An Effective P2P Search Scheme to Exploit File Sharing Heterogeneity

From IEEE Transactions on Parallel and Distributed System, February 2007

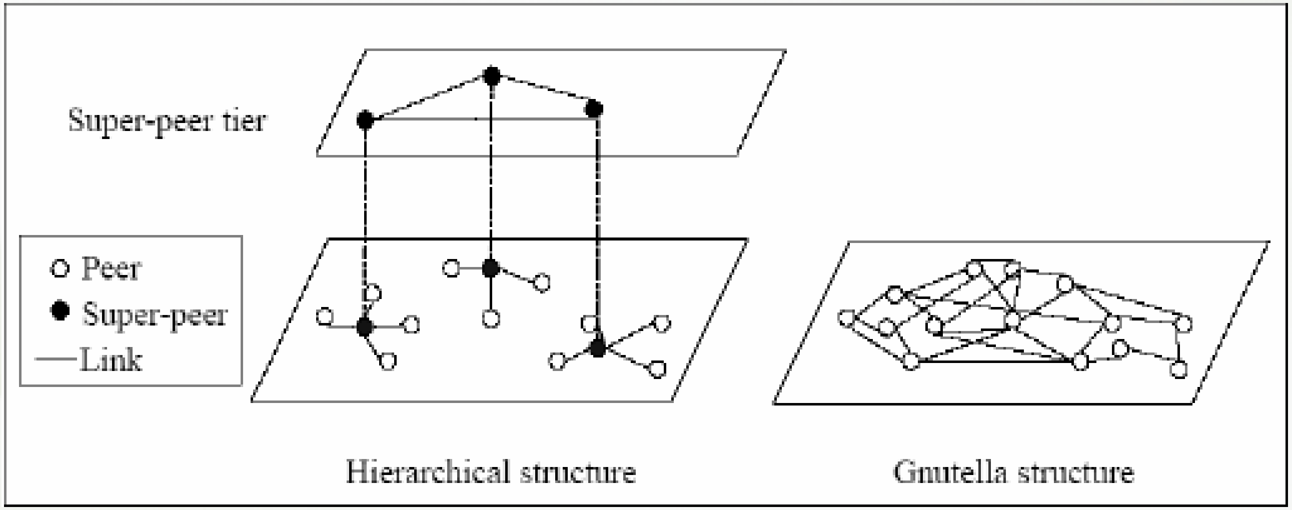
Presented by Ching-Lan Wang
January 25, 2007

Outline

- Introduction
- Heterogeneity of file sharing
  - Response distribution
  - Number of shared files
- DiffSearch algorithm
- Performance evaluation
- Conclusion

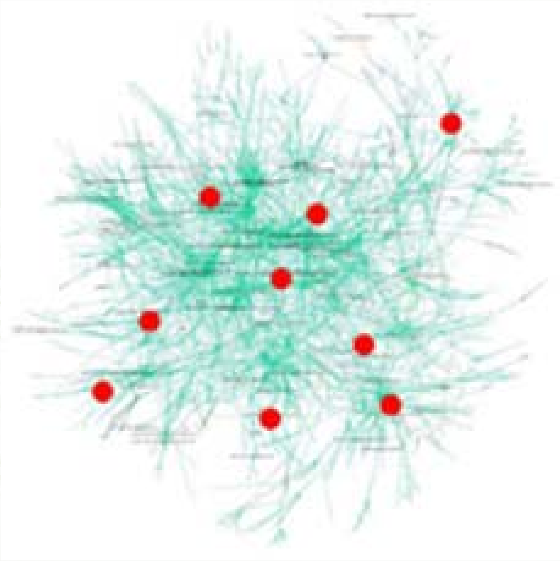
Introduction

- Hierarchical P2P networks (KaZaA)
  - Try to reduce the flooding traffic by limiting the search scope within a small area of supernodes
  - Current hierarchical designs select the ultrapeers by emphasizing their computing capabilities such as bandwidth, CPU power, and memory spaces.



