Performance of Full Text Search in Structured and Unstructured Peer-to-Peer Systems

INFOCOM 2006

Presented by C.L. Wang

2006/08/02

Outline

- Introduction
- Peer-to-Peer Full Text Search Systems
- Peer-to-Peer Network Searches
  - Structured Networks
  - Super-peer Networks
  - Unstructured Networks
- Performance Comparison
- Conclusion

Introduction

- In full text keyword search the goal is to find text documents that match keyword queries.
- Many techniques for matching queries and documents have been developed in the field of information retrieval (IR).
- Information retrieval techniques usually focus on a centralized search server that either stores the documents directly or processes searches over documents stored at remote servers.

Introduction

- Peer-to-peer full text search systems are decentralized and massively distributed, with no “central server” per se.

- A peer-to-peer system may be chosen over a centralized IR server for a number of reasons：
  - the potential for massively parallel processing of searches
  - rapidly changing data that makes it difficult to keep a central IR server up to date
  - non-functional considerations (such as the unwillingness of peers to relinquish control to a central server)

Peer-to-Peer Full Text Search Systems

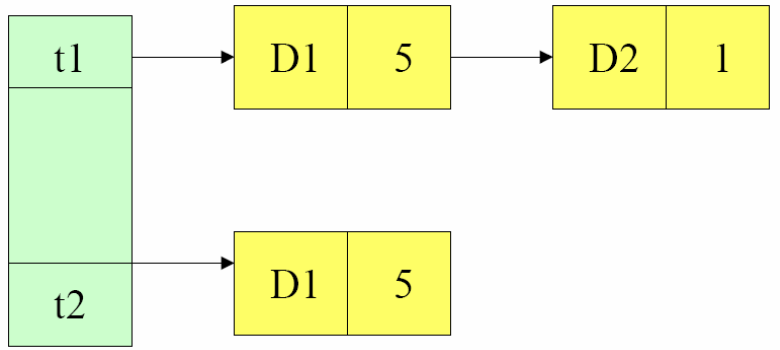
- It assumes the conjunctive query model, in which “matching” documents must contain all of the query terms.
- It evaluated both phases of full text keyword searching: document publishing and query processing.
- It measured the bandwidth used and the response time for both phases.

Peer-to-Peer Full Text Search Systems

- The traditional approach for conducting full text searches is to use an inverted index, which stores for each term a list of documents containing that term.

t_i : term

D_i : document



- Most searching operations can be efficiently executed by intersecting or unioning the inverted lists for different terms.

Peer-to-Peer Network Searches

- Structured peer-to-peer searches
  - DHT (specifically, Chord)
    - Bloom filter
    - Caching
- Super-peer searches
    - TTL
- Unstructured peer-to-peer searches
  - Random-walk
    - State-keeping
    - Square-root topology

