SplitQuest: Controlled and Exhaustive Search in P2P Networks

Péricles Lopes and Ronaldo A. Ferreira
Federal University of Mato Grosso do Sul, Brazil

IPTPS 2010
San Jose, CA, USA

Motivation

- Search in P2P networks still remains an open and challenging problem
- Scalable solutions for exact match queries (DHT – O(log n)). Limited query semantic
- Good solutions for keyword matches
- Exhaustive search for unstructured networks (BubbleStorm, Random Walks)

Existing Solutions

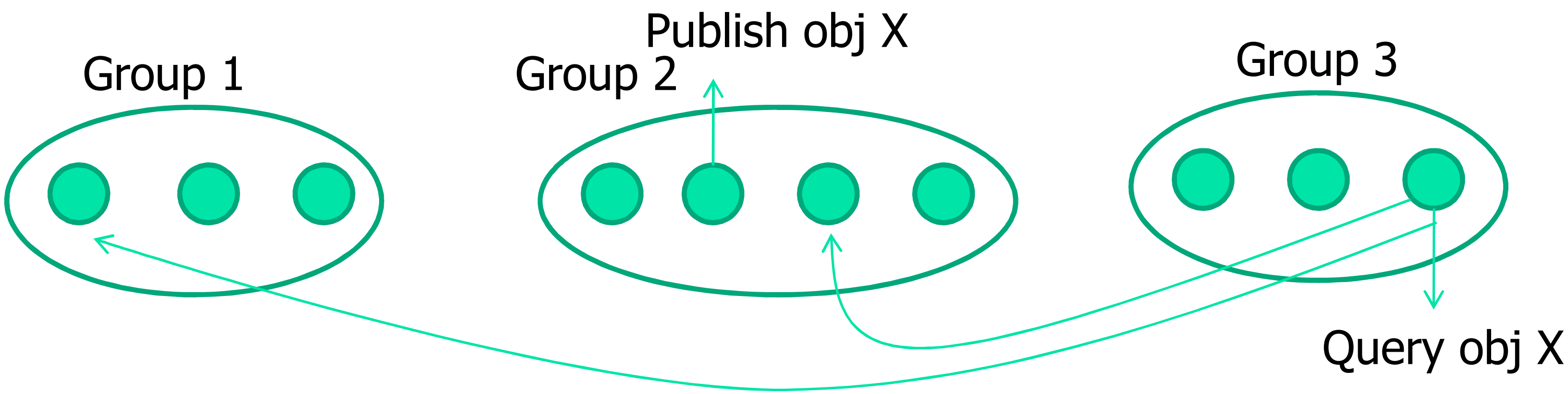
- Unstructured Networks
  - Data is replicated on the network
  - Complex queries
  - Search is difficult
  - Problems
    - Large amount of traffic (replication)
    - Network coverage is not guaranteed

Desirable Features

- Direct search for participants which have not been visited yet
- Support complex queries
- Explore peers’ heterogeneity

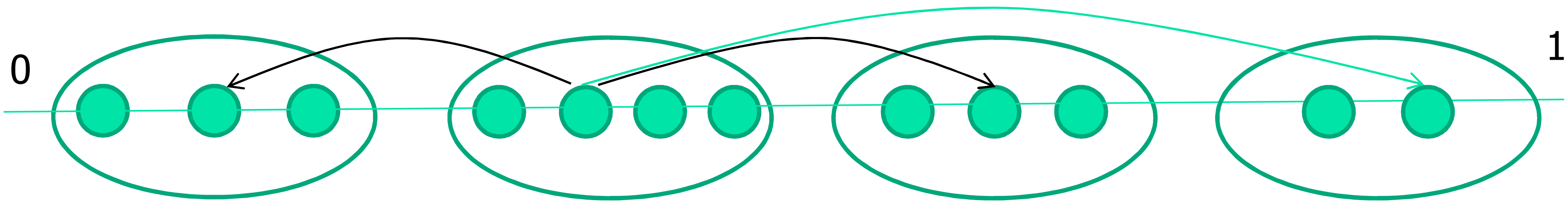
Splitquest Approach

- Peers can estimate the network size (n)
- Peers belong to groups
- Peers replicate their content in peers belonging to the same group
- Any peer from the group can answer queries for data stored on the group



