PAST: P2P Storage Utility

References:

1. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proceedings of IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), November 2001.
2. A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In Proceedings of ACM Symposium on Operating Systems Principles (SOSP'01), October 2001.
3. P. Druschel, Presentation on Pastry and PAST. http://www.cs.rice.edu/~druschel/comp413/lectures/Pastry.ppt

Presentation by Nathaniel Crowell

Outline

- Review Structured P2P
- Pastry
- Pastry Routing
- PAST

Structured P2P Systems (recall Chord)

- Requirements of Basic Protocol:
    - Route to a node corresponding to a provided key-id
- Qualities:
    - Distributed
    - Decentralized Control
    - Self-organizing
    - Symmetry
- Potential Problems:
    - Maintenance of overlay network
    - Load Balancing
    - Routing efficiency measured based on overlay

How is Pastry Different?

Chord

- Skip list
  - Routing: O(log N)
  - Memory: O(log N)

How do they differ when b=1?

Pastry

- Routing table
  - Routing: $O(\log_c N)$

Memory: $O(c * \log_c N)$

c is 2^b

- Leaf Set
  - L closest NodeIds
  - Efficient Routing
  - Robustness
- Neighborhood Set
  - Proximity metric
  - Efficient routing

Pastry: Routing Table

NodeId 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Routing table

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

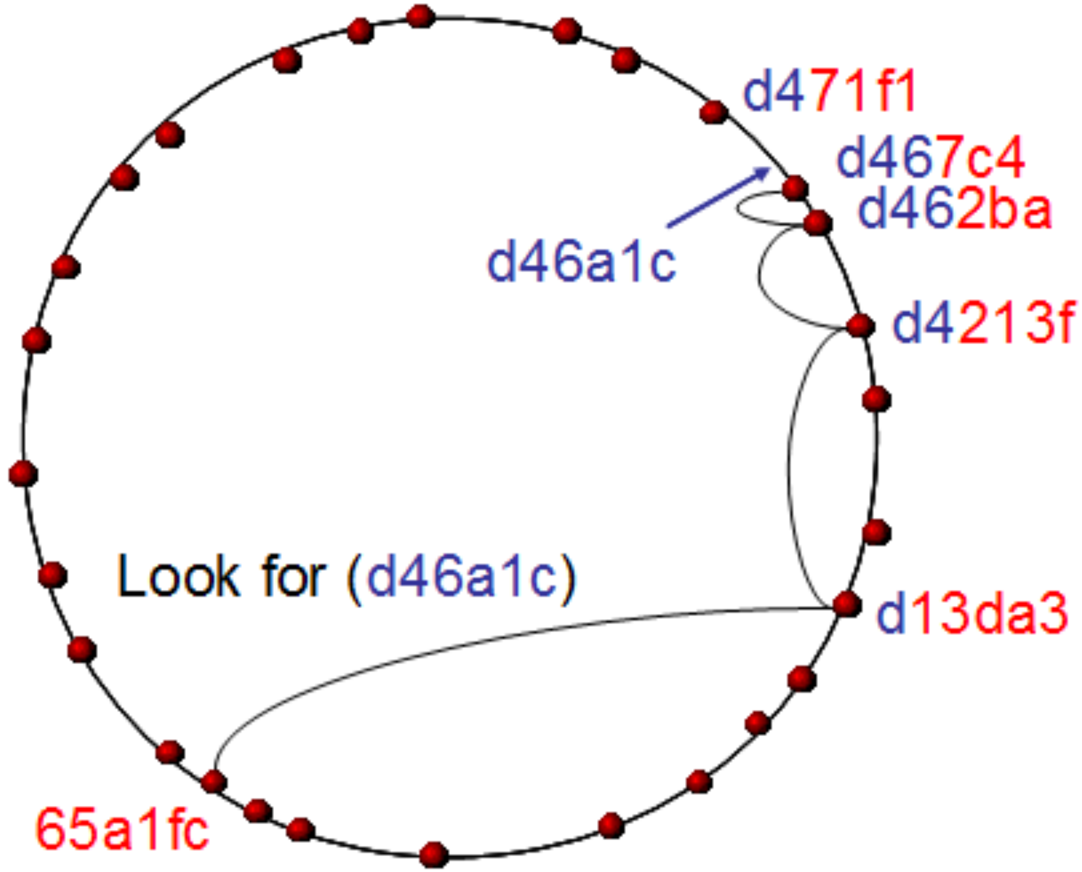
Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