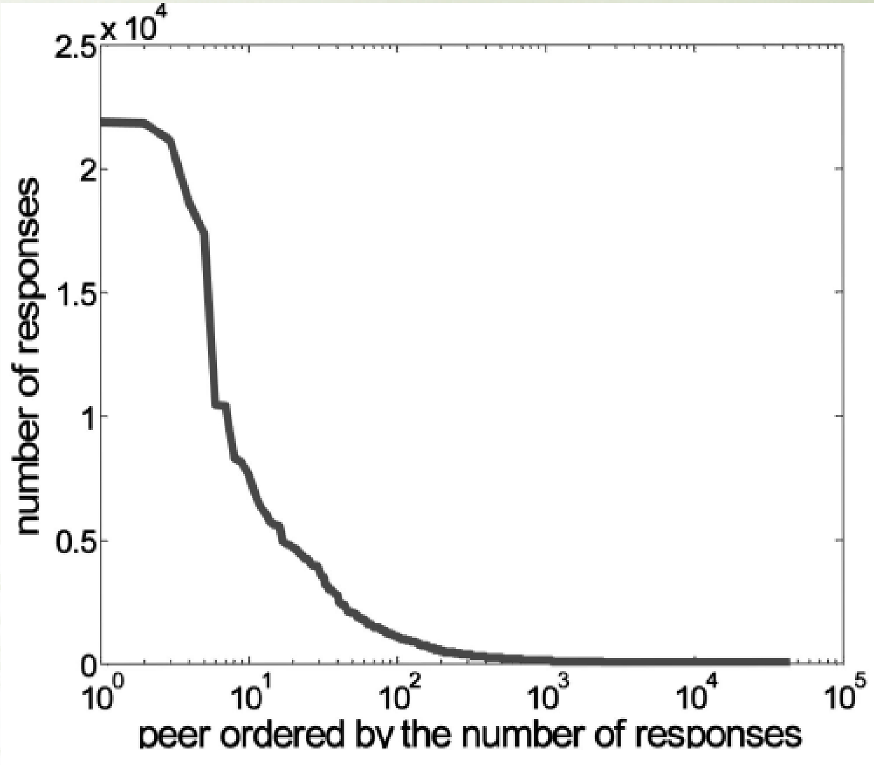
Heterogeneity of file sharing

- Seven percent of peers in the Gnutella network share more files than all of those other peers can offer and 47 percent of queries are responded to by the top 1 percent of peers.
- Some peers are more willing to share files than others.



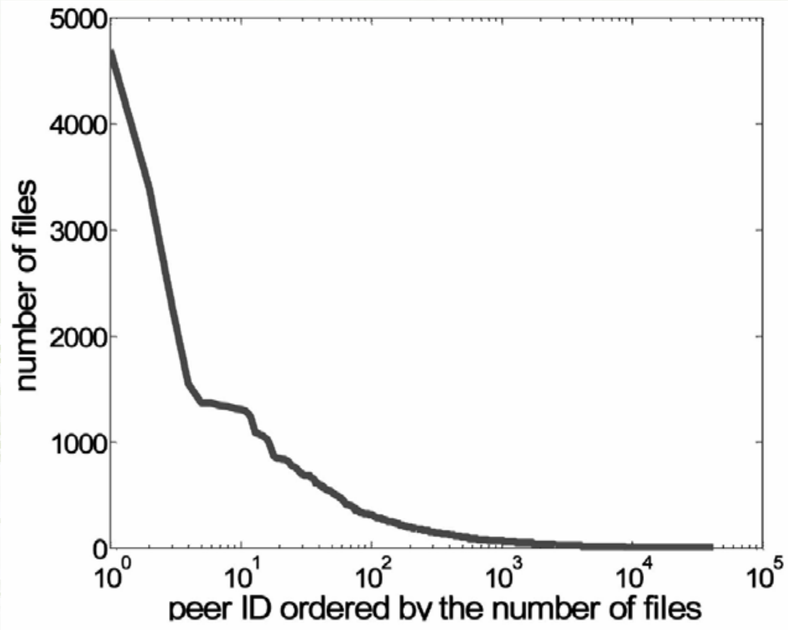
Heterogeneity of file sharing - Response distribution

- Response distribution
  - The top 1 percent of peers answers the main portion of queries.
  - If we could route all the queries to those top peers first, close to 90 percent of query traffic would have been saved.



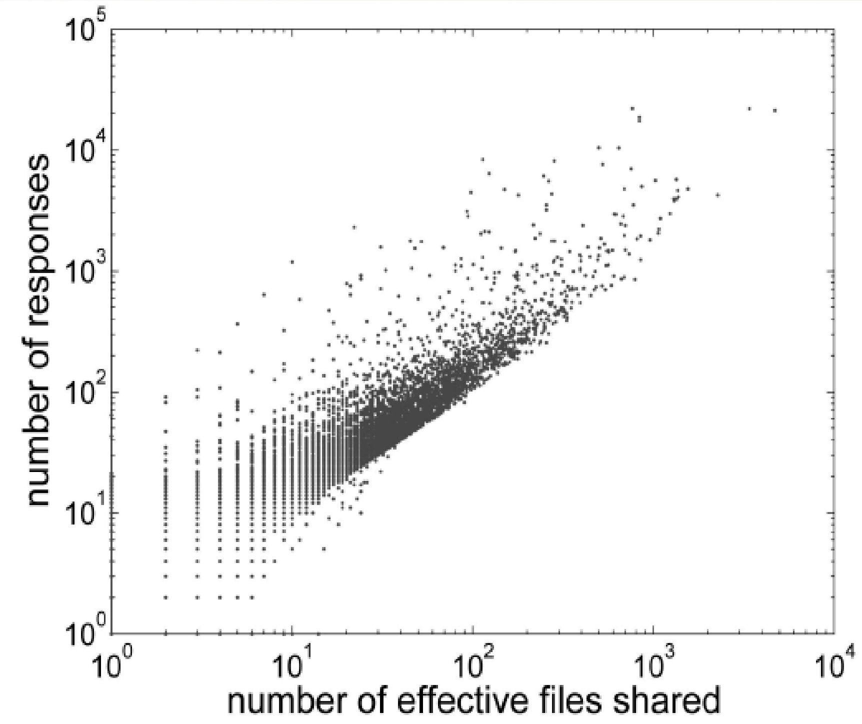
Heterogeneity of file sharing - Number of shared files

- Number of shared files
  - Very few peers share a large number of files.
  - Some useless files make no contribution to the query answering, i.e., some files are never used to answer the queries.



Effective files

- To distinguish those files from useless files, we define the files which have been used to answer the queries as *Effective Files*.
- The peers sharing more effective files have a greater tendency to answer queries.

