CS 244b - Coral

Partitioning & scalability

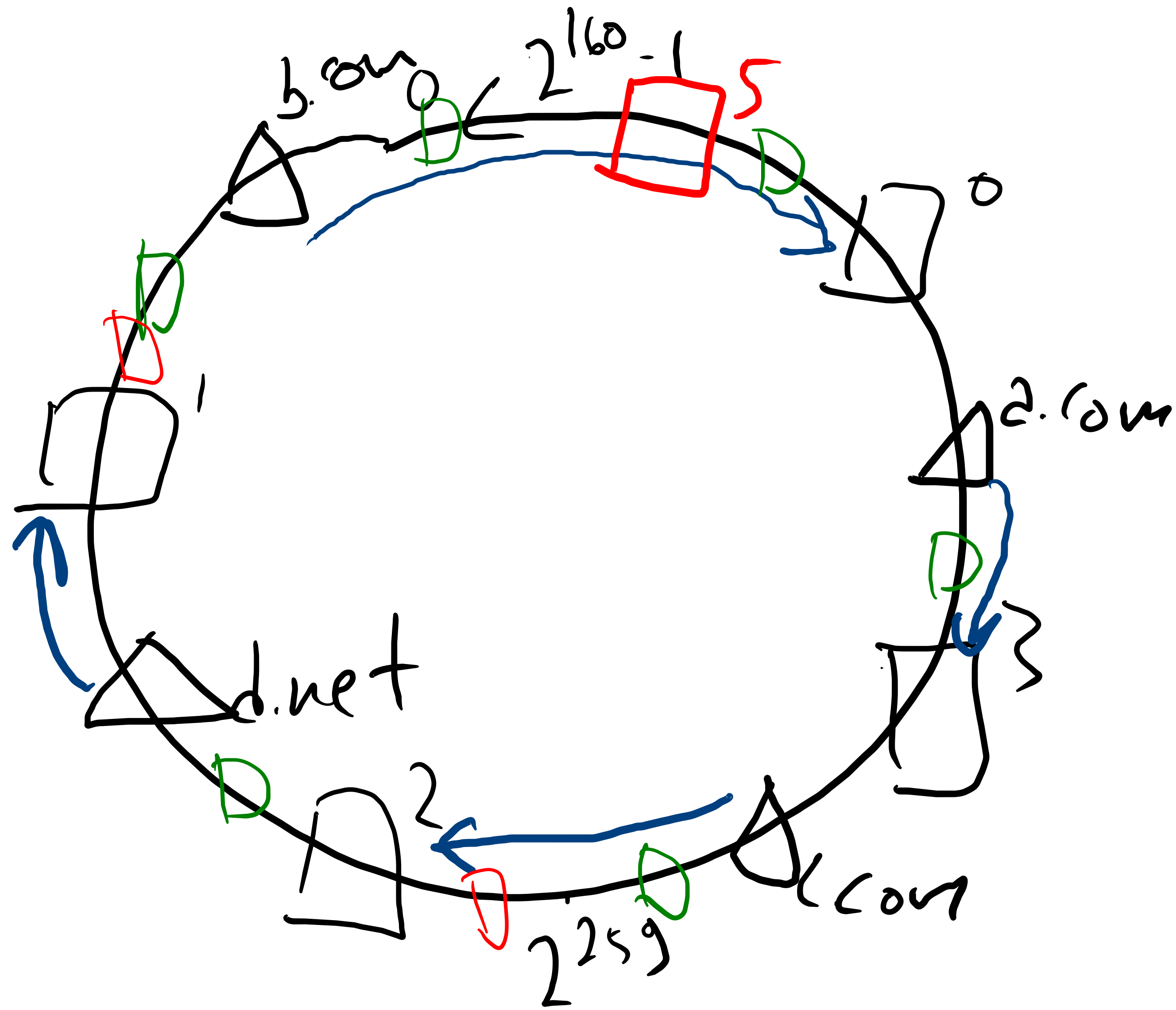
Consistent Hashing

P2P Systems

Non-linearizable storage systems

H(domain) ~~% 4~~ % 5

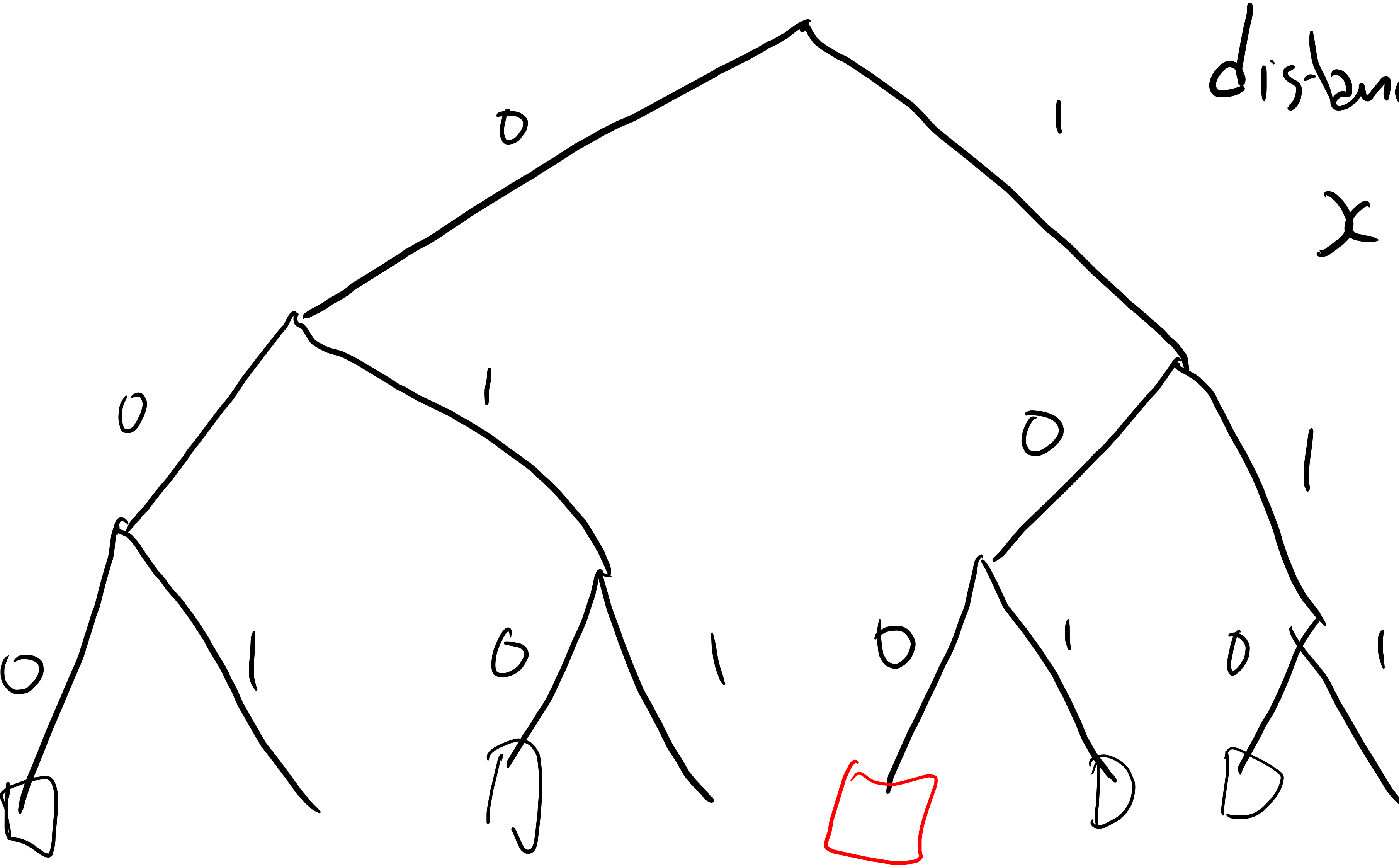
