

CSE 123b Communications Software

Spring 2003

Lecture 14: Peer-to-Peer Networks

Stefan Savage

Some slides courtesy Ion Stoica and Srinu Seshan

Peer-to-peer systems

- Examples
 - Napster, Gnutella, Freenet, KaZaA, CFS, etc.
- Definition?
 - No distinction between client and server
 - All nodes are potential users of a service AND potential providers of a service

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

2

Classifications

- What resource is shared?
 - CPU: SETI@Home
 - Storage & BW: most of the rest
- How are resources located?
 - Centralized systems
 - » Napster, Seti@Home
 - Distributed systems
 - » Unstructured: e.g. Gnutella
 - » Structured/routed: e.g. CFS/Chord, Freenet
- Search vs Lookup

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

3

Challenges

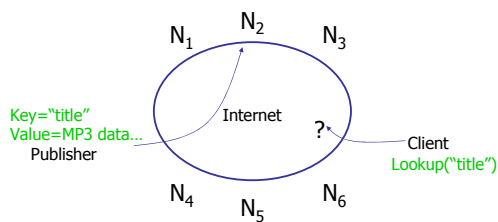
- Dynamic availability – machines go up & down
- Scale – millions of nodes
- Heterogeneity – some have big CPU/mem
- Management – each user controls own node
- Security – what about compromised nodes?
- Fairness – what about freeloaders?
- Performance

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

4

The Lookup Problem

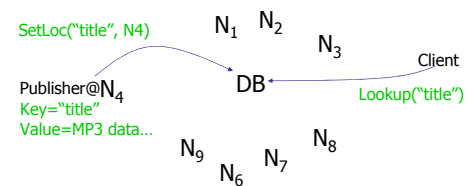


May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

5

Centralized Lookup (Napster)



Simple, but $O(N)$ state and a single point of failure

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

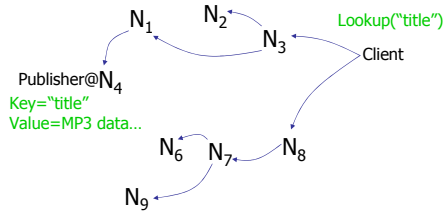
6

- Simple

Napster

- Simple centralized scheme
- How to find a file:
 - On startup, client contacts central server and reports list of files
 - Query the index system
 - » Returns addresses of one or more machines that claim to have file
 - Fetch the file directly from peer
- Advantages:
 - Simplicity, easy to implement sophisticated search engines on top of the index system
- Potential disadvantages:
 - Scalability, single point of failure (technical and legal)

Flooded Queries (Gnutella)



Robust, but worst case $O(N)$ messages per lookup

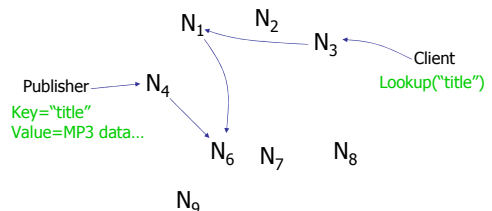
Gnutella

- Distributed location information
- Idea: *multicast* the request
- How to find a file:
 - Send request to *all* neighbors
 - Neighbors recursively multicast the request
 - Eventually a machine that has the file receives the request, and it sends back the answer
- Advantages:
 - Totally decentralized, highly robust
- Disadvantages:
 - Not scalable; the entire network can be swamped with request (can alleviate some of this problem if each request has a TTL)

Gnutella Details

- Basic message header
 - Unique ID, TTL, Hops
- Message types
 - Ping – probes network for other nodes
 - Pong – response to ping, contains IP addr, # of files, # of Kbytes shared
 - Query – search criteria + speed requirement of node
 - QueryHit – successful response to Query, contains addr + port to transfer from, speed of node, number of hits, hit results, node ID
 - Push – request to node ID to initiate connection, used to traverse firewalls
- Ping, Queries are flooded
- QueryHit, Pong, Push reverse path of previous message

Routed Queries (Freenet, Chord, etc)



Example: Freenet

- Architecture:
 - Each file is identified by a **unique** identifier
 - Each machine stores a set of files, and maintains a "routing table" to route the individual requests
- Additional goals to file location:
 - Provide publisher anonymity, security
 - Resistant to attacks – a third party shouldn't be able to deny the access to a particular file (data item, object), even if it compromises a large fraction of machines