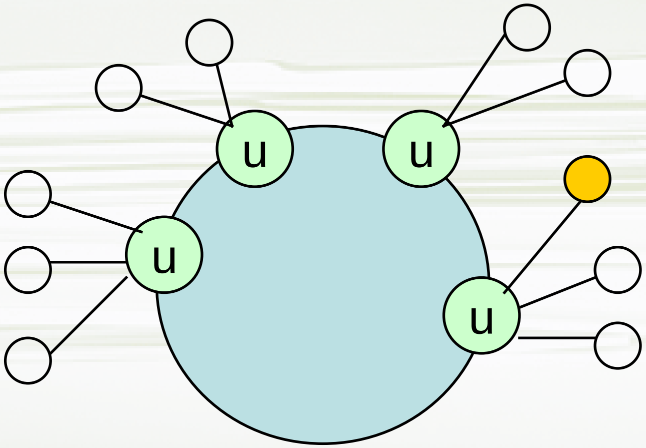


DiffSearch algorithm

- Overview of DiffSearch algorithm
- Selecting ultrapeers
- Finding ultrapeers
- Evolve an ultrapeer overlay
- Maintaining the hierarchical structure
- Fully distributed operations
- Load balance-Caching & Redirecting algorithm

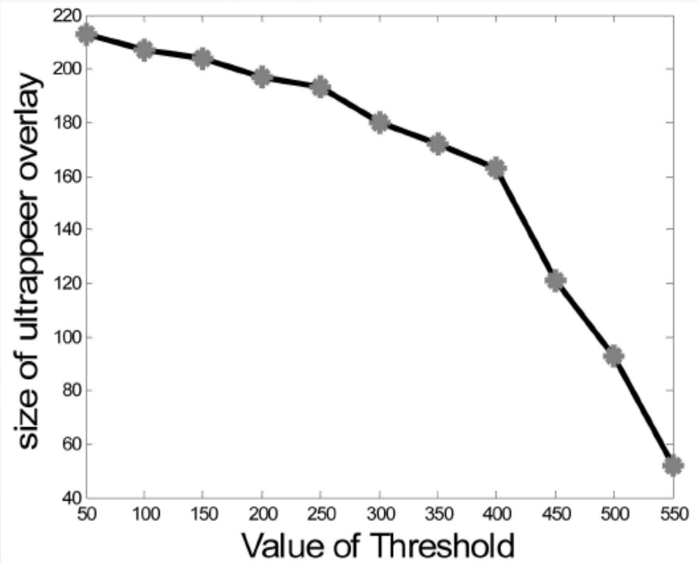
Overview of DiffSearch algorithm

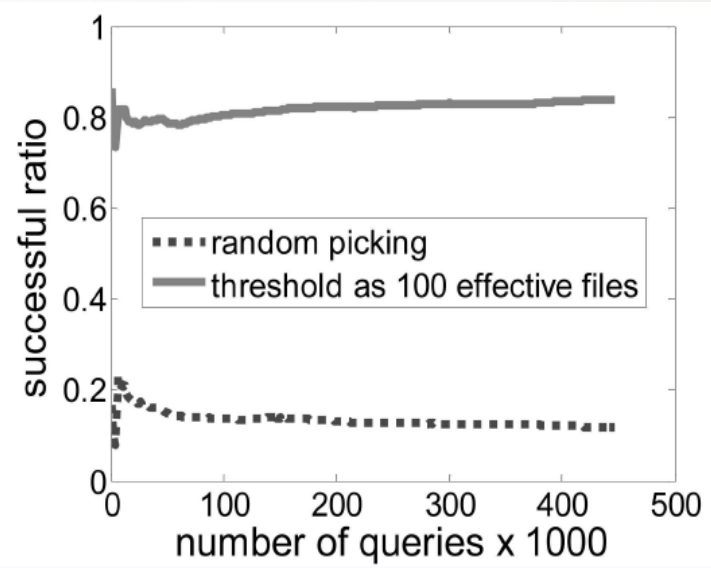
- In the DiffSearch algorithm, a query consists of two round searches.
  - In the first round search, the query is only sent to the ultrapeer overlay.
  - If the first round search fails in the ultrapeer overlay, the second round search will be evoked to query the entire network.



Selecting ultrapeers

- The number of effective files shared by a peer is a good criterion to determine if the peer should be selected as an ultrapeer.
- By setting a threshold of 100 effective files, the top 2 percent of peers are selected from 10,000 peers to form the ultrapeer overlay.





Finding ultrapeers

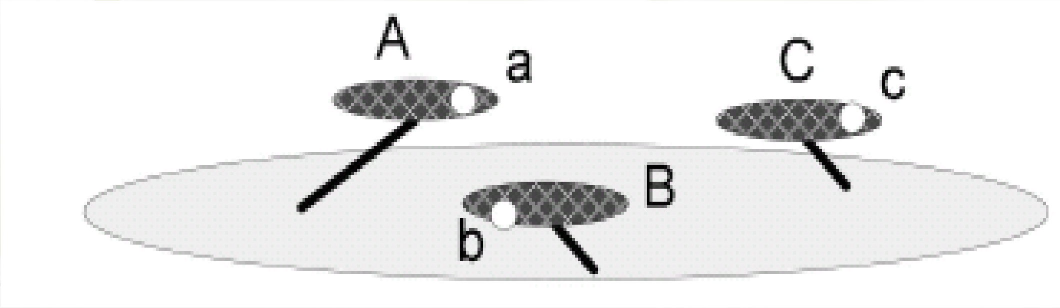
- Passive approach
- Active approach
- The topology creation message hitchhiked on the query/response messages.
  - One bit of data is appended to the reply message to indicate if the respondent is an ultrapeer.
  - All of the replies received by isolated peers will be checked and the IP addresses will be extracted from the message sent from ultrapeers.