SplitQuest: Topology

- Hybrid Approach
  - Peers are placed uniformly on a virtual ring
    - each peer has a predecessor and sucessor
    - peers are uniquely identified in the interval [0,1]
  - Peers make random connections
  - Contigous subinterval of the [0,1] interval = group
  - The size of the subinterval defines the number of peers in a group and consequently the number of groups in the network



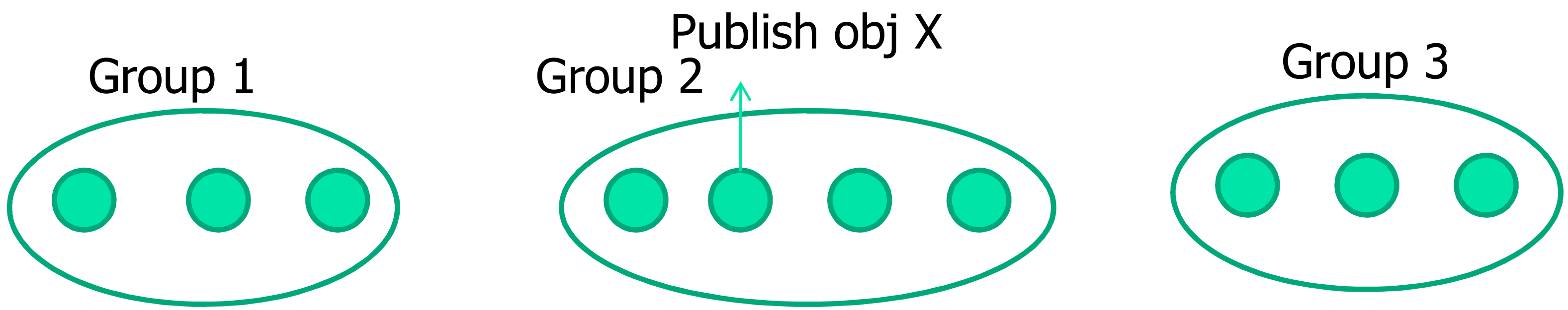
SplitQuest - Approach

- Uniform Distribution
  - Groups with approximately the same sizes

- Every peer has a shortcut for a node in the sucessor group and a shortcut for a node in the predecessor group

SplitQuest – Index Replication

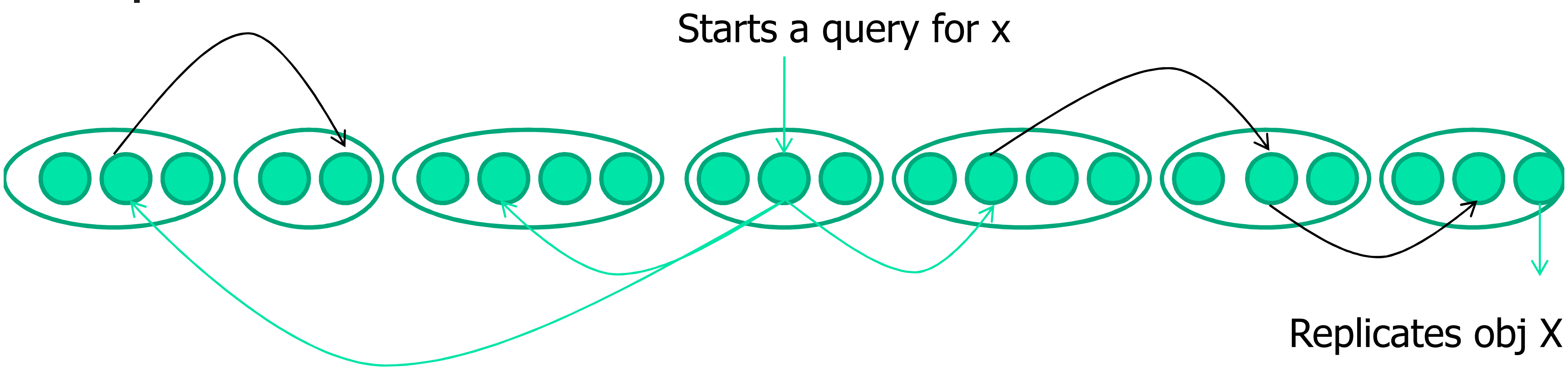
- A peer installs replicas in peers from the same group
  - Peers send replicas for the predecessor and sucessor nodes until entire group is covered



