Topics in Database Systems: Data Management in Peer-to-Peer Systems

March 29, 2005

Outline

- More on Search Strategies in Unstructured p2p
- Replication
  - general
  - review of structured
  - techniques for unstructured

Notes

- No class on April 5
- Next assignment (tomorrow in the web page)
  - Present one paper (3 papers, 1 per group)
    - MAX 35' each
      - Topology
      - Join/Search
      - Evaluation
      - Other Issues

      the presentation should also include
      - a short discussion (3-5 slides) of what replication strategies you think could be applied in the system you will be presenting

Topics in Database Systems: Data Management in Peer-to-Peer Systems

D. Tsoumakos and N. Roussopoulos, "A Comparison of Peer-to-Peer Search Methods", WebDB03

Overview

- Centralized

Constantly-updated directory hosted at central locations (do not scale well, updates, single points of failure)

- Decentralized but structured

The overlay topology is highly controlled and files (or metadata/index) are not placed at random nodes but at specified locations

"loosely" vs "highly-structured" DHT

- Decentralized and Unstructured

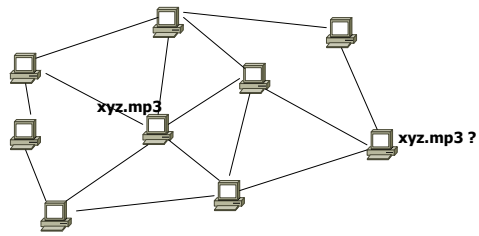
peers connect in an ad-hoc fashion

the location of document/metadata is not controlled by the system

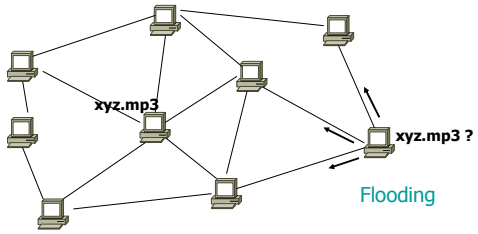
No guaranteed for the success of a search

No bounds on search time

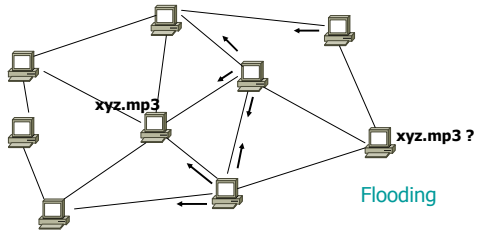
Flooding on Overlays



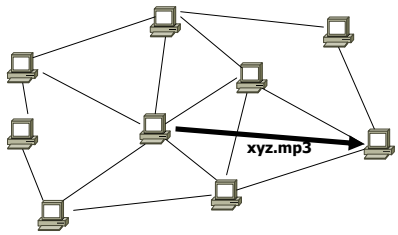
Flooding on Overlays



Flooding on Overlays



Flooding on Overlays



Search in Unstructured P2P

BFS vs DFS

BFS better response time, larger number of nodes (message overhead per node and overall)

Note: search in BFS continues (if TTL is not reached), even if the object has been located on a different path

Recursive vs Iterative

During search, whether the node issuing the query direct contacts others, or recursively.

Does the result follows the same path?

Search in Unstructured P2P

Two general types of search in unstructured p2p:

Blind: try to propagate the query to a sufficient number of nodes (example Gnutella)

Informed: utilize information about document locations (example Routing Indexes)

Informed search increases the cost of join for an improved search cost

Blind Search Methods

Gnutella:

Use flooding (BFS) to contact all accessible nodes within the TTL value

Huge overhead to a large number of peers +

Overall network traffic

Hard to find unpopular items

Up to 60% bandwidth consumption of the total Internet traffic

Modified-BFS:

Choose only a *ratio* of the neighbors (some random subset)

Blind Search Methods

Iterative Deepening:

Start BFS with a small TTL and repeat the BFS at increasing depths if the first BFS fails

Works well when there is some stop condition and a "small" flood will satisfy the query

Else even bigger loads than standard flooding

(more later …)

Blind Search Methods

Random Walks:

The node that poses the query sends out *k* query messages to an equal number of randomly chosen neighbors

Each step follows each own path at each step randomly choosing one neighbor to forward it

Each path – a walker

Two methods to terminate each walker:

- TTL-based or
- checking method (the walkers periodically check with the query source if the stop condition has been met)

It reduces the number of messages to k x TTL in the worst case

Some kind of local load-balancing

Blind Search Methods

Random Walks:

In addition, the protocol bias its walks towards *high-degree nodes*

Blind Search Methods

Using Super-nodes:

Super (or ultra) peers are connected to each other

Each super-peer is also connected with a number of lead nodes

Routing among the super-peers

The super-peers then contact their leaf nodes

Blind Search Methods

Using Super-nodes:

Gnutella2

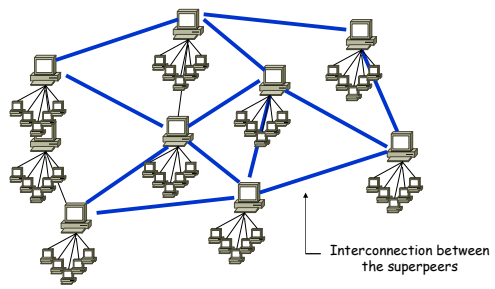
When a super-peer (or hub) receives a query from a leaf, it forwards it to its relevant leaves and to neighboring super-peers

The hubs process the query locally and forward it to their relevant leaves

Neighboring super-peers regularly exchange local repository tables to filter out traffic between them

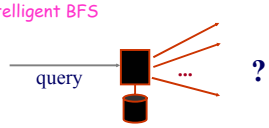
Blind Search Methods

Ultrapeers can be installed (KaZaA) or self-promoted (Gnutella)



Informed Search Methods

Intelligent BFS



Nodes store simple statistics on its neighbors:

(query, NeigborID) tuples for recently answered requests from or through their neighbors

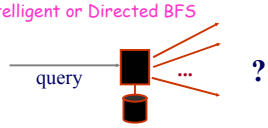
so they can rank them

For each query, a node finds similar ones and selects a direction

How?

Informed Search Methods

Intelligent or Directed BFS



- Heuristics for Selecting Direction
  - **>RES**: Returned most results for previous queries
  - **<TIME**: Shortest satisfaction time
  - **<HOPS**: Min hops for results
  - **>MSG**: Forwarded the largest number of messages (all types), suggests that the neighbor is stable
  - **<QLEN**: Shortest queue
  - **<LAT**: Shortest latency
  - **>DEG**: Highest degree

Informed Search Methods

Intelligent or Directed BFS

- No negative feedback
- Depends on the assumption that nodes specialize in certain documents

Informed Search Methods

APS

Again, each node keeps a local index with on entry for each object it has requested per neighbor – this reflects the relative probability of the node to be chosen to forward the query

k independent walkers and probabilistic forwarding

Each node forwards the query to one of its neighbor based on the local index

If a walker, succeeds the probability is increased, else is decreased –

How?

After a walker miss (optimistic update) or after a hit (pessimistic update)

Informed Search Methods

Local Index

Each node indexes all files stored at all nodes within a certain radius *r* and can answer queries on behalf of them

Search process at steps of r

Flood inside each r with TTL = r

Increased cost for join/leave