DHT Design Goals

- An "overlay" network with:
  - » flexible mapping of keys to physical nodes
  - » small network diameter
  - » small degree
  - » local routing decisions
- A "storage" or "memory" mechanism with
  - » best-effort persistence (soft state)
- We'll look at two designs:
  - » Chord
  - » Pastry

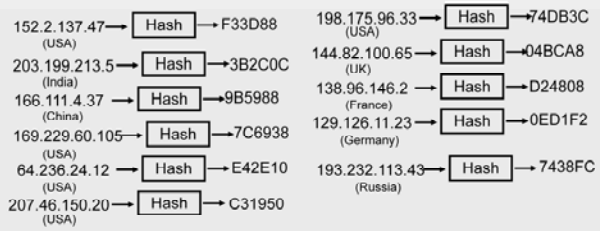
23

Pastry

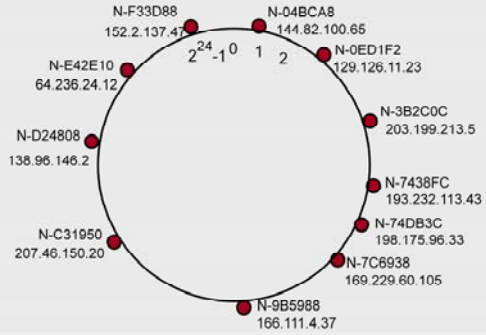
- Pastry *nodeIDs* and search keys are generated by a hash function that produces a value treated as:
  - » A sequence of digits with base 2^b (typically, b=4, *i.e.*, hexadecimal), and
  - » Modulo 2^{128}
- Given a message, ***m***, and a key, ***k***, Pastry routes the message to the *live* node with *nodeID* *numerically closest* to ***k***.

24

Hash IP address to Node ID (N) (example with b=4, modulo 2^{24})



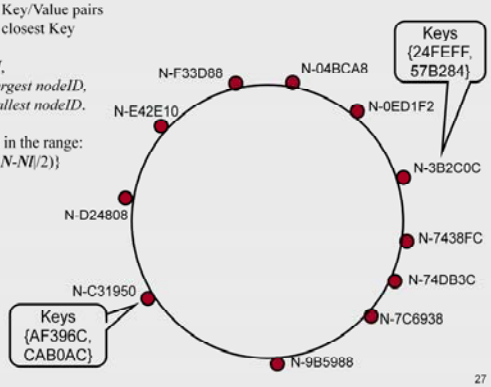
Nodes in Pastry Example (b=4, modulo 2^{24})



Key Locations in Pastry Example

Nodes should store Key/Value pairs for the numerically closest Key values.
For node with ID ***N***,
let ***Nl*** be the *next largest nodeID*,
and ***Ns*** the *next smallest nodeID*.

Node ***N*** stores keys in the range:
{***N***-(|***N***-***Ns***|/2), ***N***+(|***N***-***Nl***|/2)}



Pastry Routing State ("Prefix Routing")

Routing Table:

$\lceil \log_{2^b} N \rceil$ Rows, 2^b-1 entries (with one null) per row

The 2^b-1 entries at row ***n*** refer to remote nodes that:
- share the local node's *nodeID* in the first ***n*** digits, but
- whose ***n+1*** digit has one of the 2^b-1 possible values other than the ***n+1*** digit in the local *nodeID*.

If no such node is known, the entry is empty.

Leaf Set:

The leaf set, ***L***, is the set of nodes with the
- |***L***|/2 numerically closest **larger** *nodeIDs*,
- |***L***|/2 numerically closest **smaller** *nodeIDs*.

The typical value of |***L***| is 2^b

Nodeld 10233102

Leaf set ←	SMALLER	LARGER →	
	10233033	10233021	10233120	10233122
	10233001	10233000	10233230	10233232

Routing table

Row				
0	-0-2212102	1	-2-2301203	-3-1203203
1	0	1-1-301233	1-2-230203	1-3-021022
2	10-0-31203	10-1-32102	2	10-3-23302
3	102-0-0230	102-1-1302	102-2-2302	3
4	1023-0-322	1023-1-000	1023-2-121	3
5	10233-0-01	1	10233-2-32	
6	0		102331-2-0	
7			2	

Example with ***b***=2, |***L***|=8 (IP addresses not shown)

Pastry Routing/Lookup ("Prefix Routing")

- When a message for key *K* arrives at a node:
  - » If *K* is in the range covered by the Leaf Set, it is forwarded to the entry whose *nodeID* is numerically closest.
  - » Otherwise, the Routing Table is used to forward it to a *nodeID* that shares a ***common prefix*** with *K* by at least one more digit than does this node's *nodeID*.
  - » Otherwise, forward to a *nodeID* that shares a prefix with *K* at least as long as this node's *nodeID* but is numerically closer to K than this node's *nodeID*.