H(dom) → 160-bit   SHA-1



0.0
0.1
0.2
1.0
1.1
1.2
2
3

distance(x, y)

x XOR y



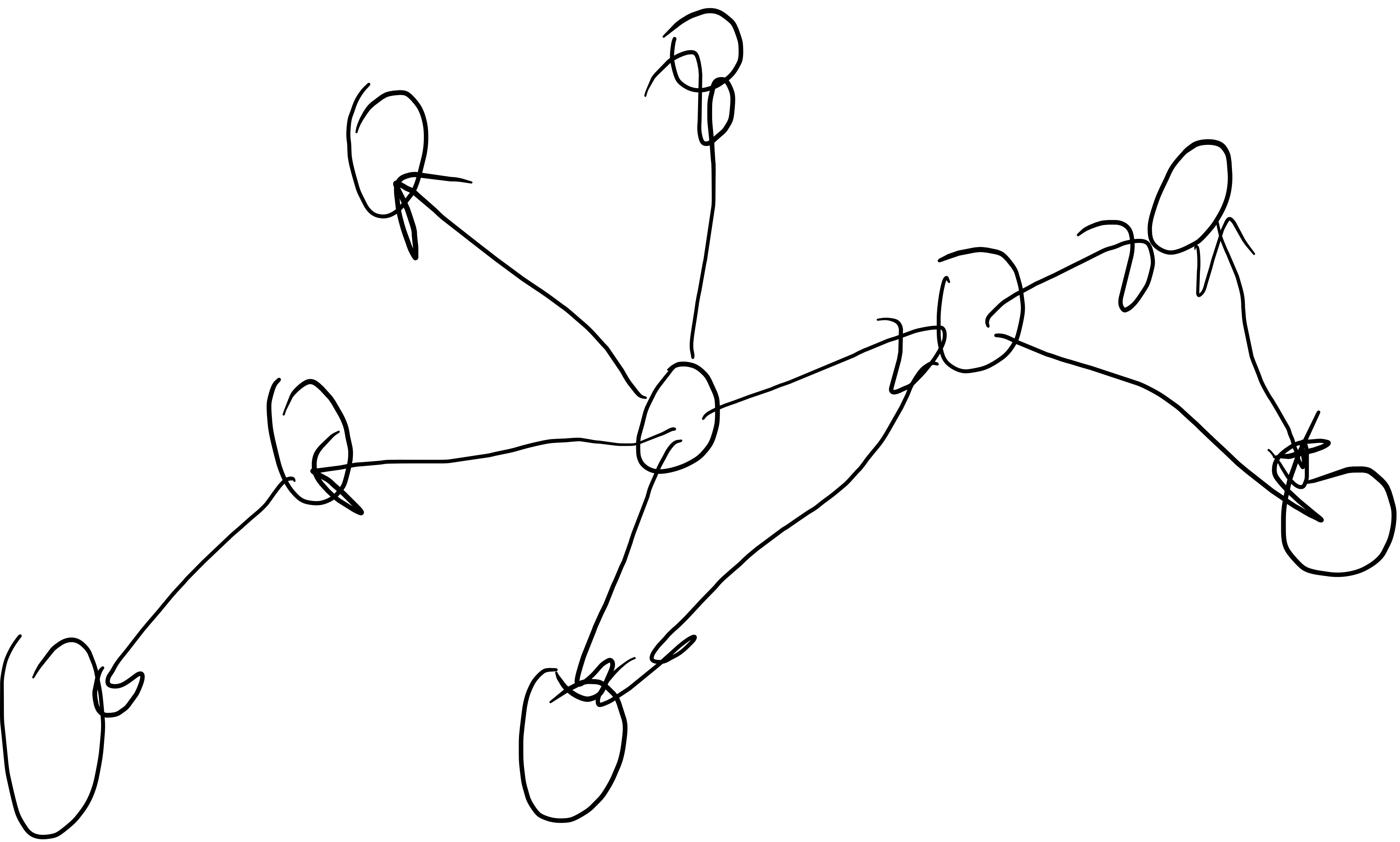
$H(\text{server Id}_i, \text{URL}) \rightarrow h_i$

