Peer-to-Peer Networks

CS 257/457
Dept. of Computer Science, University of Rochester

Case Study I: Napster (central index)

Napster central database
P-Diddy.mp3
Kai
MP3s I have
Who has P-Diddy.mp3?
Kai has it
P-Diddy.mp3
Amy
MP3s Amy has

Centralized index, distributed document repository.

Case Study II: Gnutella (query flooding)

Who has P-Diddy.mp3?
I do
I do
I do
Every node forwards each query to a neighbor heuristically closer to "L"
Who has P-Diddy.mp3?
I do
big questions remain!

Case Study III: DHT (Distributed Hashtable)

Here is P-Diddy.mp3, store it somewhere.
"P-Diddy.mp3" is deterministically hashed into location "L"

Every node forwards each query to a neighbor heuristically closer to "L"
Structured Distributed Hashtable

Here is P-Diddy.mp3, store it somewhere.

Can I have P-Diddy.mp3?

Analysis

Content-addressable network [Ratnasamy et al. 2001]:
- each node owns a region in a virtual 2-D space
- "P-Diddy.mp3" is hashed into a virtual location "L" (1.3, 2.4)
- every node forwards each query to a neighbor heuristically closer to "L"

Questions:
- Space consumption? Lookup cost for a network of N nodes?
  - 4 links per node; O(N^{1/2})
  - Can the lookup cost be improved?
  - How to take into account the actual link latency?

Chord [Stoica et al. 2001]

- each object is mapped to the first node in the clockwise direction on the ring.
- every node maintains links to
  - the node half-way in the circular ID space
  - the node a quarter-way across the circular ID space
  - ...
- Space consumption? Lookup cost for a network of N nodes?

Peer-to-peer Systems: Definition and Desirables

- Peer-to-peer systems: distributed systems of no hierarchy, equal-status.
  - Napster? Gnutella? DNS? Distributed hashtables (CAN and Chord)?
- Scalability: able to support large number of nodes
  - cost of each operation is below linear-scaling – goes up slowly when the system size increases: O(n) is terrible, O(n^{1/2}) is OK, O(log n) is good.
  - space requirement at each node is below linear-scaling.
- Self-healing:
  - node failures should not cripple the system.
  - automatic stabilization ⇒ fix non-transient failures
What makes peer-to-peer different?

- Compare with hierarchical, non equal-status systems:
  - better scalability - no performance bottleneck.
  - better availability - more tolerant to random failures, intentional attacks, or censorship.

- Challenges:
  - peer coordination without complete global knowledge

Where are we now?

- Concept of peer-to-peer systems.
- Three approaches for object location discovery in large-scale distributed systems:
  - central index
  - query flooding
  - hash + heuristics-based routing

- Many other problems:
  - load balancing, replication, ...

- Keyword search in peer-to-peer fashion.

Keyword Search

- user inputs a few keywords, the system returns a list of documents matching the keywords - Google

- Google maintains a central search index:
  - a search index contains a list of all searchable words, each of which contains a list of documents relevant to the word
  - intersection of document lists for multiple-word queries

Java:

| Page #123 | Page #157 | ... | ...

Sun:

| Page #157 | Page #468 | ... | ...

... 

Peer-to-peer Keyword Search

- What is "peer-to-peer keyword search"?
  - splitting the central search index into smaller pieces and distributing them in peer-to-peer fashion

- Why is it better?
  - better scalability?
  - better availability?

- Challenge:
  - make them work collaboratively to achieve similar speed and quality of search
Peer-to-peer Keyword Search

Solution 1: Split based on keywords

Split the index database to many pieces based on keywords and distribute them to many nodes.

Central index - Google

Distributed index

Solution 2: Split based on documents

Split the index database to many pieces based on documents and distribute them to many nodes.

Central index - Google

How to build a peer-to-peer keyword search engine?

- Split based on keywords
  - Weakness: transferring large data segments for multiple keyword queries.

- Split based on documents
  - Weakness: too many sites to visit for each query.

- Ideas to improve upon them?