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

13

Freenet Query

- User requests key XYZ – not in local cache
- Looks up nearest key in routing table and forwards to corresponding node
- If request reaches node with data, it forwards data back to upstream requestor
 - Requestor adds file to cache, adds entry in routing table
 - Any node forwarding reply may change the source of the reply → helps anonymity
- If data not found, failure is reported back

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

14

Data Structure

- Each node maintains a common stack
 - id* – file identifier
 - next_hop* – another node that store the file id
 - file* – file identified by *id* being stored on the local node
- Forwarding:
 - Each message contains the file *id* it is referring to
 - If file *id* stored locally, then stop
 - Forwards data back to upstream requestor
 - Requestor adds file to cache, adds entry in routing table
 - If not, search for the "closest" *id* in the stack, and forward the message to the corresponding *next_hop*

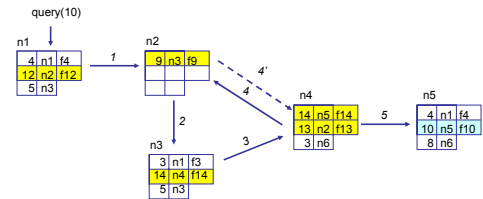
<i>id</i>	<i>next_hop</i>	<i>file</i>

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

15

Query Example



Note: doesn't show file caching on the reverse path

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

16

Freenet Summary

- Advantages
 - Totally decentralize architecture → robust and scalable
- Disadvantages
 - Does **not** always guarantee that a file is found, even if the file is in the network

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

17

Example: Chord

- Associate to each node and item a unique *id* in a *uni*-dimensional space
- Goals
 - Scales to hundreds of thousands of nodes
 - Handles rapid arrival and failure of nodes
- Properties
 - Routing table size $O(\log(N))$, where N is the total number of nodes
 - Guarantees that a file is found in $O(\log(N))$ steps

May 21, 2003

CSE 123b – Lecture 14 – Peer-to-Peer Networks

18

Data Structure

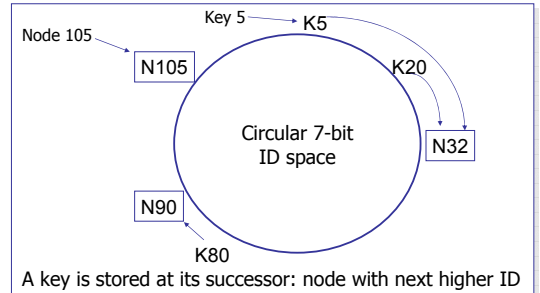
- Assume identifier space is $0..2^m$
- Each node maintains
 - Finger table
 - » Entry i in the finger table of n is the first node that succeeds or equals $n + 2^i$
 - Predecessor node
- An item identified by id is stored on the successor node of id

May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

19

Consistent Hashing [Karger 97]

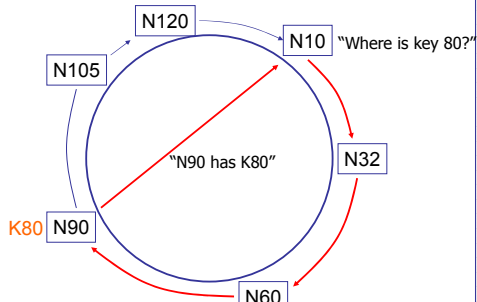


May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

20

Basic Lookup



May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

Simple Lookup Algorithm

```

Lookup(my-id, key-id)
  n = my successor
  if my-id < n < key-id
    call Lookup(id) on node n // next hop
  else
    return my successor // done
    
```

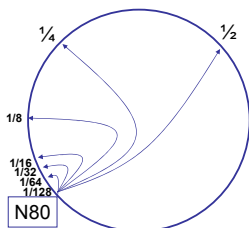
- Correctness depends only on successors
- $O(N)$ "hops" to find item