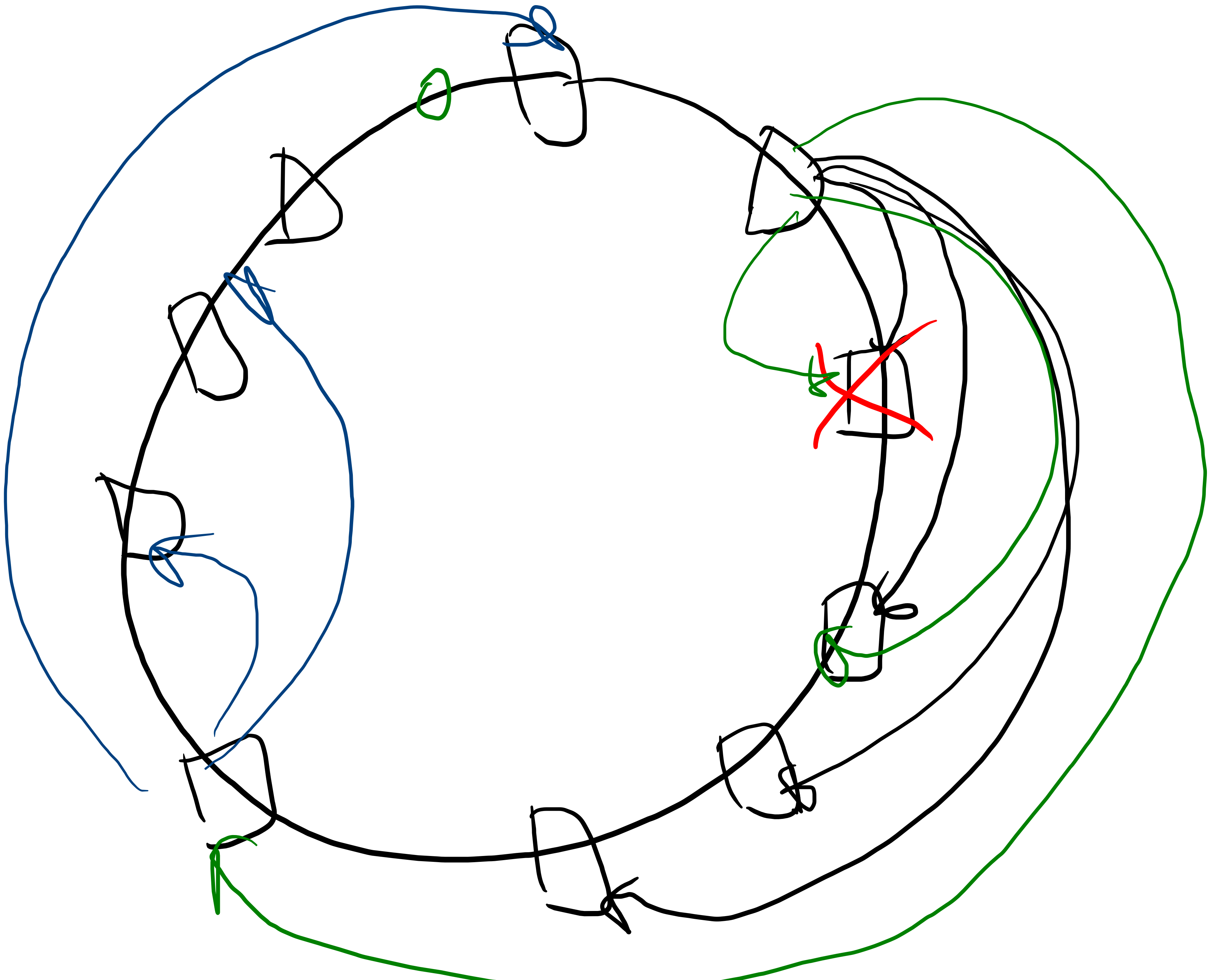
h_i / capacity

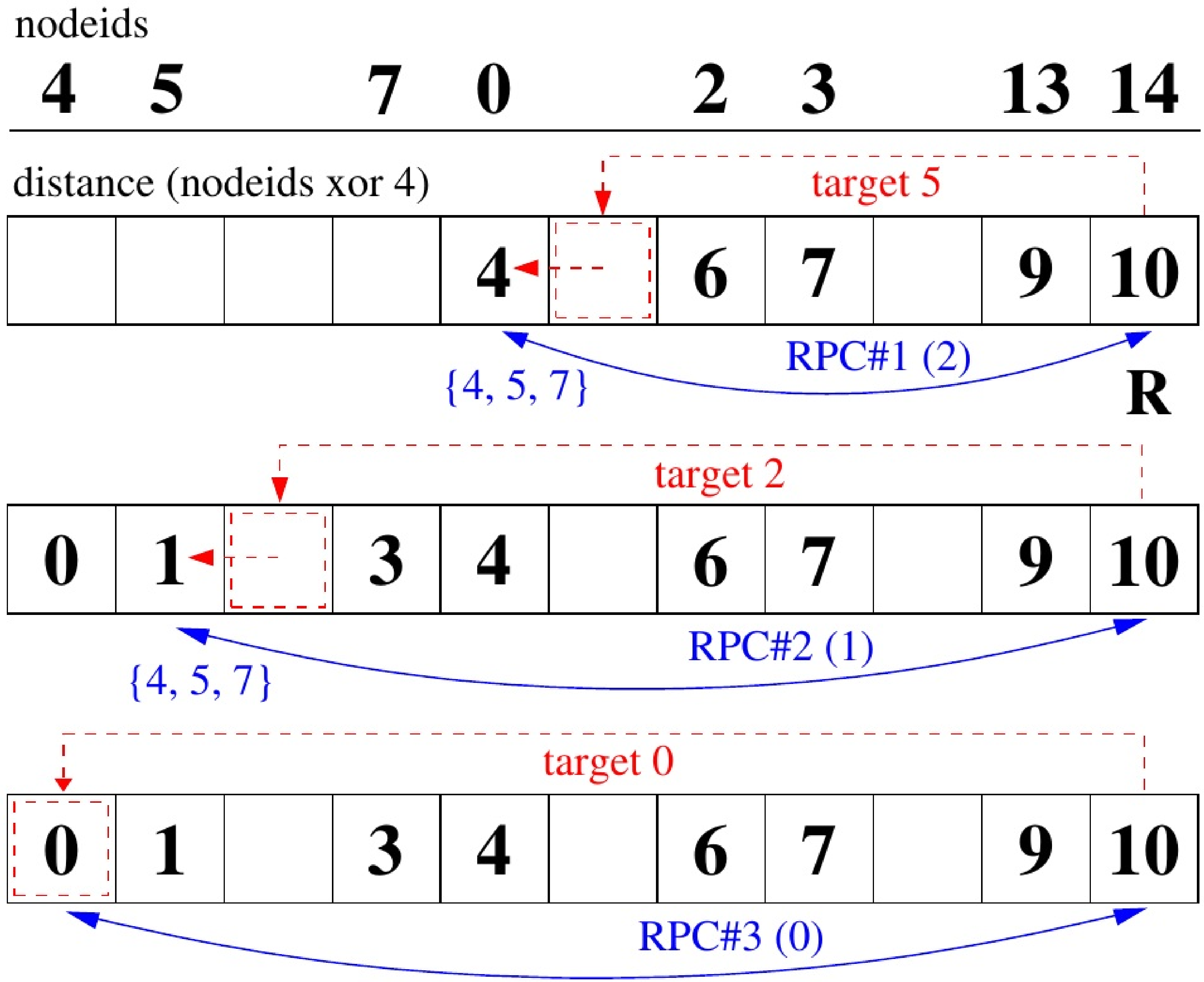
h_2

CARP

Gnutella






nodeids
4 5 7 0 2 3 13 14
distance (nodeids xor 4)
target 5
4
6
7
9
10
RPC#1 (2)
{4, 5, 7}
R
target 2
0
1
3
4
6
7
9
10
RPC#2 (1)
{4, 5, 7}
target 0
0
1
3
4
6
7
9
10
RPC#3 (0)

$\underbrace{b_0\, b_1}\, \underbrace{b_2}\, \cdots$

$2x, \quad \underbrace{2x+1}$

KOORDE

2004

No EC2 .nyud.net:8090

P2P DHT

Strawman1: K = H(URL) V = web page
use Kademlia to stor (k, v)

Strawman2: network of web proxies
use kad. for K = H(URL), V = proxy addr.