SplitQuest – Search Algorithm

- Search is directed to cover all groups in the network
  - When a message reaches a peer in the group, it means it reaches all peers in that group
  - Every message has an associated subinterval of [0,1]
    - The subinterval indicates which groups have not been covered yet
  - When a peer receives a message
    - Checks if it has connections (groups or shortcuts) for groups not covered yet
    - Sends a message to a connected peer with a subinterval of the initial message, the new subinterval does not contain the subinterval covered by the peer

Search Example



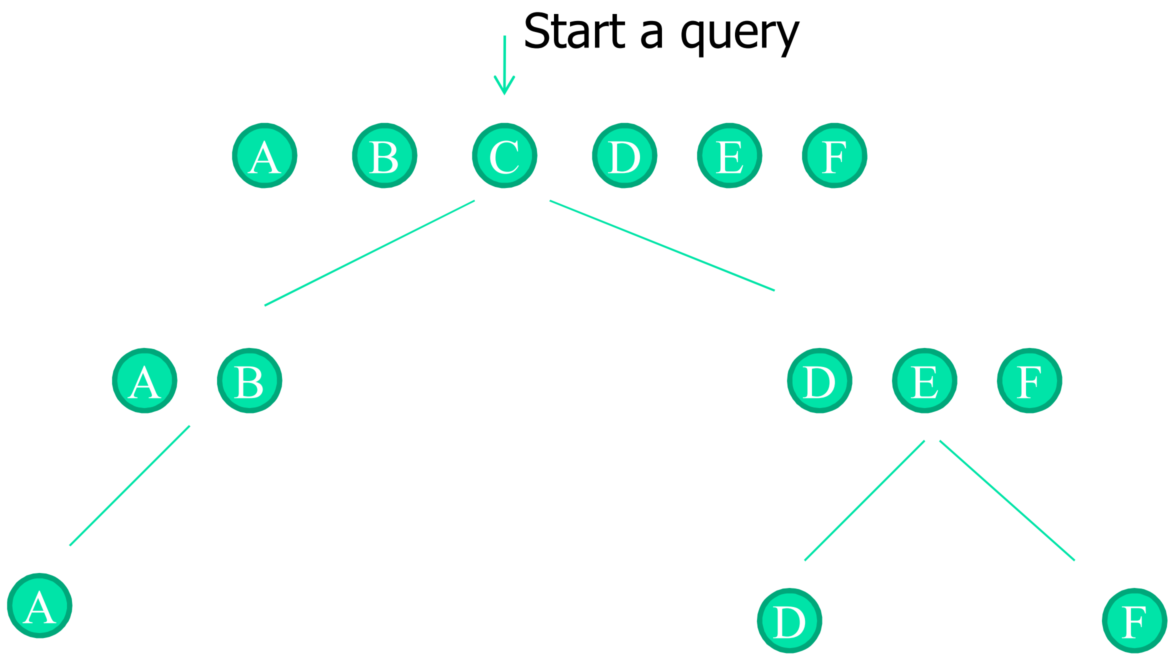
Group Size

- The expected number of peers in a contigous subinterval is proportional to the subinterval length.
- Choose a group size that minimizes the overhead of index replication and query propagation.
  - n: number of peers
  - d: size of a group
  - q: number of groups in the network (n/d)
  - M: number of search and data messages
  - $M = q + d$
  - $M = n/d + d$
  - Optimal solution $d = sqrt(n)$

Analysis

- Groups are visited only once
- Query messages propagate through groups in a random way → broadcast in a random tree

Analysis



All groups covered!

