CS345: Unstructured P2P Networks
Tyson Condie

Problem
- Potentially millions of users
  - Aggregate processing and bandwidth
  - Wide range of heterogeneity
  - Large transient user population
- Existing search solutions
  - Flooding-based solutions limit capacity
  - Not scalable
  - Distributed Hash Tables (DHTs) not necessarily appropriate for all queries
  - Lots of overhead in most cases

Outline
- Gnutella (quick recap)
- Super-Peer Network
  - [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker]
- Evaluation of results
- GIA: Scalable Gnutella
  - [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker]
- Results: Simulations & Experiments
- Conclusion

Query in Gnutella
- Query source sends query to all neighbors.
- Incoming queries
  - Check for match (possibly respond)
  - Forward query (reverse-path broadcast).
- Query Response
  1. Forward back along the reverse path.
     - Uses more bandwidth.
  2. Send directly to query source
     - No anonymity.
     - Overload query source with connections.
- Use time-to-live to limit the reach (scope) of the query.

Gnutella recap
- Bootstrap method: Contact pong server to obtain a list of (possibly) active peers
- Distributed search and download
- Unstructured: ad-hoc topology
  - Peers connect to random nodes
- Random search
  - Flood queries across network
- Query can take any appropriate form
  - Exact match, range, keywords (with regular expressions)

Gnutella Problems
- Scaling problems
  - As network grows, search overhead increases
- Does not return all possible results of a search query
  - The reach of a query

TTL = 2 & P1 has madonna who has madonna

...
Outline

- Gnutella (quick recap)
- Super-Peer Network
  - [B. Yang, H. Garcia-Molina]
- Evaluation of results
- GIA: Scalable Gnutella
  - [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker]
- Results: Simulations & Experiments
- Conclusion

Super-Peer Network

- Advantages
  - Client’s query can reach many more nodes
  - Reduced traffic overhead
  - Clients shielded from query processing and traffic
    - Weak peers can be made into clients
- Disadvantages
  - Does not guarantee all files queried for will be found
    (still need DHT's for this)
  - Super-peer becomes a single point of failure for its
    cluster (and a potential bottleneck)

Designing a Super-Peer Network

- Bootstrap method: Pong server provides (random) super-peer IP address
- Node Model
  - Super-peer is a node that acts as a centralized server to a subset of
    clients. Super-peer + clients = cluster
  - Cross between hybrid systems (Napster) and pure P2P systems
    (Gnutella)
- Query Model
  - Client sends query to its representative super-peer
  - Super-peer checks its index and forwards query to other super-peers
  - Topology (among super-peers)
    - Power Law, random, strongly connected

Super-peer redundancy

- A super-peer is k-redundant if there are k nodes sharing
  the super-peer load
  - Client(s) index replicated at all k nodes
- Query Model
  - Client sends its query to one of the k nodes
  - Load balance: send to each node in a round-robin fashion
- Advantages (over single super-peer)
  - Increased reliability
  - Reduces aggregate load
- Disadvantages
  - Client join method k times greater than before (client must send
    meta-data to all k nodes)
  - Number of open connections among super-peers increases by
    k^2

Outline

- Gnutella (quick recap)
- Super-Peer Network
  - [B. Yang, H. Garcia-Molina]
- Evaluation of results
- GIA: Scalable Gnutella
  - [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker]
- Results: Simulations & Experiments
- Conclusion

Evaluation Model

- Two types of metrics
  1. Load: in/out bandwidth and processing power
  2. Quality of results: number of results returned per query and quality of results (subjective
     and application specific)
- Performance Evaluation
  - Compare different configurations of systems
    (e.g. power law vs. strongly connected)
Configuration steps

1. Generating and instance
   - Topology, cluster size, bootstrap
2. Calculating expected cost of actions
   - Expected number of messages per query (including possible responses)
     - Approximate query propagation using a breadth-first search (depth limited by the TTL)
3. Calculating load from actions
4. Repeated Trials
   - Repeat analysis of expected aggregate load (95% confidence intervals)

Results

1. Increasing cluster size decreases aggregate load, but increases individual load
2. Super-peer redundancy is good
3. Maximize out degree of super-peers
4. Minimize TTL

1. Increase cluster size decreases aggregate load

   - The fewer super-peers in a system, the less communication overhead. The cost is that each super-peer will maintain a larger cluster index.
   - Extreme:
     - single super-peer, cluster size is 10000 (Napster)
     - 10000 super-peers, cluster size is 1 (Gnutella)
   - Large cluster size increases individual load

Exception 1: Incoming Bandwidth

   - Most incoming bandwidth load for a super-peer is forwarding queries and query results
     - Cluster A consists of a fraction \( f \) of all nodes in the network \( \rightarrow \) super-peer \( A \) submits \( \sim f \) of all queries. Moreover, super-peer \( A \) has \( \sim f \) of all responses \( \sim (1-f) \) of all results sent from other clusters to super-peer \( A \)
     - Expected incoming results proportional to \( f^2 (1-f) \). Maximum at \( f = 1/2 \)

Exception 2: processing load for large out degree

   - Super-peer processing load increases when topology = SCC and cluster size is small
     - OS overloaded with file descriptors (e.g. legacy code that does a linear search)
     - Connection overhead << query processing overhead

2. Super-peer redundancy is good

   - Decreases individual super-peer load significantly
     - Query traffic spread out among the \( k \) super-peer nodes
   - Has no significant affect on aggregate bandwidth
   - Savings in cost comes mostly from the decrease in total per-message overhead of sending results
3. Maximize out degree of super-peers
   • Increasing out degree can reduce individual load
     – Improves performance because it reduces the expected path length (EPL)
     – EPL is a rough measure of the average response time of a query (since each hop takes time). Shorter EPL => lower response time
   • Peers with a large out degree receive more query results (query reach)
     – If only a few super-peers increase their out degree then those few nodes may get swamp. It is important that increasing out degree be a uniform decision!!!