Structured Networks

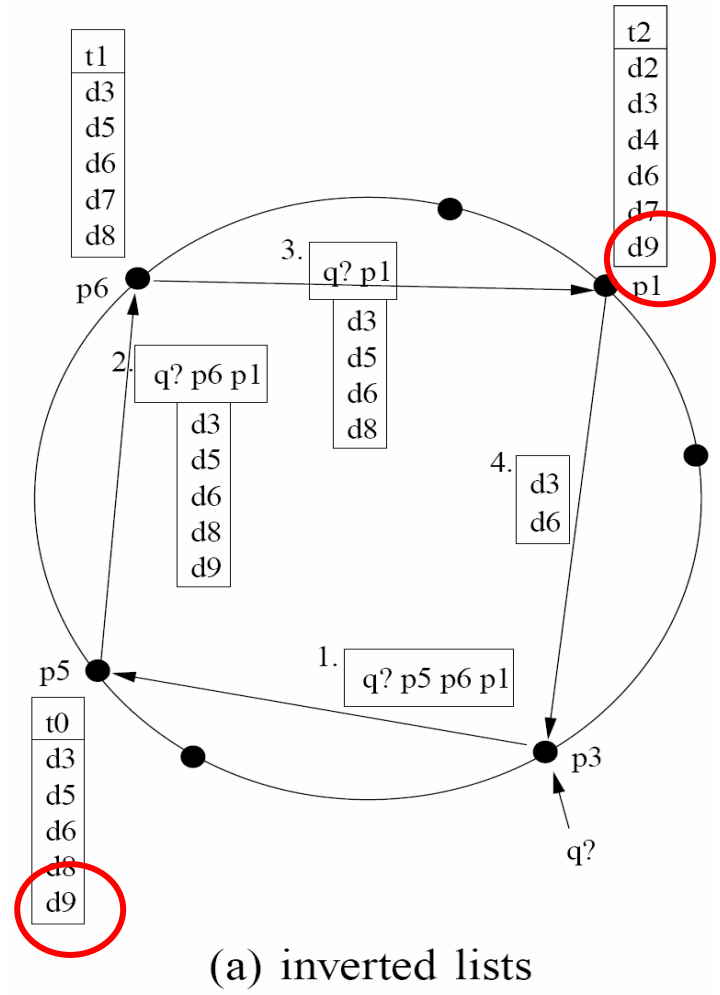
- The way to implement full text search is to use DHTs to store and retrieve inverted lists in a Chord network.
  - The “keys” inserted into the DHT would be the terms.
  - The “values” inserted would be the associated inverted lists.
- To conduct a search, it performs a lookup for each query term to retrieve its inverted list, and then intersect the inverted lists to obtain the list of matching documents.

DHT-Document Publishing

- The inverted lists would be constructed incrementally by "publishing" documents.
- Such publishing would take place when a peer joins the network, or adds a new document to a collection.
- To publish a document, the peer must parse its documents to extract the terms, and then send a *publish*(*term, documentID*) message for each term.
- The DHT would route each publish message to the peer responsible for the given term; this peer would then add the document ID (and the ID of the peer holding the document) to its inverted list for that term.

DHT-Document Publishing Example

- To publish a document *d*, the publishing peer p_d sends a *publish*(*ti*, (*d*, p_d)) message for each distinct term *ti* in *d*.
  - Ex：d9-t0,t2
- Chord routes the message to the peer p_{ti} assigned to term *ti*.
- Peer p_{ti} adds the ID of document *d* (and the peer ID p_d) to the inverted list of term *ti*.