Evolve an ultrapeer overlay

- To guarantee that each peer in the ultrapeer overlay can be reached by the first round search in DiffSearch, all the peers in the ultrapeer overlay should form a connected topology.
- The basic approach is to detect all the separated clusters consisting of ultrapeers and connect them with each other.

Evolve an ultrapeer overlay

- If peer a fails to search keyword k in cluster A during the first round search of DiffSearch
  - The keyword k is not shared by ultrapeers.
  - The file k is shared by ultrapeers, but they are located in separated clusters B or C.
- For any case, peer a will initiate the second round search to the whole network.

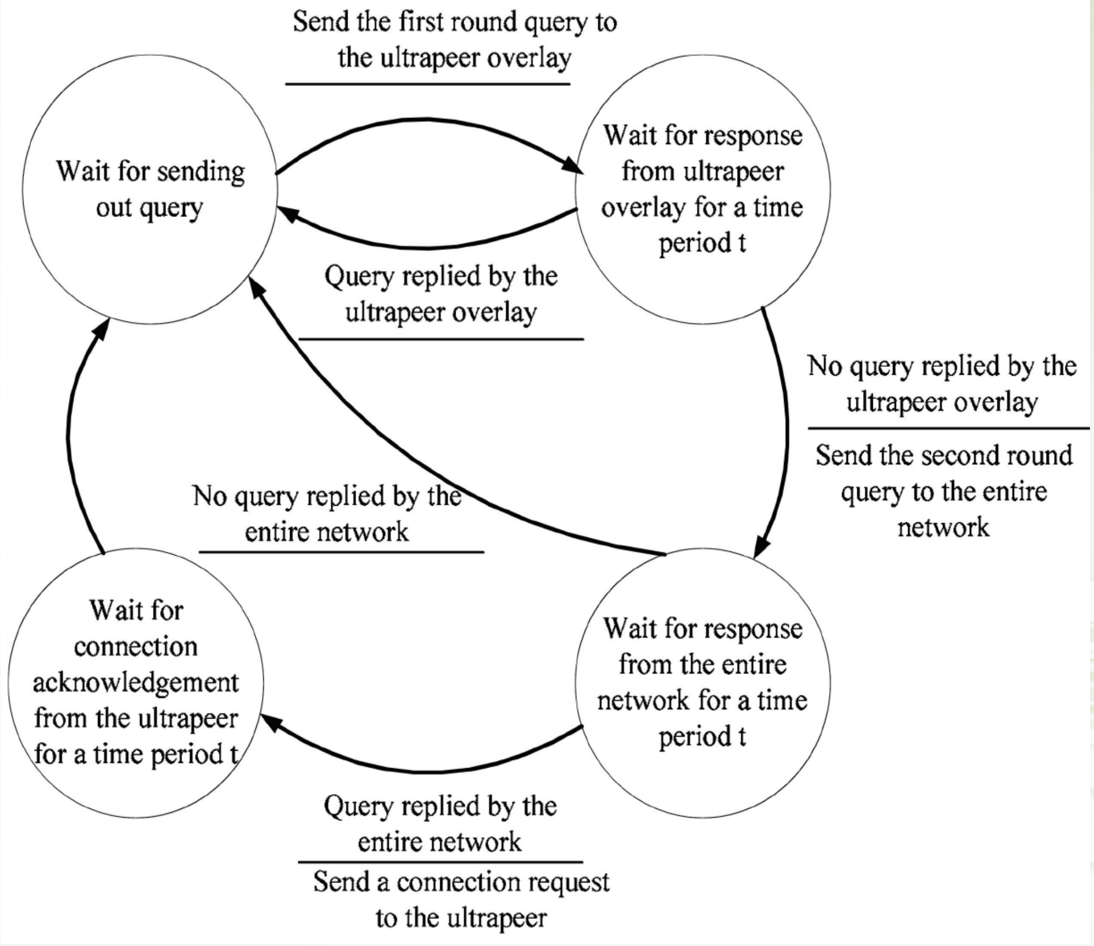


Maintaining the hierarchical structure

- To hitchhike the overlay construction to the search messages, three bits need to be appended to the original query and reply messages.
  - One bit is used in the query message to show whether the query is in the first round or second round.
  - Two bits are used in the reply message to show whether the reply is from an ultrapeer and to which round search the reply is responding.