Analysis

- Devroye [2] shows that for well-behaved distributions the height of the three is:

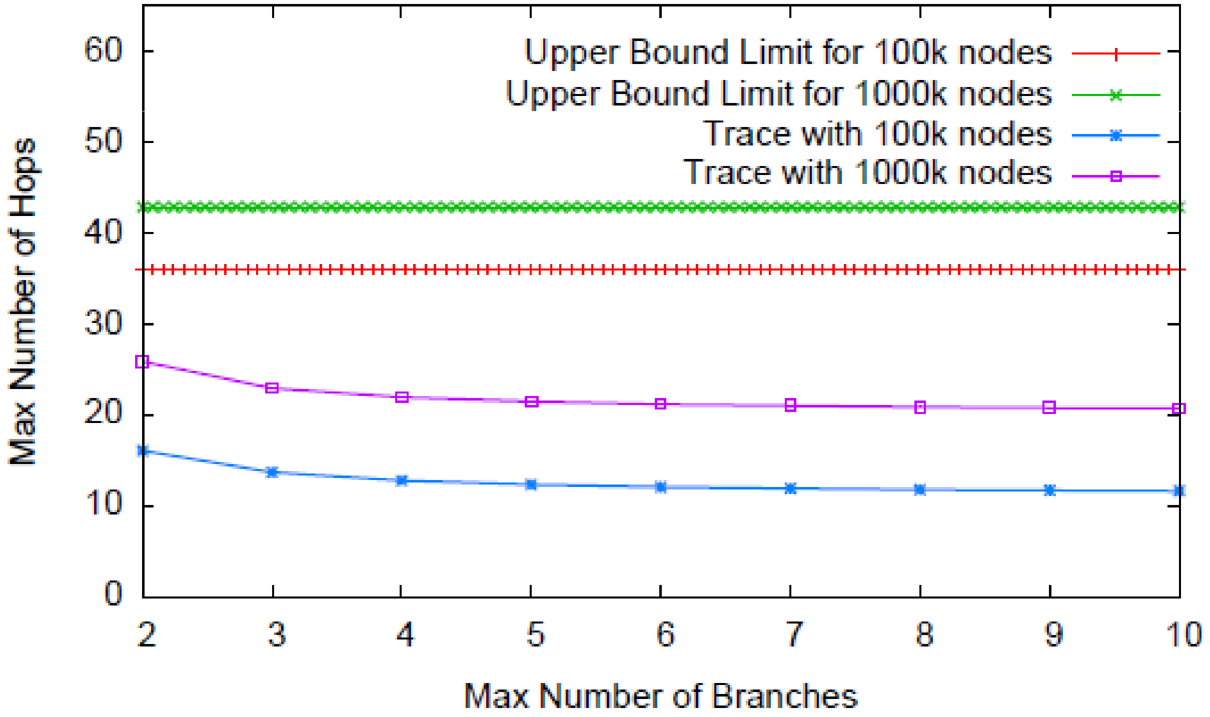
  - $H_q = 4.31 * \log q$

In SplitQuest, *q* is the number of groups

[2] Devroye, L. Universal Limit Laws for Depths in Random Trees. SIAM Journal on Computing, 1998.

