparse covers	$\log^2 n \log \delta$	$\log n$	$\log \sigma$	$\log^2 n$	?	no
T.Z.	sampling	$\log^2 n$	$\log^2 n$	$\log n$	$\log n$	no	no
Chord	consistent hashing	$\log n$	-	$\log n$	$\log n$	yes	yes
CAN	grid	$dn^{1/d}$	-	$dn^{1/d}$	d	yes	yes
Tapestry et.al.	sampling/prefix trees	$\log_b n$	$O(1)$*	$\log n$	$b \log_b n$	yes	yes
GHT	411	$O(1)$	-	1	1	no	yes?
Everyone	phone book	$O(n)$	1	1	$O(n)$	yes	yes

Table 2: Summary

7.1 *Algorithm sketch

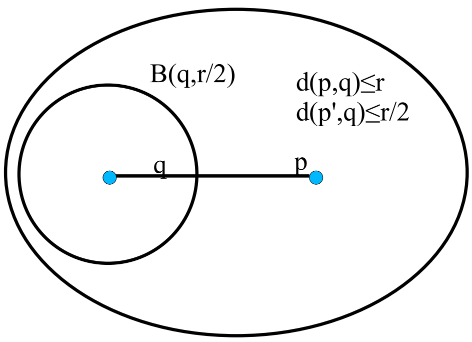

Figure 5: Growth-restricted metric

Sample p's ball.
$|B(p, 2r)| \leq c^3 |B(q, r/2)|$
Take a sample of c^3 $\rightarrow$ expect one point in q's ball
$\rightarrow$ miss the point with probability $(1 - \frac{1}{c^3})^{c^3} \leq \frac{1}{e}$
Take enough samples and use ping to find out the closest point. To find the closest neighbor $\Theta(\log \delta + \log n)$ steps are required.

References

[1] Baruch Awerbuch and David Peleg. Concurrent online tracking of mobile users. In *SIGCOMM*, pages 221–233, 1991.

[2] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[3] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pages 149–160, 2001.

[4] Mikkel Thorup and Uri Zwick. Approximate distance oracles. In *ACM Symposium on Theory of Computing*, pages 183–192, 2001.

[5] Mockapetris P. V. Rfc 1035: Domain names — implementation and specification, 1987.

[6] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, UC Berkeley, April 2001.