

Searching in peer to peer networks

Jan Demter
jan@demter.de

Institut für Informatik
Georg-August Universität Göttingen

Advanced Topics in Computer Networking
July 1. 2005

What is a peer to peer network (P2P)?

- ▶ a (probably only partially) decentralized (overlay) network where endnodes are directly communicating with each other
- ▶ Examples:
 - ▶ Freenet (1999, Clarke)
 - ▶ Napster (1999, Fanning)
 - ▶ Gnutella (2000, Frankel/Pepper)
 - ▶ eDonkey (2000, McCaleb)
 - ▶ FastTrack (2001, Zennström/Friis, Kazaa, Morpheus etc., based on Gnutella)
 - ▶ BitTorrent (2002, Cohen)

What is searching?

- ▶ here: trying to find existing data by some characteristic of it, i.e. a filename, words appearing in the data, size, etc.
- ▶ can be sped up a lot by building an index

What is an index?

- ▶ here: a data structure optimized for looking up entries in the structure

Challenges of searching in P2P networks

- ▶ an index has to be built:
 - ▶ What attributes of the data should be indexed?
 - ▶ Who stores the index?
 - ▶ How is the information on the peers indexed there?
 - ▶ How do peers search the index?
 - ▶ Index has to be kept up-to-date

- ▶ Centralized indexing:
 - ▶ Not P2P, index is stored on dedicated servers
 - ▶ examples are Napster and original eDonkey
 - ▶ a straightforward approach
 - ▶ vulnerable to attack: single point of failure
- ▶ Distributed indexing:
 - ▶ index is distributed among the peers
 - ▶ no single point of failure
 - ▶ but does of course not come for free:
 - ▶ searching and indexing consume more bandwidth
 - ▶ additional computation and storage costs

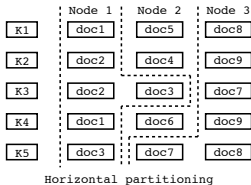
Indexing in practice

- ▶ Centralized indexing:
 - ▶ trivial, peers send their local index to the servers
 - ▶ peers direct their searches to the servers
- ▶ Distributed indexing:
 - ▶ depends...

Index partitioning

The index has to be divided among the peers:

- ▶ horizontal partitioning: each peers' index contains many documents and their keywords



- ▶ vertical partitioning: each peers' index only contains documents associated with one to a few keywords



Vertical partitioning

Index partitioning

Horizontal partitioning (local indexing):

- ▶ searching is expensive, broadcasting necessary
- ▶ updating the index is easy
- ▶ query throughput does not scale with additional peers
- ▶ Example: Gnutella

Vertical partitioning (global indexing):

- ▶ searching is cheap, at most one peer for every keyword has to be contacted
- ▶ updating the index is harder

Hybrid indexing:

- ▶ vertical partitioning, but each node has a horizontal index too
- ▶ saves bandwidth on multi-word queries
- ▶ trade-off: (much) higher storage costs

Problems with index partitioning

- ▶ "hot spots": few of the nodes are responsible for the most common keywords in the index
- ▶ security issues, malicious peers can probably alter results

Querying - naive approach

Horizontal partitioning:

- ▶ Easy: broadcast query into the network, receive replies

Vertical partitioning:

- ▶ Query nodes responsible for the given keywords, intersect their replies

Query optimization: Horizontal partitioning

Searching in peer
to peer networks

Jan Demter

Introduction

Challenges

Indexing

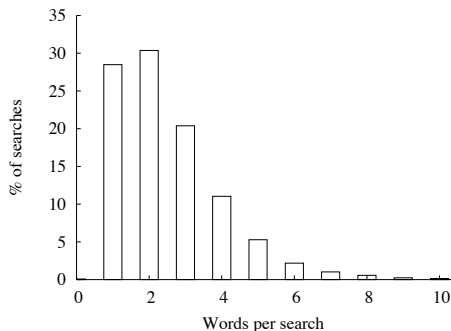
Distributed Indexing

Querying

- ▶ Caching gears up querying a lot

Query optimization: Vertical partitioning

Potentially huge overhead arises from the transmission of redundant results if one searches with multiple keywords, a regular case:



Mitigating overhead

- ▶ Bloom filters:
 - ▶ Bloom filters summarize set memberships with tunable accuracy at the cost of falsely positive membership tests.
- ▶ Idea: Send multi-word query to one of the responsible peers, (a), (a) then calculates Bloom filter of matching documents in his index, hands filter on to a node responsible for another word in the query, (b).
- ▶ (b) then computes results matching both terms, potentially plus some false positives, using the Bloom filter and sends it either back to (a), who could then remove false positives, or directly to the initiating peer

Mitigating overhead - continued

- ▶ Caching:
 - ▶ Caching of Bloom filters and queries
- ▶ Incremental results:
 - ▶ Peers exchange results incrementally, the user most likely cannot cope with all results in bigger networks anyway

Conclusions

- ▶ searching in traditional P2P-networks can be improved a lot
- ▶ but only at the expense of higher computation and storage costs

The End

Thank you for your attention.

Sources:

- ▶ 'Efficient Peer-to-Peer Keyword Searching', Patrick Reynolds and Amin Vahdat
- ▶ 'Hybrid Global-Local Indexing for Efficient Peer-to-Peer Information Retrieval', Chunqiang Tang and Sandhya Dwarkadas