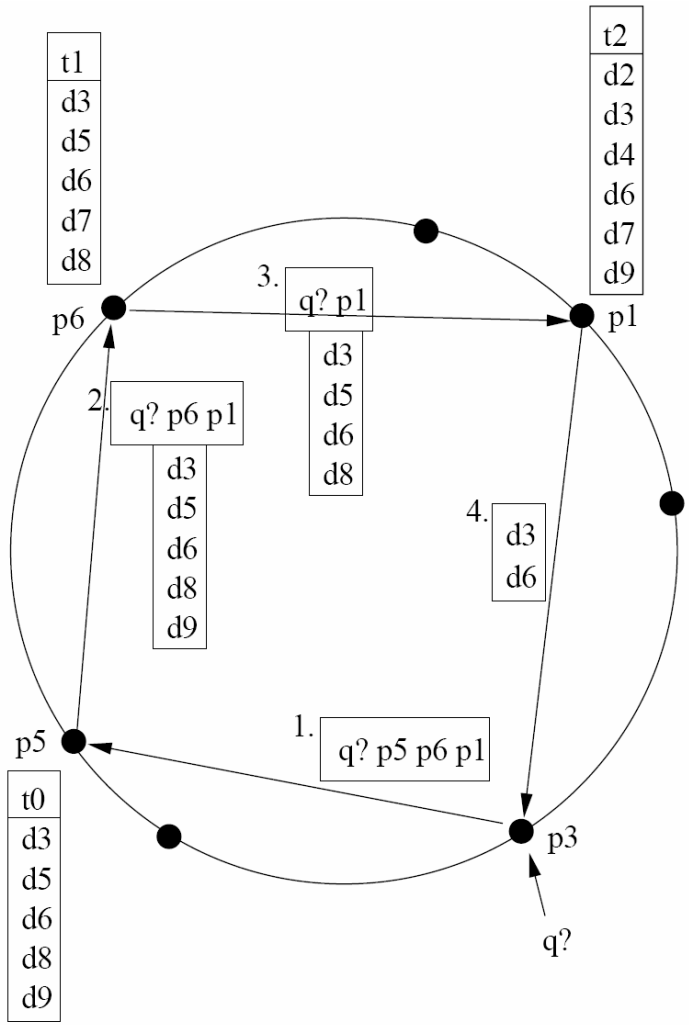


DHT-Query Processing

- Discovery phase
  - To answer a query with *k* terms, we first determine which peers store the inverted lists for those terms.
- Intersection phase
  - Since we already know which peers hold each inverted list, we only need to intersect each inverted lists for different terms to find the query result.

DHT-Query Processing Example

- After the discovery phase, *p*3 knows that the relevant peers are *p*5, *p*6 and *p*1.
- After the Intersection phase, query result is then returned to the searching peer *p*3.



(a) inverted lists

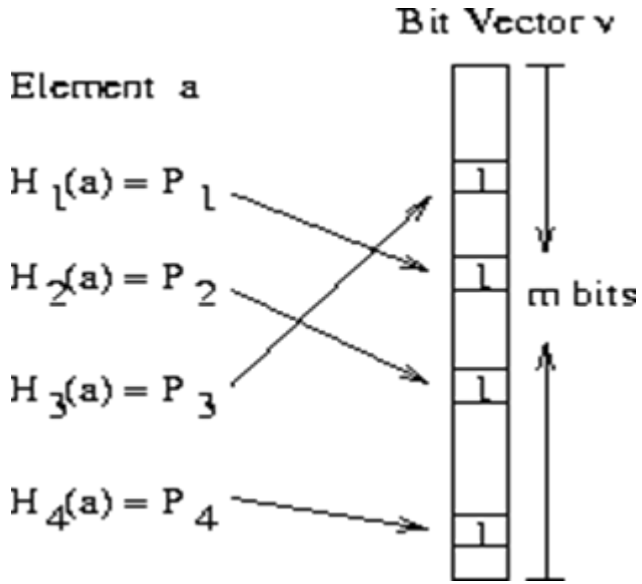
DHT with Bloom Filters

- To reduce the communication cost for an intersection, we can use Bloom filters to summarize the inverted lists.

A Bloom Filter with 4 hash functions. **a** ε **A**

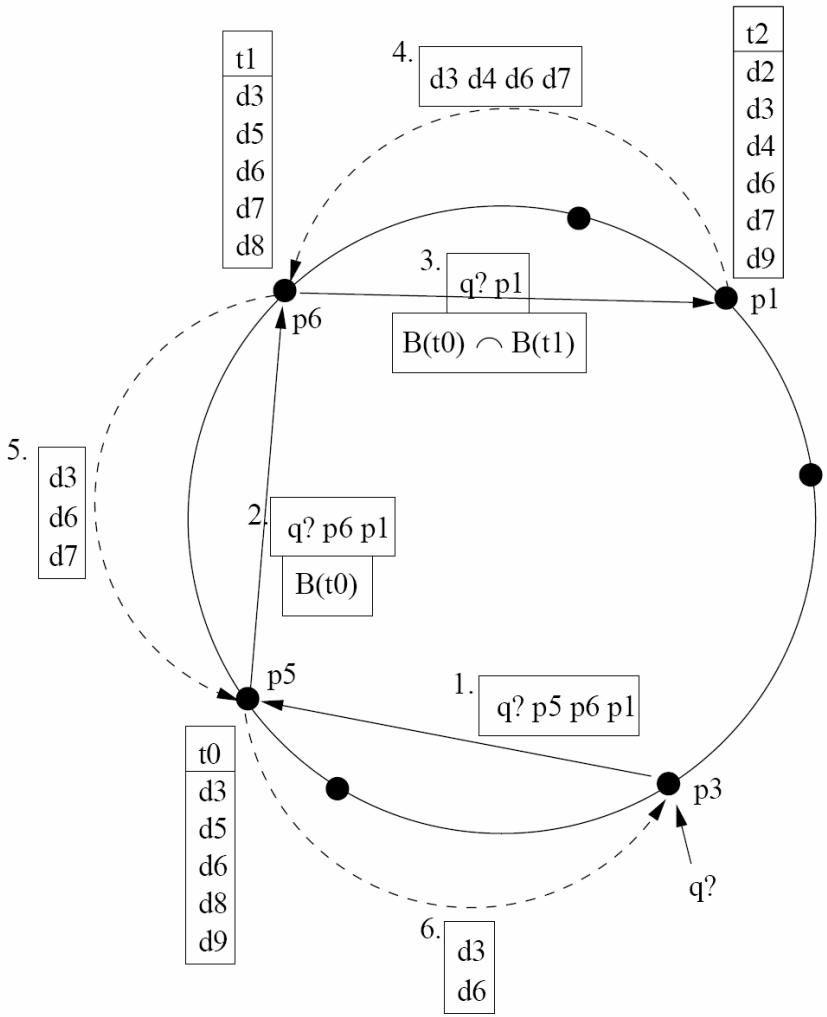
Given a query a, we will check bits at positions *h1*(*a*), *h2*(*a*), ..., *hk*(*a*).