4. Minimize TTL
   • For a given out degree and topology, reach is a function of TTL
   • Select the minimum TTL for the desired reach
     – If TTL=x allows all nodes to be reached, TTL=x+1 will have the same reach
   • Correct global TTL can be obtained by predicting the EPL for the desired reach and avg. out degree

Local Decisions
1. A super-peer should always accept new clients (unless it is overloaded)
2. A super-peer should increase its out degree
   • As long as its cluster is not growing and it has enough resources to spare
3. A super-peer should decrease its TTL
   • As long as it does not affect its reach

Outline
• Gnutella (quick recap)
• Super-Peer Network
  – [B. Yang, H. Garcia-Molina]
• Evaluation of results
  • GIA: Scalable Gnutella
    – [Y. Chawathe, S. Ratnasamy, L. Brebels, N. Lanham, S. Shanker]
• Results: Simulations & Experiments
• Conclusion

GIA Design
• Make high-capacity nodes easily reachable
  – Dynamic topology adaptation
• High-capacity nodes have more answers
  – One-hop replication (Anonymity sacrificed?)
• Search efficiently
  – Biased random walks
• Prevent overloaded nodes
  – Active flow control

Dynamic Topology Adaptation
• High-capacity nodes have high degree (i.e., more neighbors)
• Per-node level of satisfaction, S:
  – 0 ⇒ no neighbors, 1 ⇒ enough neighbors
  – Function of:
    • Node’s capacity
    • Neighbors’ capacities
    • Neighbors’ degrees
    • Their age
  – When S << 1, look for neighbors aggressively
Active Flow Control

• Accept queries based on capacity
  – Actively allocate “tokens” to neighbors
  – Send query if we have received token
  – Susceptible to malicious attack?
• Incentives for advertising true capacity
  – High capacity neighbors get more tokens
  – Allocate tokens with weighted fair queuing

Practical Considerations

• Query resilience: node death
  – Periodic keep-alive messages
  – Query responses are implicit keep-alives
• Determining node capacity
  – Function of bandwidth and “age” of node
• Finding rare items
  – Bifurcate the random walk every 10 hops

Outline

• Gnutella (quick recap)
• Super-Peer Network
  – [B. Yang, H. Garcia-Molina]
• Evaluation of results
• GIA: Scalable Gnutella
  – [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shenker]
  - Results: Simulations & Experiments
• Conclusion

Simulation Results

• Compare four systems
  – FLOOD: TTL-scoped, random topologies
  – RWRT: Random walks, random topologies
  – SUPER: Supernode-based search
  – GIA: search using GIA protocol suite
• Metric:
  – Collapse point: aggregate throughput that the system can sustain

Questions

• What is the relative performance of the four algorithms?
• Which of the GIA components matters the most?
• How does the system behave in the face of transient nodes?

System Performance

GIA outperforms SUPER, RWRT & FLOOD by many orders of magnitude in terms of aggregate query load
Factor Analysis

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Collapse point</th>
</tr>
</thead>
<tbody>
<tr>
<td>RWRT</td>
<td>0.00005</td>
</tr>
<tr>
<td>RWRT+OHR</td>
<td>0.005</td>
</tr>
<tr>
<td>RWRT+BIAS</td>
<td>0.0015</td>
</tr>
<tr>
<td>RWRT+TADAPT</td>
<td>0.001</td>
</tr>
<tr>
<td>RWRT+FLWCTL</td>
<td>0.0006</td>
</tr>
</tbody>
</table>

No single component is useful by itself; the combination of them all is what makes GIA scalable.

Transient Behavior

Even under heavy churn, GIA outperforms the other algorithms by many orders of magnitude.

Progress of Topology Adaptation

Nodes quickly discover each other and soon reach their target “satisfaction level.”

Outline

- Gnutella (quick recap)
- Super-Peer Network
  - [B. Yang, H. Garcia-Molina]
- Evaluation of results
- GIA: Scalable Gnutella
  - [Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, S. Shanker]
- Results: Simulations & Experiments
- Conclusion

Conclusion

- Query flooding in Gnutella does not scale (low query reach)
- Super-peers
  - Combine efficiency of centralized model with autonomy and robustness of distributed search
  - Take advantage of heterogeneity of capabilities across peers
- GIA: scalable Gnutella
  - 3–5 orders of magnitude improvement in system capacity
- Unstructured approach is good enough
  - Keyword searches are more prevalent, and more important than exact-match queries
  - Most queries are for hay, not needles \(\rightarrow\) DHTs overkill?