Height of the Three

- Theorical Upper Bound Limit x Practical Limits

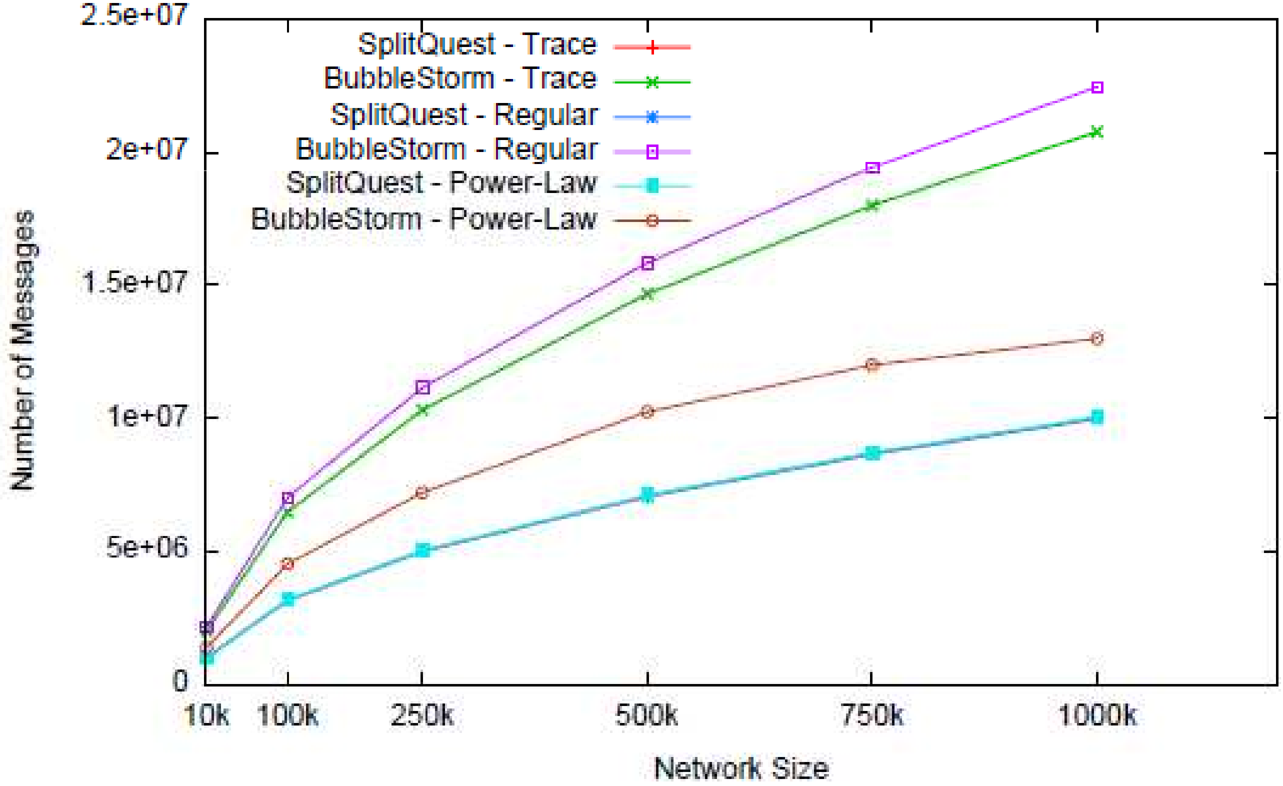


Simulations

- C++ simulator
- BubbleStorm (SIGCOMM'07)
- Metrics:
  - Success rate / Latency / Number of messages
- Topologies: power-law, regular and real trace
- Network sizes: from 1000 to 1000000 peers
- Similar to BubbleStorm simulation (wiki)
- Scenarios: static and dynamic
  - Data rate: insert 100 articles / second in random peers
  - Wait time of replication
  - Start search from a random peer
- Subinterval length = 1/sqrt(n)