If any of them is 0 then a is not in the set A



DHT with Bloom Filter Example

- $B(t_i)$ indicates a Bloom filter representing the inverted list for term t_i.
- Bloom filters and their intersections are forwarded between peers.



(b) Bloom filters

Bloom Filter with Caching

- Each peer caches the Bloom filters it received during the processing of previous queries.
- Each cached Bloom filter that is used reduces the number of hops needed, with a corresponding decrease in bandwidth used.

Super-peer Networks

- In a super-peer network, each peer is classified as either a “super-peer” or “leaf peer”.
- Each peer constructs a full inverted index over their documents.
- This allows super-peers to process searches over both their own content and the content of their leaf peers.

Super-peer :Document Publishing

- A super-peer also has documents, and constructs an inverted index.
- Leaf peers send a copy of their index to their assigned super-peers during the “publishing” phase.
- However, unlike leaf peers, super-peers handle searches over their own content, and thus do not need to explicitly “publish” their documents.

Super-peer : Query Processing

- When a query is issued from a leaf peer, it is first sent to the super-peer.
- The super-peer both processes the query and forwards it to other super-peer neighbors.
- The bandwidth required for processing queries consists of the bandwidth needed to forward search messages and return results.
- The response time of the query depends on the structure of the connections between super-peers.

Super-peer with TTL

- Consider the time required for the search message starting at super-peer *p* to reach all of the super-peers it will be forwarded to.

- If the maximally distant super-peer (that is, the super-peer the most hops away from *p*) is more than TTL hops away, then the response time is bounded by the TTL; otherwise, it is bounded by the number of hops to the maximally distant super-peer.

- Therefore, the upper bound of the response time is controlled by the user-defined TTL.

Unstructured Networks

- Random walk searches, reduce the bandwidth used for searching, compared to Gnutella's original flooding protocol, by contacting a random subset of the peers.

- Recall that in this technique, publishing is not necessary; each peer stores, indexes and processes searches over its own content.

Random walk : Query Processing

- With a random walk search, each peer that receives a query processes it over its own content, returns any results, and forwards it to *n* random neighbors.

- The walk terminates when a user-specified threshold *T* number of results have been found, a user-specified TTL is reached, or some other stopping condition is satisfied.