- Leaf Set
  - Nearest by NodeId
- Routing Table
- Neighborhood Set
  - Nearest by proximity

- Maintenance more costly than Chord but not overwhelming

Pastry: Routing



b is 4 in this example

Pastry: Routing Procedure

if (destination is within range of our leaf set)
forward to numerically closest member
else
let l = length of shared prefix
let d = value of l-th digit in D’s address
if (Rld exists)
forward to Rld
else
forward to a known node that
(a) shares at least as long a prefix
(b) is numerically closer than this node

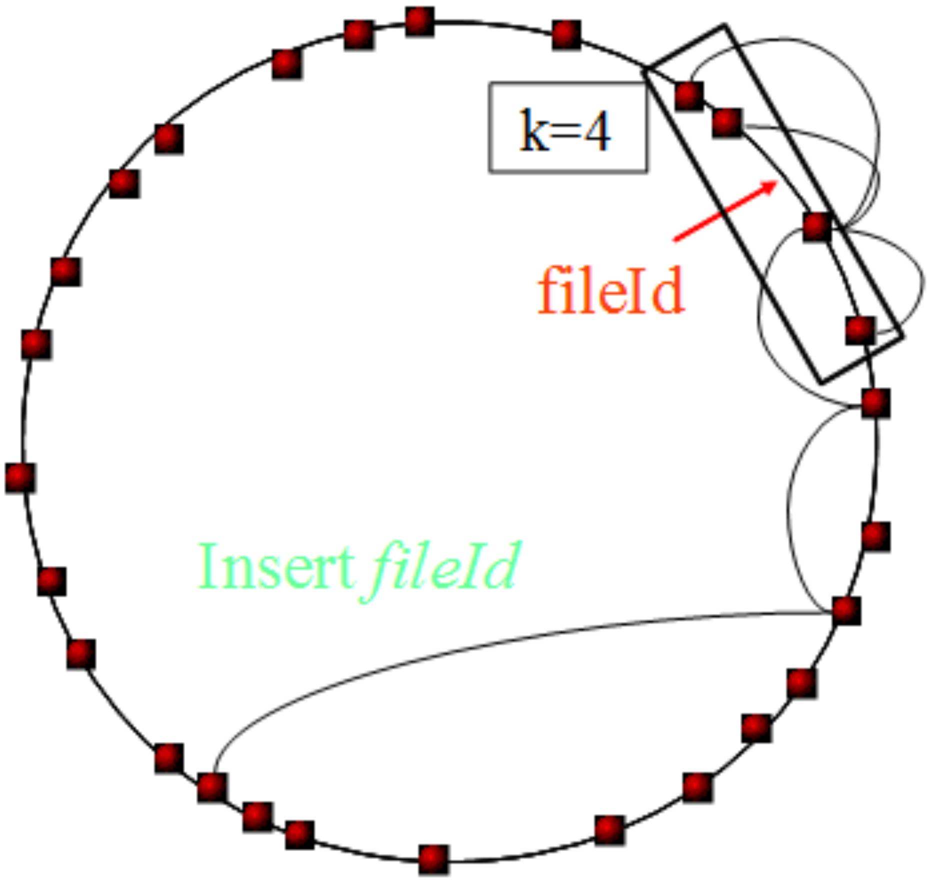
Pastry: Additional Details

- Routing Table
  - Preference given to nodes with better proximity.
- Performance
  - Success provable unless L/2 adjacent failures.
  - Routing usually O(log N), worst case O(N)
- Like Chord has additional functionality for
  - Node addition
  - Node departure
- Experiments show expected results
  - Protocol works under stated assumptions
  - Efficiency deteriorates without repair from failures