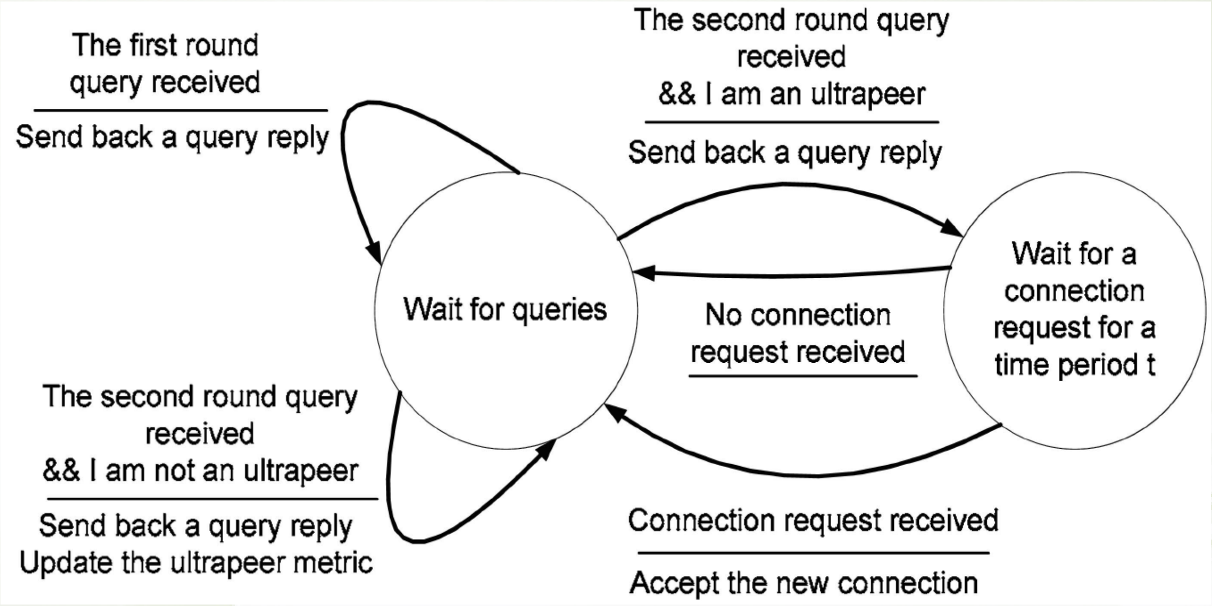
Fully distributed operations

- Two round query operation of an individual peer.



Fully distributed operations

- Query reply operation of an individual peer.



Load balance-Caching & Redirecting algorithm

- Each ultrapeer overhears query reply messages and caches the IP addresses of other ultrapeers which are less loaded than itself.
- When a fully loaded ultrapeer cannot accommodate more incoming connection requests, it will redirect the requests to other ultrapeers in the caching list.

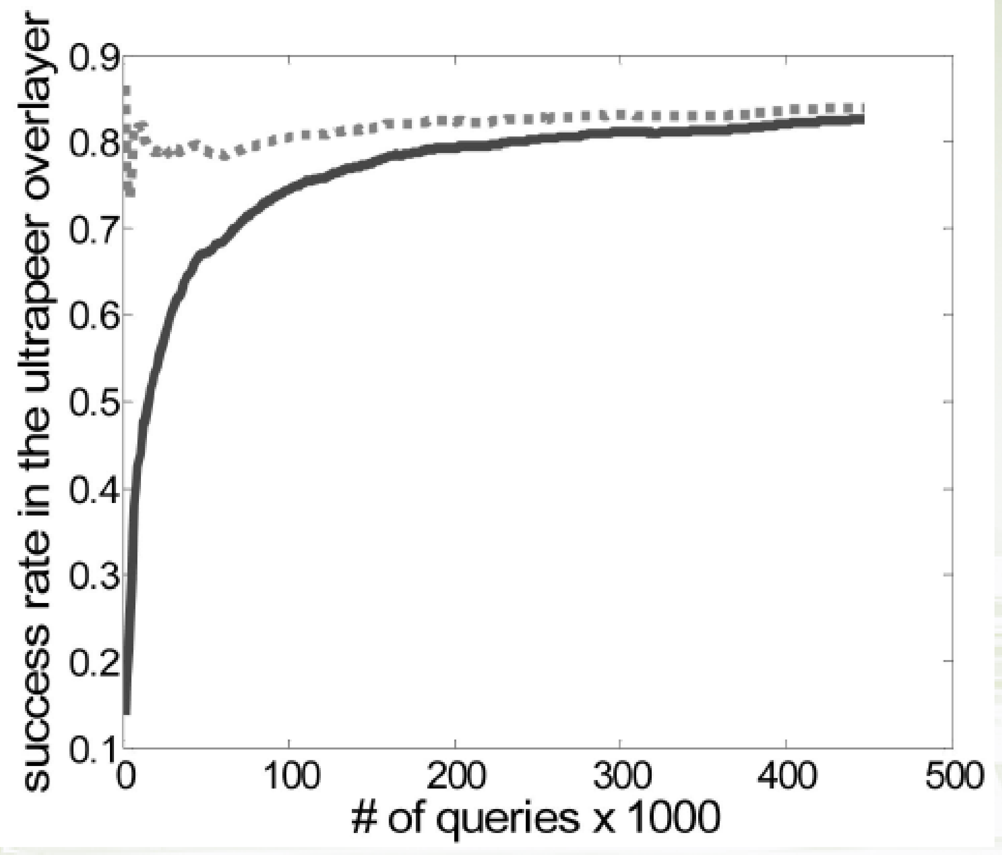
Algorithm 1: Caching and redirecting load balance algorithm

```
While true
  Wait for message m;
  If m is a query reply
    If load_of_responder < local_load
      Cache the responder's IP address in the list L;
    End;
  Else if m is a connection request
    If local_load < max_load
      Accept the new connection;
      local_load = local_load + 1;
    Else
      Forward the request connection to other
      ultrapeers in the cache list L;
    End;
  End;
End;
```