May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

22

"Finger table" - $\log(N)$ -time lookups

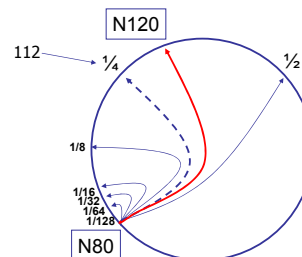


May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

23

Finger i Points to Successor of $n+2^i$



May 21, 2003

CSE 123b - Lecture 14 - Peer-to-Peer Networks

24

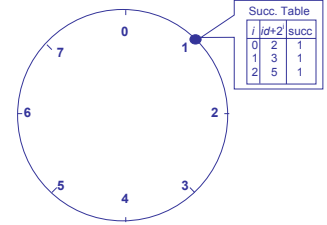
Lookup with Fingers

```

Lookup(my-id, key-id)
  look in local finger table for
    highest node n s.t. my-id < n < key-id
  if n exists
    call Lookup(id) on node n // next hop
  else
    return my successor // done
    
```

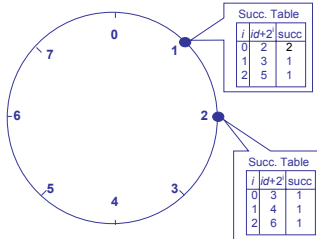
Chord Example

- Assume an identifier space 0..8
- Node n1:(1) joins → all entries in its finger table are initialized to itself



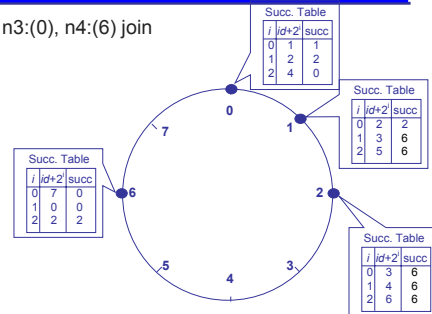
Chord Example

- Node n2:(3) joins



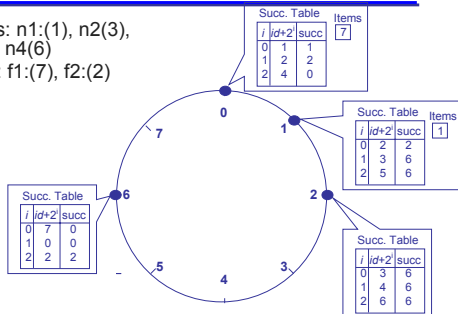
Chord Example

- Nodes n3:(0), n4:(6) join



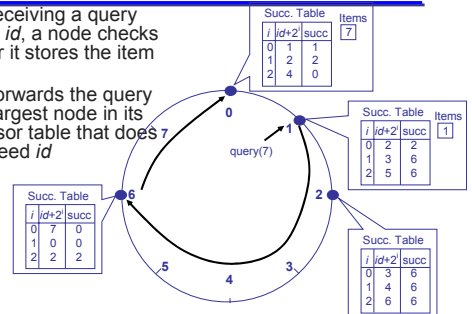
Chord Examples

- Nodes: n1:(1), n2:(3), n3(0), n4(6)
- Items: f1:(7), f2:(2)



Query

- Upon receiving a query for item id , a node checks whether it stores the item locally
- If not, forwards the query to the largest node in its successor table that does not exceed id



Chord Summary

- $O(\log N)$ guaranteed lookup performance
 - Possible to do even better: $O(\log \log N / \log n)$
- Lookup semantics, not search
- Performance: routing in the overlay network can be more expensive than in the underlying network
 - Because usually there is no correlation between node ids and their locality; a query can repeatedly jump from Europe to North America, though both the initiator and the node that store the item are in Europe!
 - Partial solution: Weight neighbor nodes by RTT
 - » when routing, choose neighbor who is closer to destination with lowest RTT from me

Discussion

- Freeloading problem
 - Does everyone participate?
- Trust?
- Availability/reliability?

Summary

- A key challenge of building wide area P2P systems is a scalable and robust location service
- Solutions covered in this lecture
 - Napster: centralized location service
 - Gnutella: broadcast-based decentralized location service
 - Freenet: intelligent-routing decentralized solution (but correctness not guaranteed; queries for existing items may fail)
 - Chord (and others): intelligent-routing decentralized solution
 - » Guarantee correctness
 - » May not be efficient
- Lots of open questions

Next time

- Network Security