Agenda

- **Last time**
  - Finish naming: DNS
  - P2P
  - PA#2 back
- **This time**
  - P2P
  - Midterm back
  - Assignment #3 due
- **Next time (tonight/tomorrow)**
  - Time and global states

Schedule

Sun	Tues	Thurs	Fri
	DNS P2P (10) 27	P2P, #3 due,#4 out Time/global(11) 29	Time/global (11) 30
	Time/global PA#5 Out 3	Coordination/agreement (12) 5	
	Coordination/agreement PA#4 due 10	Transactions (13) PA#5 proposal due 12	
	Transactions (13) Dis transactions (14) 17	Replication (15) 19	
	Student presentations PA#5 due 24	Student presentations 26	

Before we start

- **Old**
  - Assignment #4 (no book): due Tues Apr 10
  - Assignment #5 (no book): out Tues April 3, proposal due Thurs April 12, due April Tues 24
- **New ?????**
  - Assignment #4 (no book):
    - P2P out now, due Tues April 17 OR
    - "comparable" project proposal due Thurs Apr 5, due Tues Apr 17
  - Assignment #5 (only book): out April 5, due Tues April 24

Chapter 10: P2P

- **Client/Server vs. P2P**
  - Where is the information and/or computational capacity?
- **Range of architectures from Client/Server to P2P**
- **5 general properties/characteristics of P2P systems**
  1. Each user contributes resources to the system
  2. Although each may contribute different resources, every node generally has the same functional capabilities and responsibilities
  3. Their correct operation does not depend on the existence of any centrally-administered systems
  4. Nodes arrive and depart continuously
  5. *Key issue: how to (dynamically) place/locate data across nodes?*

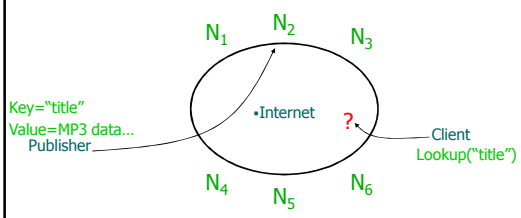
Generations of P2P systems

- **Gen I: Exploit PC resources**
  - Napster
- **Gen II: Eliminate centralized components**
  - Freenet, Gnutella, KaZaa
- **Gen III: Use structured overlays to guarantee number of network hops to find information**
  - Pastry, Tapestry, CAN, Chord

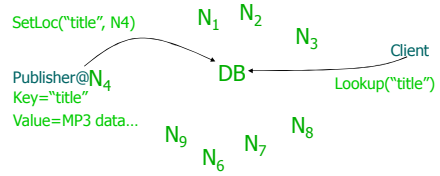
P2P: Overlay Networks

- **Overlays: all in the application layer**
  - Flexibility: protocol flexibility, messaging over TCP or UDP
  - Underlying physical net is transparent to the developer
- **Overlay consists of nodes and arcs**
  - Nodes: participants
  - Arcs: a logical communication channel between two nodes
- **Unstructured:**
  - arcs appear to be random (arbitrary)
  - No correlation between peer and content managed by it
- **Structured:**
  - employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired content
  - Example: DHTs

P2P: The lookup problem

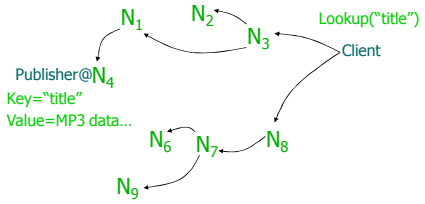


Centralized lookup (Napster)



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

Flooded queries (Gnutella)



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

Gnutella Issues

- **How much traffic does one query generate?**
- **How many hosts can it support at once?**
- **What's the latency associated with querying?**
- **Is there a bottleneck?**

(Other) Problems of Gnutella

- **Limited horizon**
  - The number of reachable nodes is limited by *TTL* and the number *N* of concurrent connections
    - Example: tree network, TTL=5, N=4, nodes = 4^5 = 1024
  - Solution: <u>increase TTL and N</u>
- **Scalability**
  - The number of messages exchanged increases exponentially with the increase of TTL and N
    - Example: tree network, TTL=5, N=4, messages = 4^5 = 1024
  - Solution: <u>decrease TTL and N</u>
  - Solution: caching policies based on query popularity
    - over 50% cache hits with 5-minute caching

