Information Searching Methods In P2P file-sharing systems

Nuno Alberto Ferreira Lopes
PhD student
(nuno.lopes@di.uminho.pt)

Grupo de Sistemas Distribuídos
Departamento de Informática
Universidade do Minho

JOIN 2004

1. Introduction to Searching
   - Motivation
   - P2P Systems
   - Search Description
2. Search Methods in P2P Systems
   - Centralized Index
   - Decentralized Index
   - Distributed Index
3. Conclusions

Motivation

- Why do we need search on P2P file-sharing?
- Isn't the search problem already resolved?

Characterization of P2P systems

Peer-to-Peer (P2P) file-sharing systems can be characterized by having:

- high number of nodes,
- dynamic node membership,
- dynamic node contents,
- two independent file functions: search/location and downloading.

Requirements For P2P Systems

Administrative requirements

- Scalability up to millions of nodes
- Completely decentralized solution
- Correct system wide searches

Operational requirements

- minimize network load (message number and size)
- require small node storage and processing overhead

A search model

- Keyword matching search with *and* and *or* operators.
- Inverted Index model ($keyword \mapsto file_reference_{list}$)
- Announce and query operations are performed on the index

1 Introduction to Searching
- Motivation
- P2P Systems
- Search Description

2 Search Methods in P2P Systems
- Centralized Index
- Decentralized Index
- Distributed Index

3 Conclusions

Single centralized index

Features

- Multiple clients with a single server
- Server required for every search operation
- Example: Naspter

Advantages/Disadvantages

+ Complete system wide searching
+ Best search performance
– Single point of failure of server
– Single authority administration
– High resource demand on server

Multiple mini-centralized index

Features

- Static set of servers to which clients connect individually
- Examples: Edonkey and Direct Connect protocols

Advantages/Disadvantages

+ Searches continue to be as efficient as in a centralized solution
– Servers must support high network load
– Servers are static to clients which require a bootstrap server list

Completely decentralized index

Features

- Clients use neighbor broadcast routing
- Searching operates on a per-node basis
- Example: Gnutella protocol

Advantages/Disadvantages

+ No central authority
+ Individual node query processing
– Broadcast based queries with limited horizon

Multiple semi-decentralized index

Features

- No exclusive server role, clients become super-peers as needed, weaker clients connect to super-peers
- Newer systems try to optimize message routing to avoid broadcast but may limit search results
- Examples: FastTrack (kazaa), Gnutella2 (shareazaa)

Advantages/Disadvantages

+ Dynamic peer role according to its availability
– Super-peers require more resources available and continue to use message broadcast among them

Distributed Inverted Index

- Each client stores a piece of the index
- Popular keywords will create contention points on some clients and probably a storage overload
- Caching could be used to reduce hot spots, but must maintain data consistency across peers

Distributed Hash Table (DHT) Systems

- Examples: Chord, Pastry and others
- Stores ($hash_key \mapsto value$) pairs
- Scalable up to millions of nodes
- Efficient key location with logarithmic number of messages
- Scalable storage cost for management data on each peer
- **Unable to perform key searching!**

Possible solution for a distributed index implementation? ...

A Naive DHT Implementation

- Search keywords are converted into a hash key, list of file references is stored as corresponding value
- Searching operations are required to access individual keywords, any *and* or *or* operations must be performed on the client

This is the case for the *Overnet* P2P network, however:

- How will the system respond to a "mp3 + foo" query?
- Since cache is not used, the hot spot problem will arise for popular keywords; overloading specific peers
- Worst, for keywords which will have a high number of file references, clients that request them have to download the entire list.

Possible Solutions to Overcome The DHT Limitations

- Caching must be used to limit the number of accesses made to peers storing popular keywords.
- The file reference list of very popular keywords may not even fit into a single peer; such list be splitted in smaller pieces across several peers.
- Both *and* and *or* operations must be performed at the peers storing the lists.
- Again, the caching technique must maintain consistency of data across peers

Conclusions

- Centralized P2P systems do provide the best searching performance.
  However, its design does not scale efficiently to millions of nodes.
- Current P2P systems are not well suited for a truly P2P environment.
- DHTs are very efficient in resource usage but they lack the search functionality.
  Using specific extensions to such systems, it could be possible to obtain a system that could be both scalable and fully searchable according to P2P requirements.