Efficient Structure for Random Walk

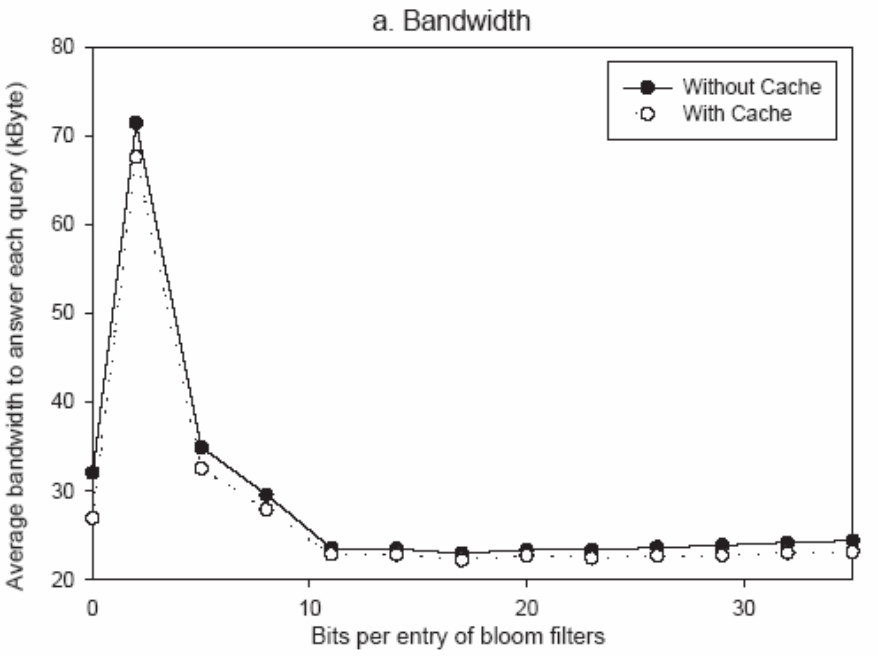
- State-keeping
  - Each peer keeps track of who they forward searches to, and avoid forwarding the same search to the same neighbor repeatedly.
- Square-root topology
  - Each peer tracks the popularity of its content, and adjusts its degree (number of neighbors) so that the degree is proportional to the square root of the popularity of the content.

Performance Comparison

- Experimental Results
  - DHT with Bloom filter and caching
- Document Publishing
- Query Processing
  - Forwarding the query
  - Answering the query

Experimental Results-DHT

- This shows the performance results using Bloom filters and caching.