- Query far away nodes to find nearby data
  - Join multiple DHTs - diameter
  - Stop put at full + loaded node
- Download from nearby nodes

- Map clients to nearby coral proxies
  Probe DNS resolver

DNAME *.nyud.net →

*.L2.L1.L0.nyucd.net

OASIS

- $put(key, val, ttl, [levels])$: Inserts a mapping from the key to some arbitrary value, specifying the time-to-live of the reference. The caller may optionally specify a subset of the cluster hierarchy to restrict the operation to certain levels.

- $get(key, [levels])$: Retrieves some subset of the values stored under a key. Again, one can optionally specify a subset of the cluster hierarchy.

- $nodes(level, count, [target], [services])$: Returns *count* neighbors belonging to the node's cluster as specified by $level$. *target*, if supplied, specifies the IP address of a machine to which the returned nodes would ideally be near. Coral can probe *target* and exploit network topology hints stored in the DSHT to satisfy the request. If *services* is specified, Coral will only return nodes running the particular service, *e.g.*, an HTTP proxy or DNS server.

- $levels()$: Returns the number of levels in Coral's hierarchy and their corresponding RTT thresholds.

- Avoid hotspots in Coral

  Don't take shortcuts in routing

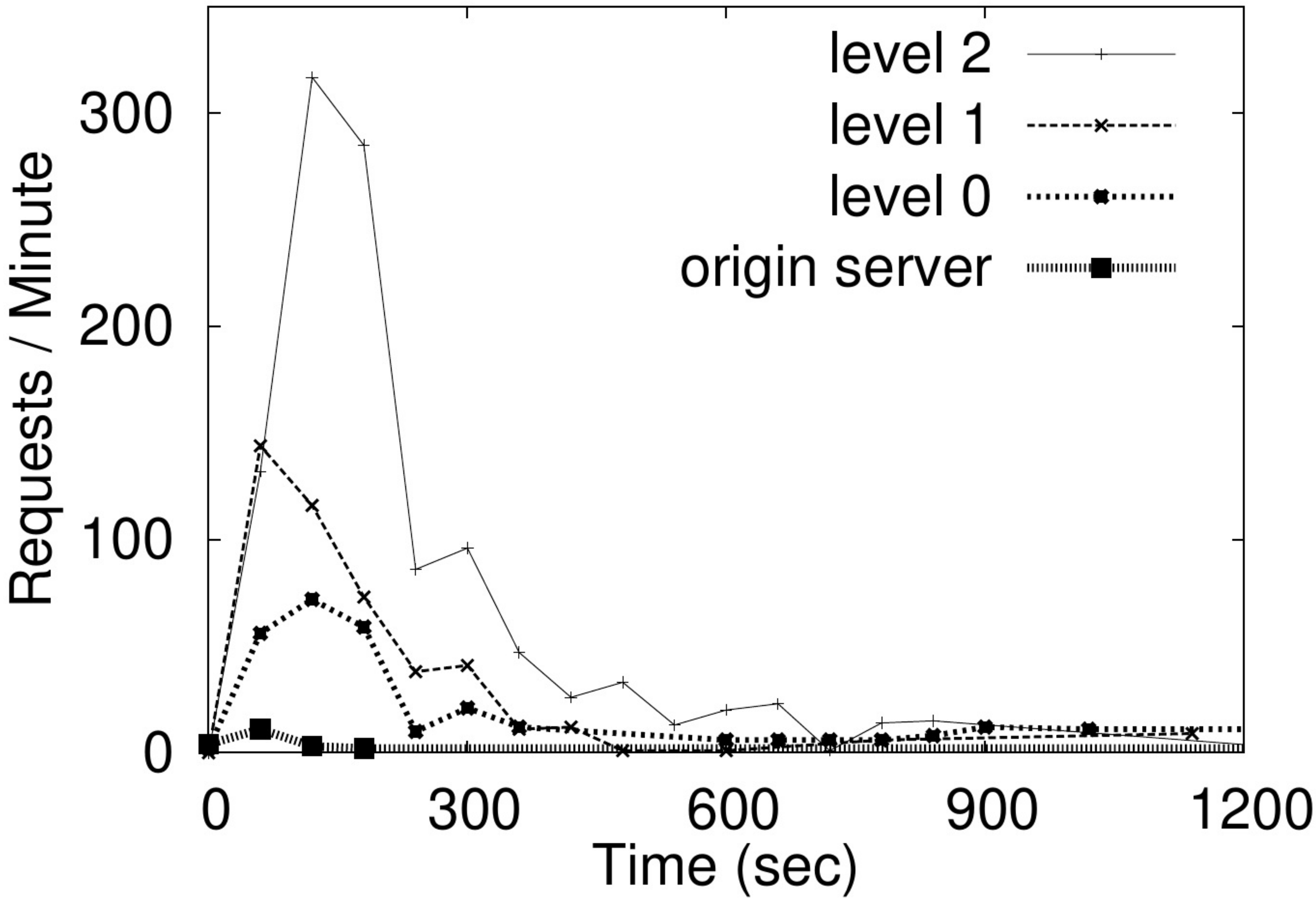


Figure 4: The number of client accesses to *CoralProxies* and the origin HTTP server. *CoralProxy* accesses are reported relative to the cluster level from which data was fetched, and do not include requests handled through local caches.