Pastry Routing Examples

Arriving Key, *K*

21032113 → 22301203

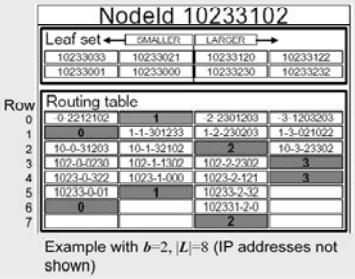
10311230 → 10323302

10233003 → 10233001

10233321 → 10233232

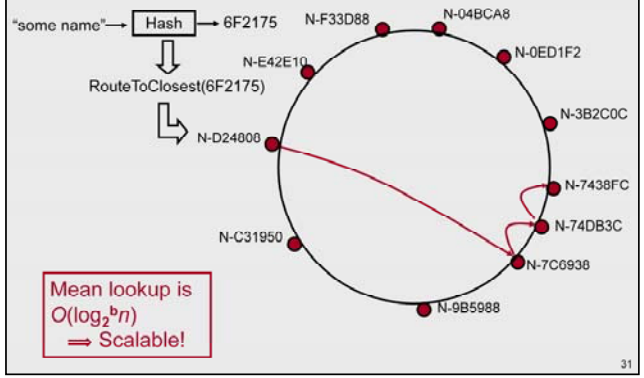
10223112 → 10222302

10233103 → 10233102



Example with b=2, $|L|$=8 (IP addresses not shown)

Pastry Routing Example

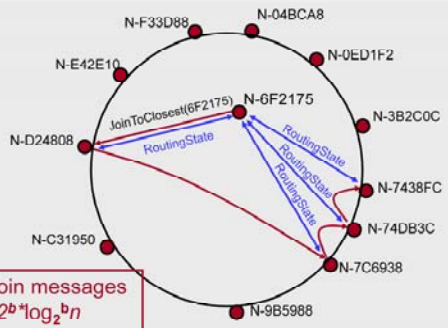


Mean lookup is $O(\log_2{}^b n)$
⟹ Scalable!

Pastry Node *X* Joins

- Node with *nodeID* = *X* knows about existing node with *nodeID A*
  - » *X* contacts *A* and sends *join* message with key=*X*
  - » Pastry nodes route message to some *nodeID* = *Z* that is numerically closest to *X*
  - » All nodes contacted in routing the *join* message return their routing state to *X*
- Reallocating Key/Value mappings to nodes is left to the application (it is notified of changes).

Pastry Join Protocol



Mean Join messages are ~$3*2^b*\log_2{}^b n$
Scalable!

Pastry Node *X* Joins (cont.)

- Z has the *nodeID* numerically closest to *X* so its Leaf Set becomes the base for *X*'s Leaf Set
- For Routing Table (*RT*) rows at *X*:
  - » $RT_X[0] = RT_A[0]$
    - row zero independent of *nodeID* prefix in all nodes
  - » $RT_X[i] = RT_i[i]$
    - The i^{th} row of *X*'s routing table can be taken from the i^{th} row of the table in the i^{th} node encountered while routing to *Z*.
    - This works because *X*'s *nodeID* shares a prefix at least as long as each successive *nodeID* along the path to *Z*.
- X transmits a copy of its new routing state to each *nodeID* found in its state.

Node Failure/Leave

- Failure of a node in some node's Leaf Set (detected with periodic pings):
  - » Contact node in Leaf Set with largest *nodeID* on the side (larger *vs* smaller) with the failed node and get its Leaf Set. It will contain an appropriate replacement.
- Failure of a node in some node's Routing Table (detected on attempt to contact during routing):
  - » Contact nodes in same (or higher, if necessary) row(s) as the failing entry and ask for one of their entries with the appropriate prefix.

36

DHTs discussion

- What systems can you build using DHTs?
- Is node diversity useful?
- How to reduce stretch?
- How to support range requests or partial matches between request and key?
- What real applications use DHTs today?
  - » Why or why not?
- Pros and cons of an unstructured p2p system?

3-37

37