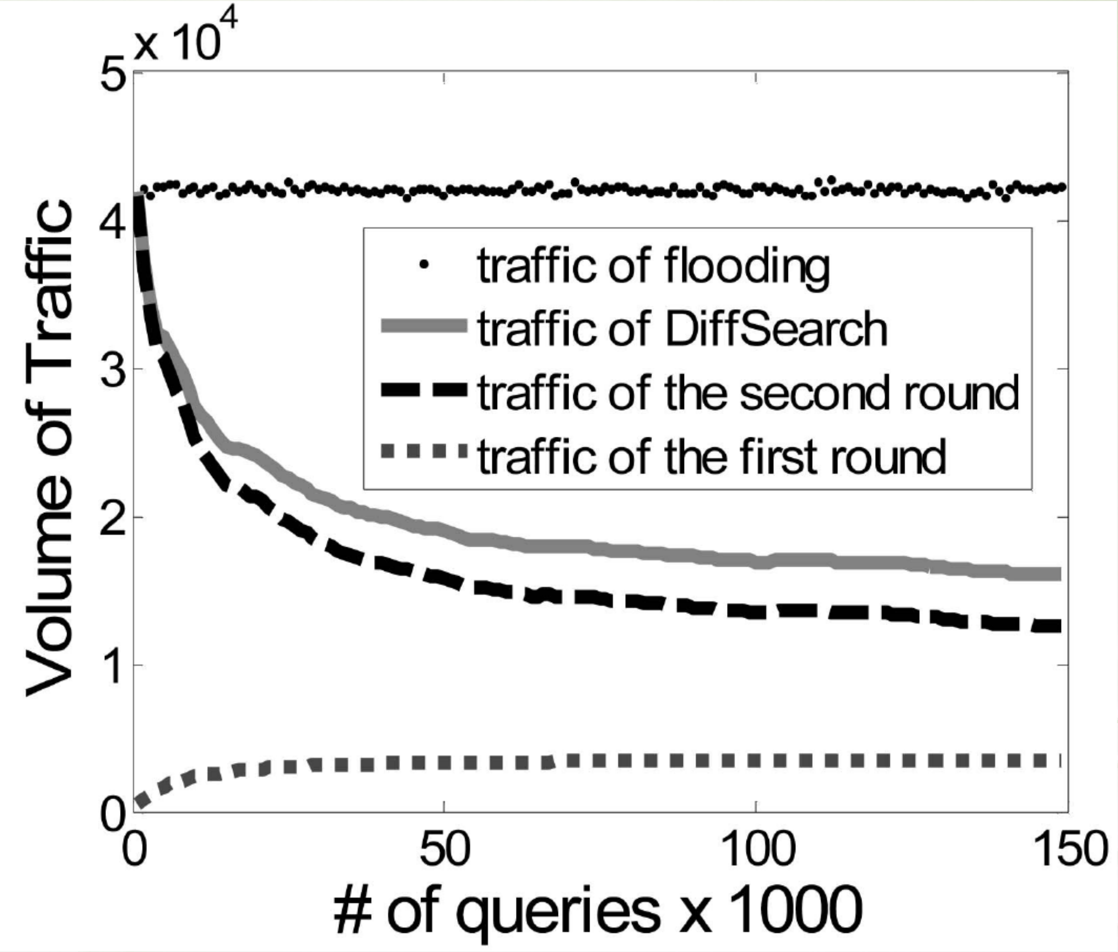
Performance evaluation

- Convergence Speed
- Performance improvement
  - Average Network Traffic
  - Average Response Time
  - Query success rate
- Load balance of ultrapeers