Request latency (sec)	**All nodes**		**Asian nodes**	
	50%	96%	50%	96%
single-level	0.79	9.54	2.52	8.01
multi-level	0.31	4.17	0.04	4.16
multi-level, traceroute	0.19	2.50	0.03	1.75

Figure 5: End-to-End client latency for requests for Coralized URLs, comparing the effect of single-level vs. multi-level clusters and of using traceroute during DNS redirection. The top graph includes all nodes; the bottom only nodes in Asia.

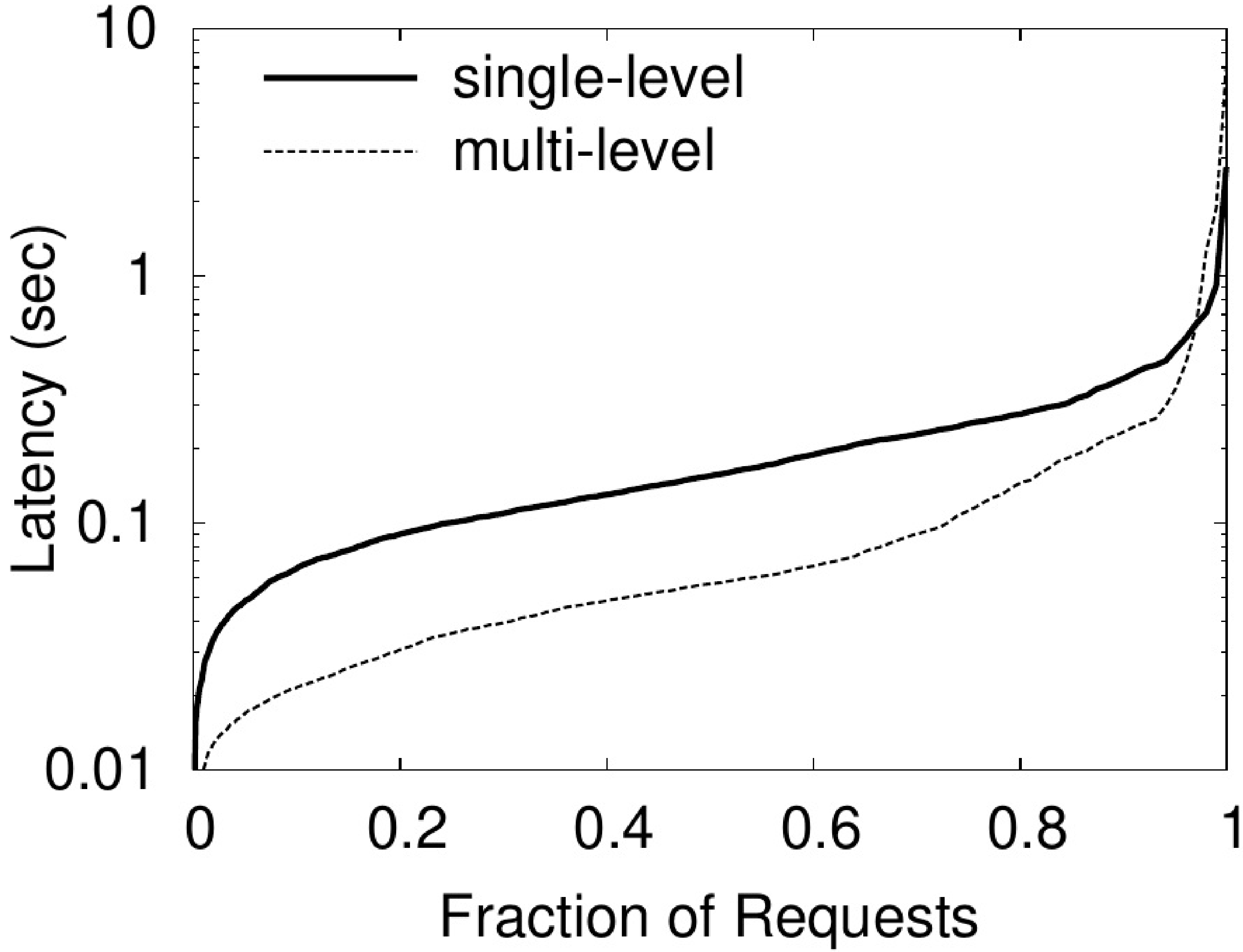
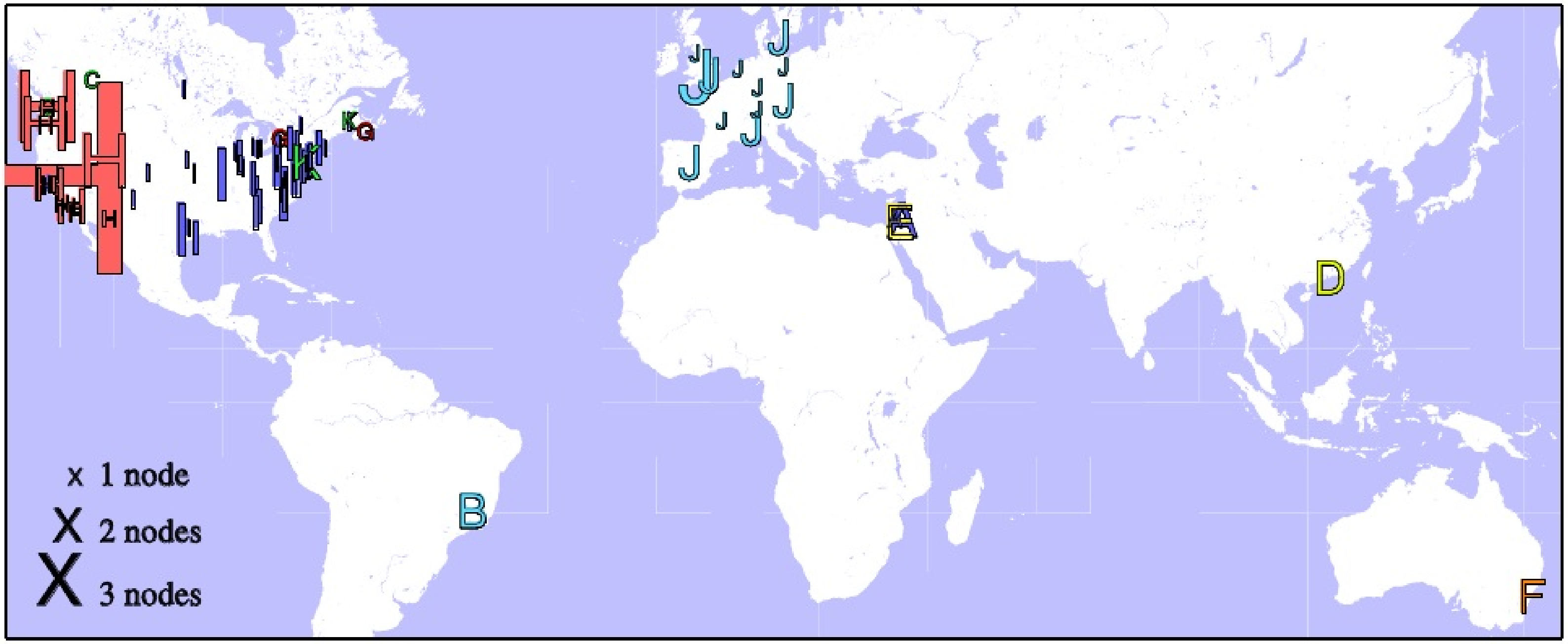
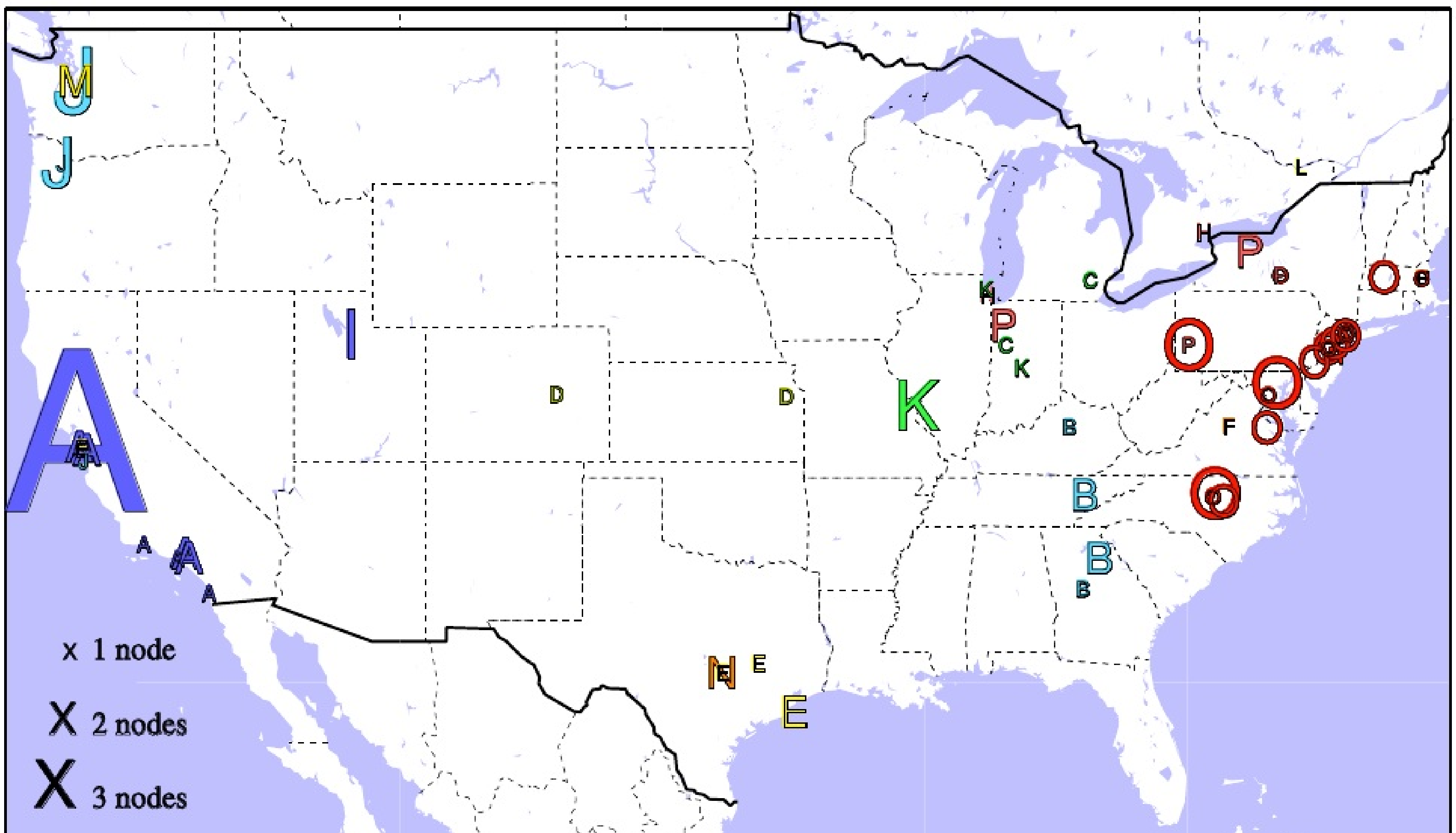