(Other) Problems of Gnutella

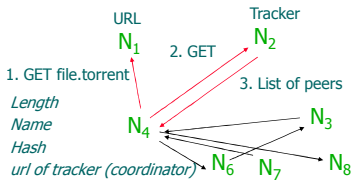
- **Denial of Service attacks**
  - Flooding the system with requests
    - Strange traffic observed in Gnutella
  - Solution: keep statistics about frequency of requests and close connections with offending nodes
- **Privacy attacks**
  - A site advertised file names that appeared to offer a popular new music CD
  - It logged the IP address and domain name of every download request (included in HTTP)
  - Solutions: none at present

KaZaA

- **"Fixes" some things in "vanilla gnutella"**
  - Hierarchy: cross between gnutella and napster ("supernodes")
  - Queue management
  - Parallel download

BitTorrent

- **Bram Cohen, 2001-2003**
- **"Traditional": I provide content, I pay huge server costs (*bandwidth and/or server crashes)* as people grab it**
- **Basic idea:**
  - If you download file X, at the same time you also upload file X to someone else ("clients bring both supply and demand – not just demand")



More on BitTorrent

- **Peer distributing the file:**
  - Breaks content down into equal-size pieces (64K-1M), checksummed
- **Seeders – peers that provide the complete file**
- **Swarm -- a group of peers connected to each other to share a torrent**
- **Usually tracker-based**
  - in a trackerless system (decentralized tracking) every peer acts as a tracker (DHT)
- **Clients download segments in a random order**
- **Bit torrent and copyright material: hmmmm….**
  - Bit torrent does not actually provide a "search" mechanism
  - BitTorrent makes no attempt to conceal the host ultimately responsible for facilitating the sharing (the *torrent* file)
  - Feb 26, 2007: BitTorrent, Inc: now offering file-studio-sanctioned downloads

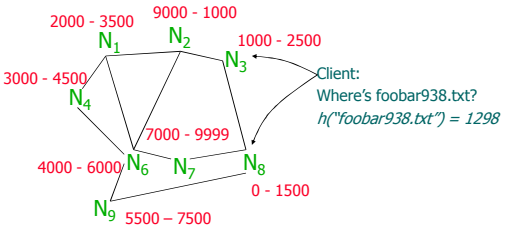
Intro to DHTs

- **Problem: finding something in a P2P system**
  - Central DB and floods both have problems
- **DHT-based P2P: A distributed system in which…**
  - Responsibility for maintaining the mapping from names to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption.

DHT: Locating Content via Directed Searches

- **Instead of flooding the network, assign particular nodes to hold particular content (or pointers to the content, like an "information booth")**
- **Challenges**
  - "evenly" distribute responsibilities over participants in the overlay
  - Nodes can enter and leave dynamically
    - Load balancing AND even finding the content in the first place!

Intro to DHTs



Nodes do not have to have the content directly, just *know* where the content is.

Routing

- **Client finding the content; client does not necessarily have to:**
  - Go *directly* to node that knows about knows about hash content
  - Go *directly* to node that has content
- **But "success" = "short path"**
- **The different approaches differ primarily in the routing approach (e.g., O(log n))**
  - CAN, Chord, Pastry, Tapestry

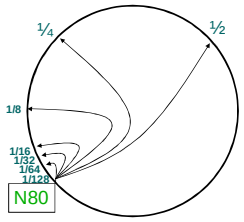
DHT API

- **Each data item has a key in some ID space**
- **DHT API:**
  - *Lookup(data item) ➔ IP address of the node(s) that is responsible for the data item*

DHT Implementation Issues

1. **Mapping keys to nodes in a load-balanced way**
   - Each key is stored at one or more nodes whose IDs are "close" to the key in the ID space.
2. **Forwarding a lookup for a key to an appropriate node – must get "closer"**
3. **Distance function (how to implement "closeness"?)**
   - Chord: numeric difference between two IDs
   - Pastry and Tapestry: the number of common prefix bits
4. **Building routing tables adaptively**
   - Must tolerate asynchronous and concurrent node joins and failures

Chord: "Finger table"



- **Chord**
  - Finger table: O(log N)
    - Contains IP addr of nodes in ID space
  - A node forwards a query for key k to the node in its finger table with the highest ID not exceeding k
  - Node IDs assigned randomly