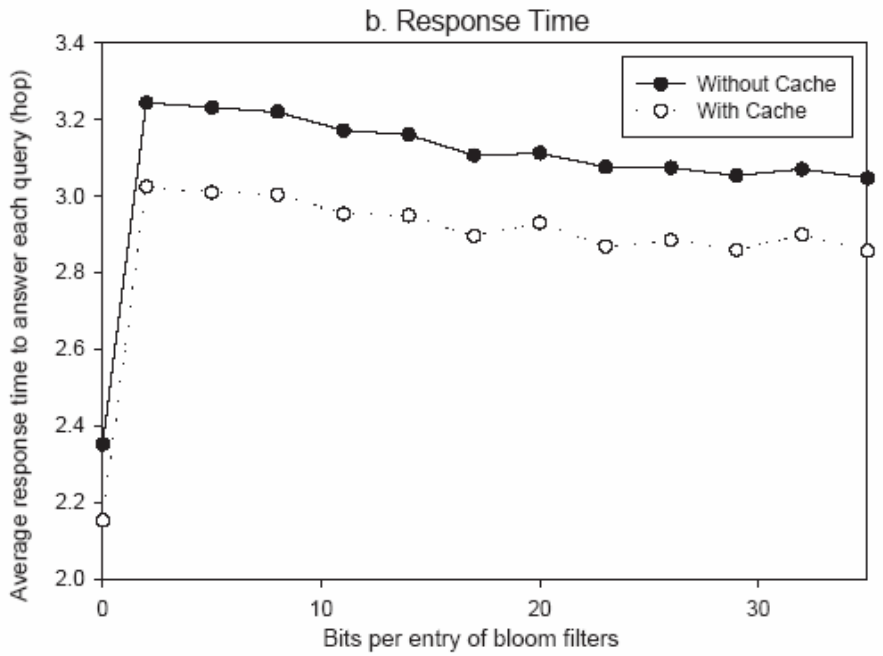


Fig. 2. Performance of answering each query

Document Publishing

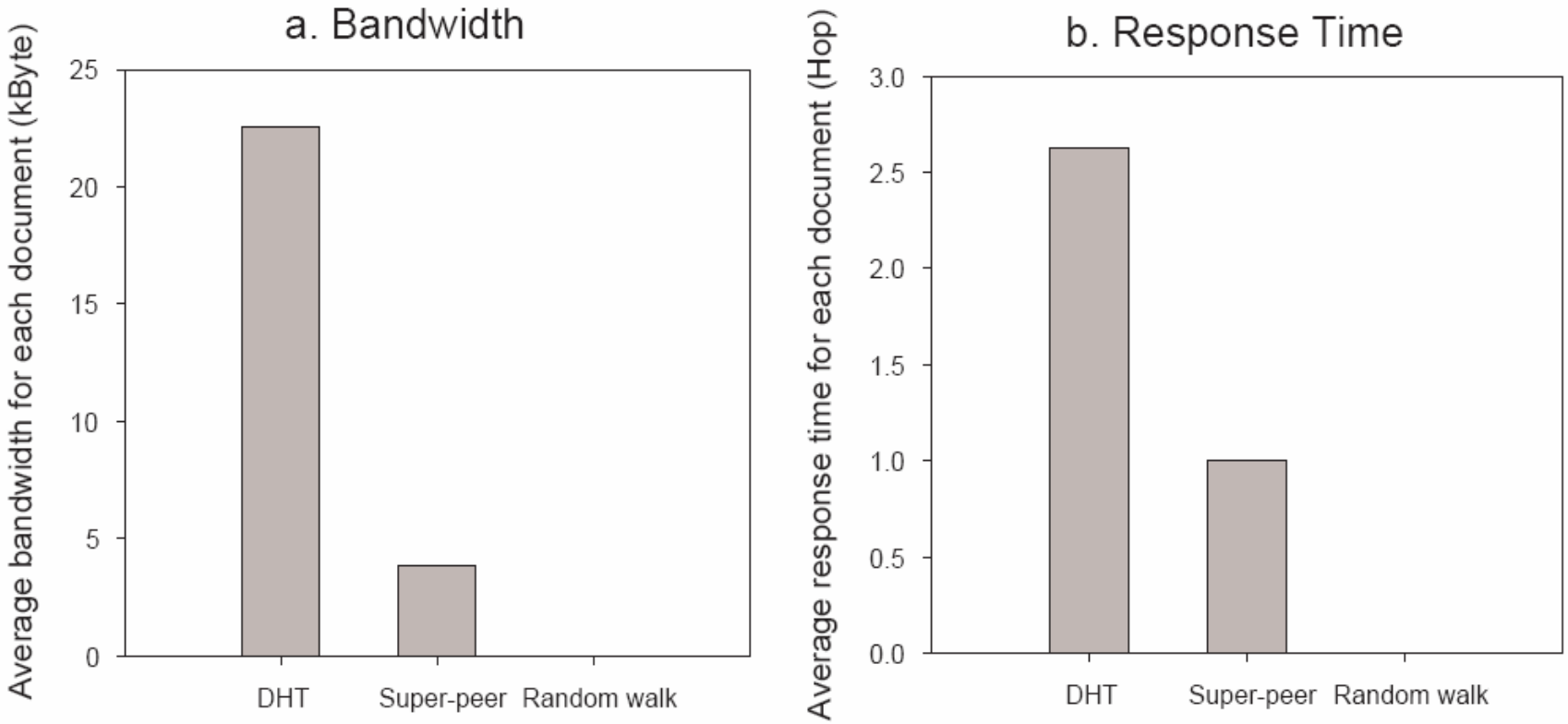


Fig. 6. Comparison of Document Publishing

Query Processing

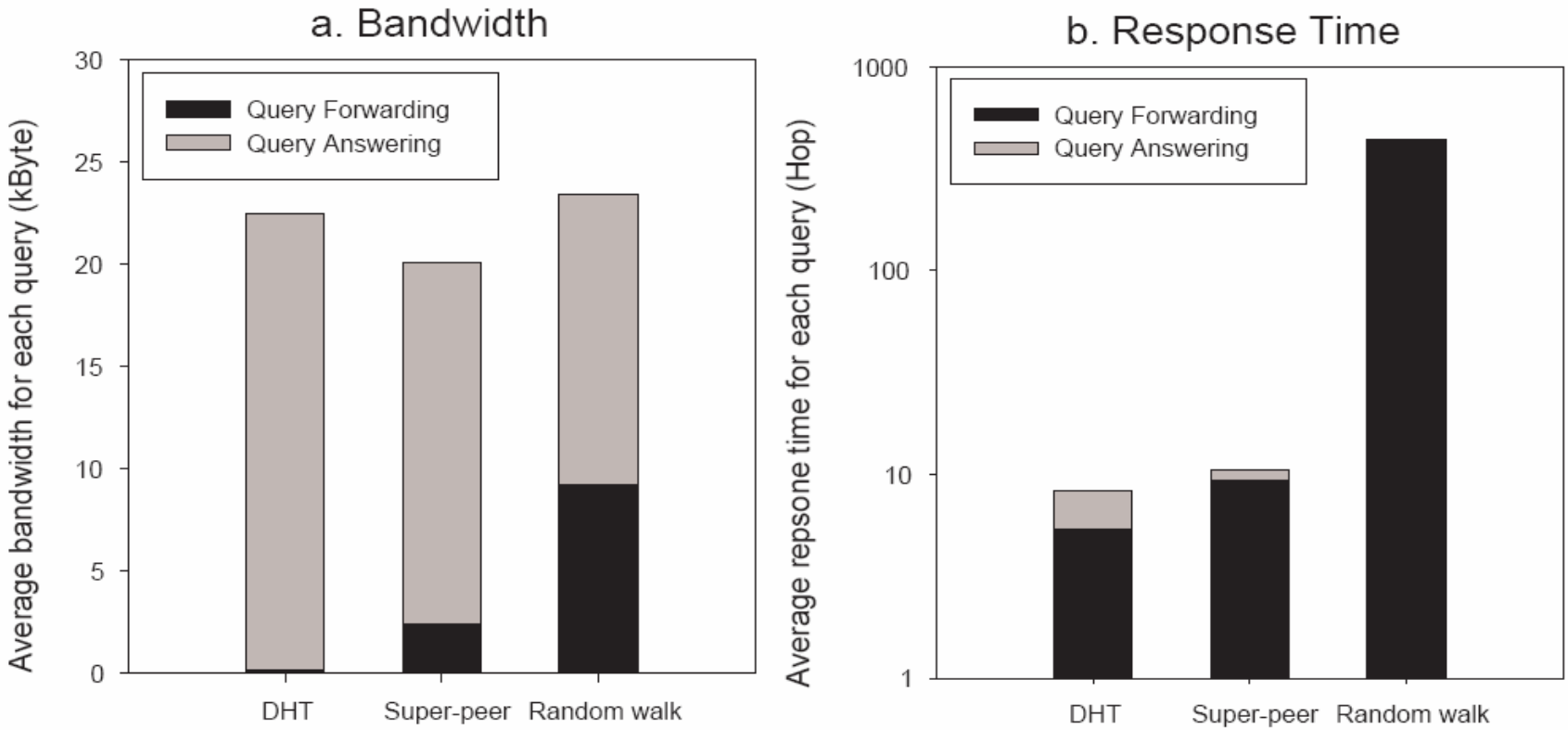


Fig. 7. Comparison of Query Processing

Conclusion

- It has provided a quantitative evaluation and direct comparison of structured and unstructured P2P systems for full text search.
- Using real web documents and user queries, it has examined the performance of the publishing and searching phases of three different techniques.
- This results show that all three techniques use roughly the same bandwidth to process queries.