Figure 6: Latencies for proxy to *get* keys from Coral.


x 1 node
X 2 nodes
X 3 nodes


x 1 node
X 2 nodes
X 3 nodes

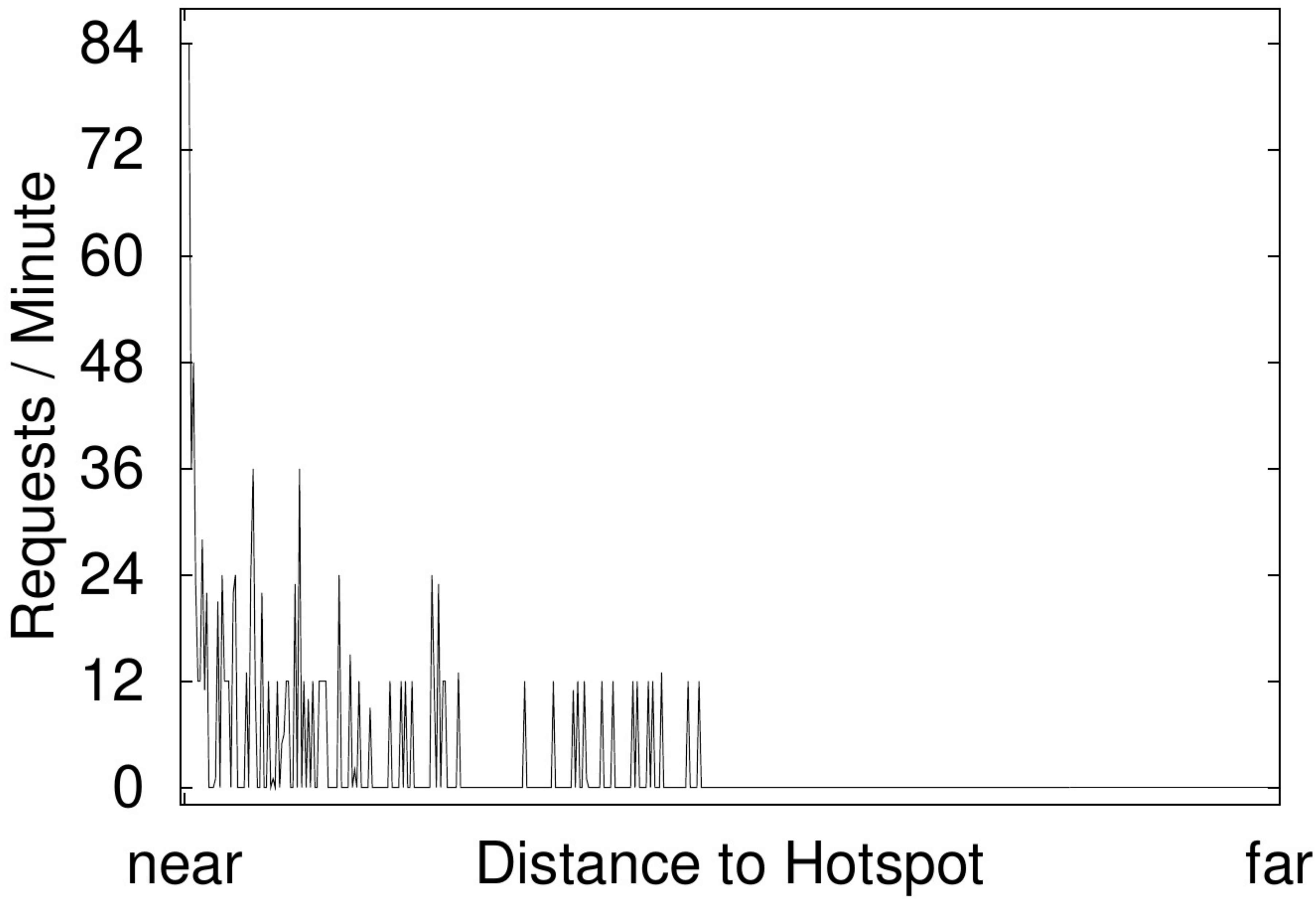


Figure 8: The total number of *put* RPCs hitting each Coral node per minute, sorted by distance from node ID to target key.