416 Distributed Systems

Mar 1, Peer-to-Peer
Outline

• P2P Lookup Overview

• Centralized/Flooded Lookups

• Routed Lookups – Chord
Scaling Problem

- Millions of clients $\Rightarrow$ server and network meltdown
P2P System

- Leverage the resources of client machines (peers)
  - Traditional: Computation, storage, bandwidth
  - Non-traditional: Geographical diversity, mobility, sensors!
Peer-to-Peer (storage) Networks

• Typically each member stores/provides access to content
• Basically a replication system for files
  • Always a tradeoff between possible location of files and searching difficulty
  • Peer-to-peer allow files to be anywhere → searching is the challenge
  • Dynamic member list makes it more difficult
• What other systems have similar goals?
  • Routing, DNS
The Lookup Problem

Key=“title”
Value=MP3 data...
Publisher

Internet

N₁
N₂
N₃
N₄
N₅
N₆

Client
Lookup(“title”)
Searching

• Needles vs. Haystacks
  • Searching for top 40, or an obscure punk track from 1981 that nobody’s heard of?

• Search expressiveness
  • Whole word? Regular expressions? File names? Attributes? Whole-text search?
Framework

- Common Primitives:
  - **Join**: how do I begin participating?
  - **Publish**: how do I advertise my file?
  - **Search**: how to I find a file?
  - **Fetch**: how to I retrieve a file?
Outline

• P2P Lookup Overview
• Centralized/Flooded Lookups
• Routed Lookups – Chord
Napster: Overview

- Centralized Database:
  - **Join**: on startup, client contacts central server
  - **Publish**: reports list of files to central server
  - **Search**: query the server => return someone that stores the requested file
  - **Fetch**: get the file directly from peer
Napster: Publish

I have X, Y, and Z!

123.2.21.23

insert(X, 123.2.21.23)
Napster: Search

Where is file A?

123.2.0.18

search(A) --> 123.2.0.18

Fetch

Query

Reply

Where is file A?
Napster: Discussion

- **Pros:**
  - Simple
  - Search scope is $O(1)$
  - Controllable (pro or con?)

- **Cons:**
  - Server maintains $O(N)$ State
  - Server does all processing
  - Single point of failure
“Old” Gnutella: Overview

- **Query Flooding:**
  - **Join**: on startup, client contacts a few other nodes; these become its “neighbors”
    - “unstructured overlay”
  - **Publish**: no need
  - **Search**: ask neighbors, who ask their neighbors, and so on... when/if found, reply to sender.
    - TTL limits propagation
  - **Fetch**: get the file directly from peer
Gnutella: Search

I have file A.

Where is file A?

Reply

I have file A.
Gnutella: Discussion

• Pros:
  • Fully de-centralized
  • Search cost distributed
  • Processing @ each node permits powerful search semantics

• Cons:
  • Search scope is $O(N)$
  • Search time is $O(???)$
  • Nodes leave often, network unstable

• TTL-limited search works well for haystacks.
  • For scalability, does NOT search every node. May have to re-issue query later; no guarantee that it will find the file!
Flooding: Gnutella, Kazaa

- Modifies the Gnutella protocol into two-level hierarchy
  - Hybrid of Gnutella and Napster
- Supernodes
  - Nodes that have better connection to Internet
  - Act as temporary indexing servers for other nodes
  - Help improve the stability of the network
- Standard nodes
  - Connect to supernodes and report list of files
  - Allows slower nodes to participate
- Search
  - Broadcast (Gnutella-style) search across supernodes
- Disadvantages
  - Kept a centralized registration → allowed for law suits 😞