PAST: P2P Storage and Distribution

- Goals:
  - Strong persistence
  - High availability
  - Scalability
  - Security
- Simple API:
  - Insert
  - Lookup
  - Reclaim (delete)

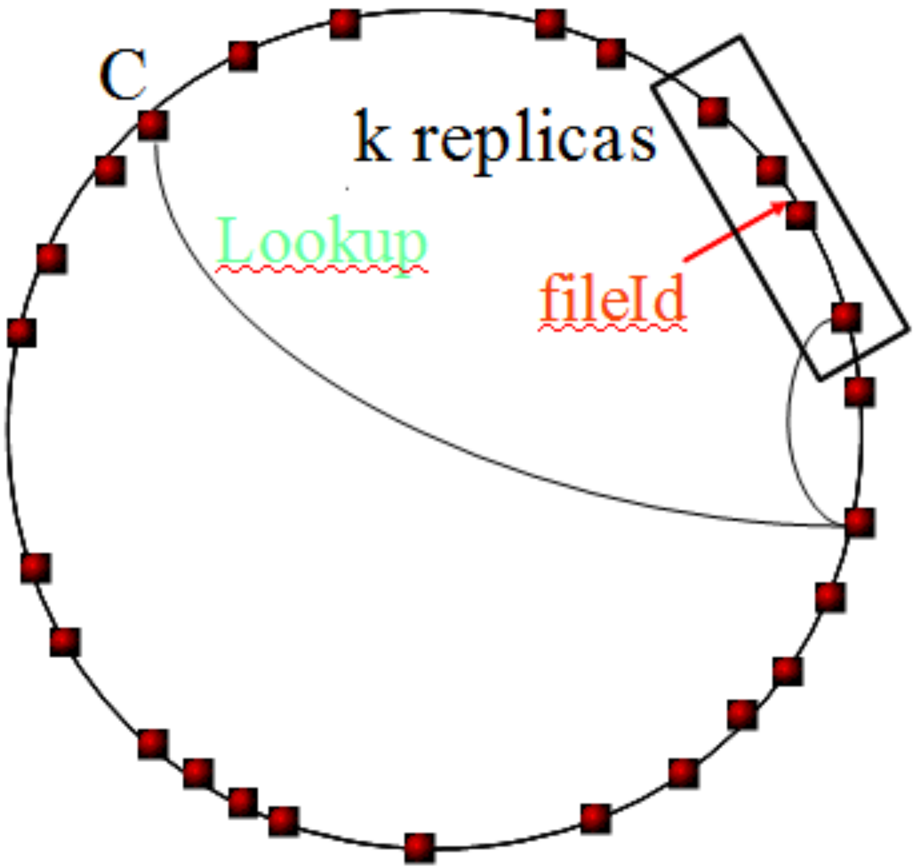
PAST: File Storage



Storage Invariant:
File “replicas” are stored on k nodes with nodeIds closest to fileId

(k is bounded by the leaf set size)

PAST: File Retrieval



file located in O(log N) steps (expected)

usually locates replica nearest client C

PAST: Exploiting Pastry

- Random, uniformly distributed nodeIds
  - replicas stored on diverse nodes
- Uniformly distributed fileIds
  - e.g. SHA-1(filename,public key, salt)
  - approximate load balance
- Pastry routes to closest live nodeId
  - availability, fault-tolerance

PAST: Diversion

- Diversion occurs as available space decreases
- Replica Diversion
  - Load-balancing over a leaf set
- File Diversion
  - Load-balancing over entire network
- Experiment results: Diversion necessary
  - No diversion ($t_{pri} = 1$, $t_{div} = 0$):
    - max utilization 60.8%
    - 51.1% inserts failed
  - Replica/file diversion ($t_{pri} = .1$, $t_{div} = .05$):
    - max utilization > 98%
    - < 1% inserts failed

PAST: Caching

- Nodes cache files in the unused portion of their allocated disk space
- Files caches on nodes along the route of lookup and insert messages

Goals:

- maximize query throughput for popular documents
- balance query load
- improve client latency

Questions?