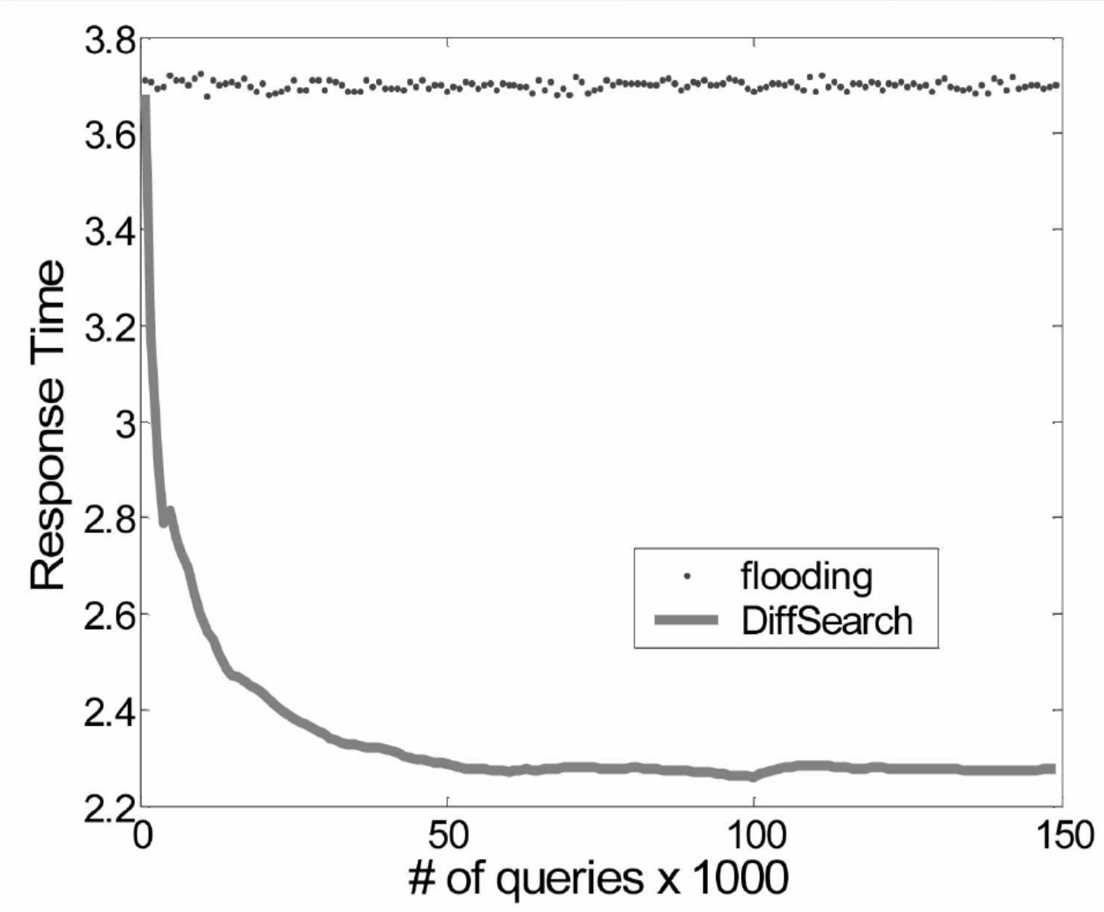
Convergence Speed



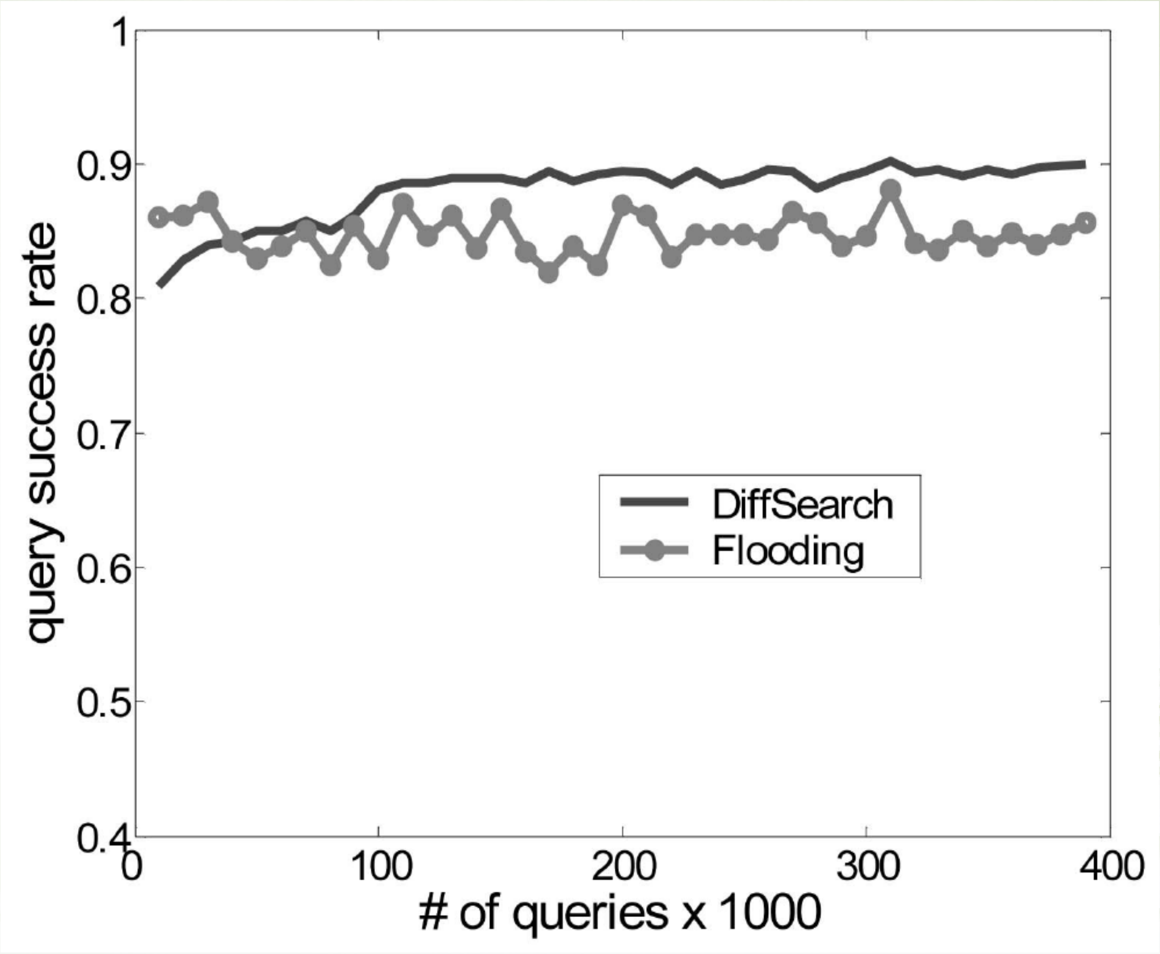
Average Network Traffic



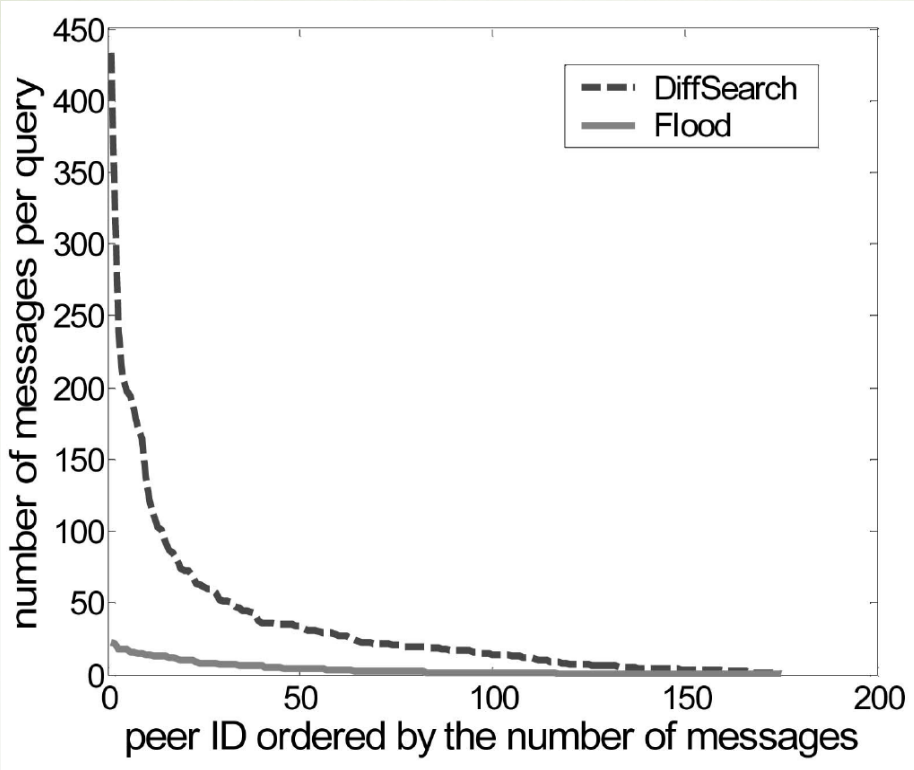
Average Response Time

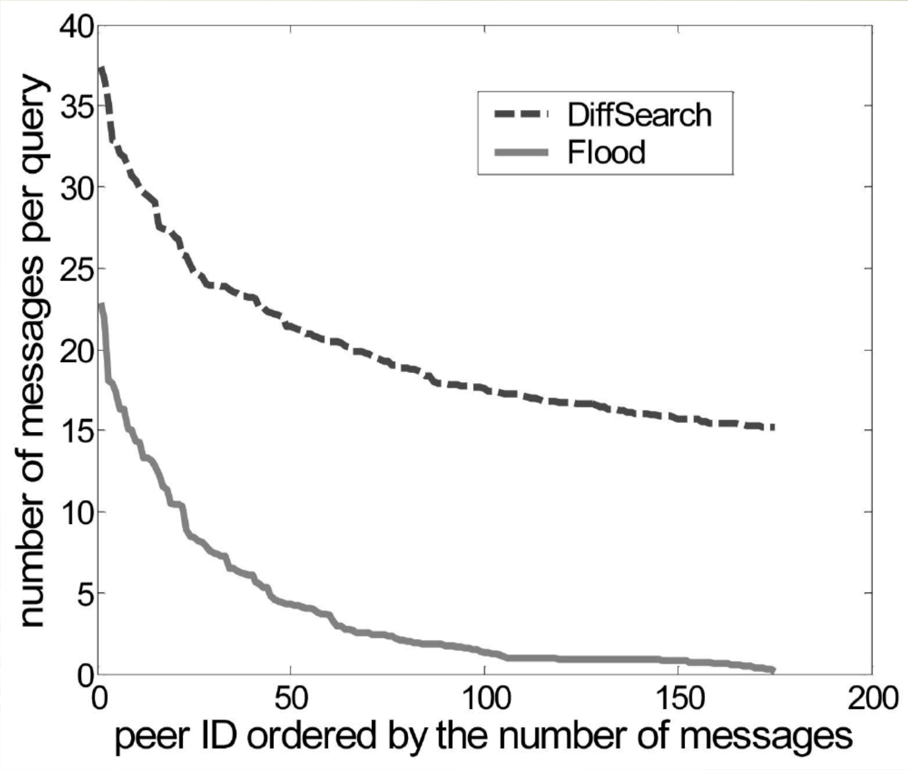


Query success rate



Load balance of ultrapeers





Conclusion

- In this paper, we propose the DiffSearch algorithm, a fully distributed approach which can evolve a two-tier hierarchical structure P2P network.

- By hitchhiking the topology operations to query/ reply messages and prompting content-rich peers to the ultrapeer overlay, the DiffSearch algorithm can achieve significant performance improvement with a little overhead on topology maintenance and index operations.