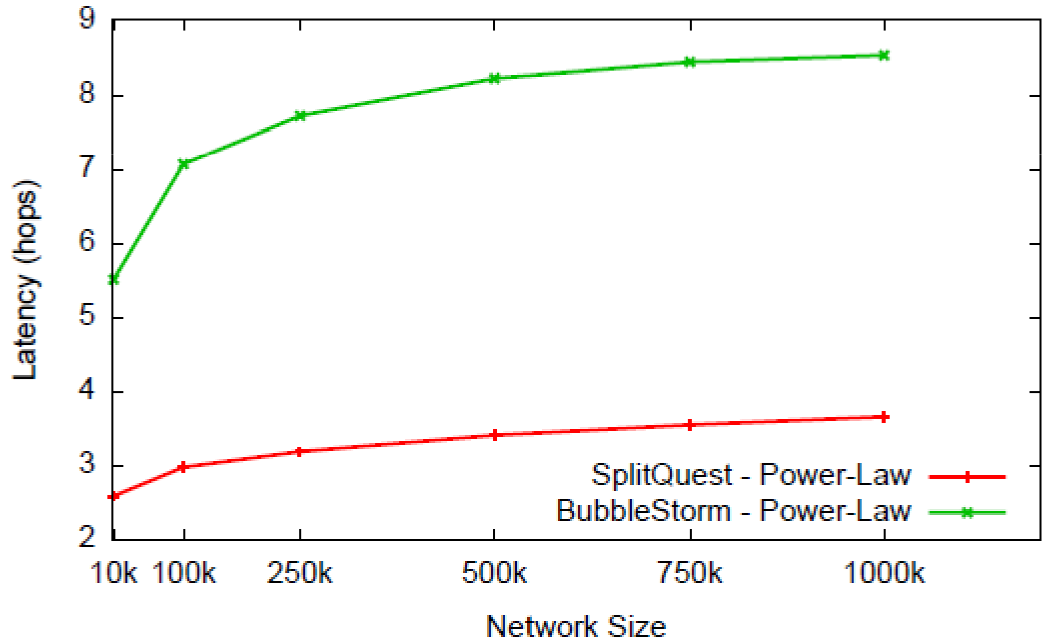
Preliminary Results

- Number of Messages



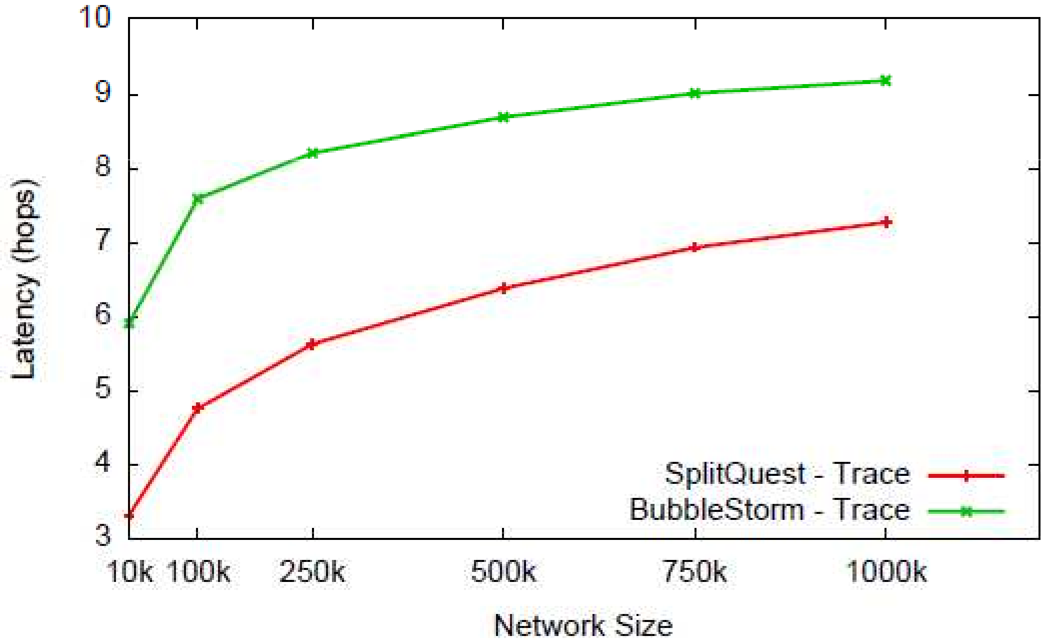
Preliminary Results

- Number of Hops: First Match Latency



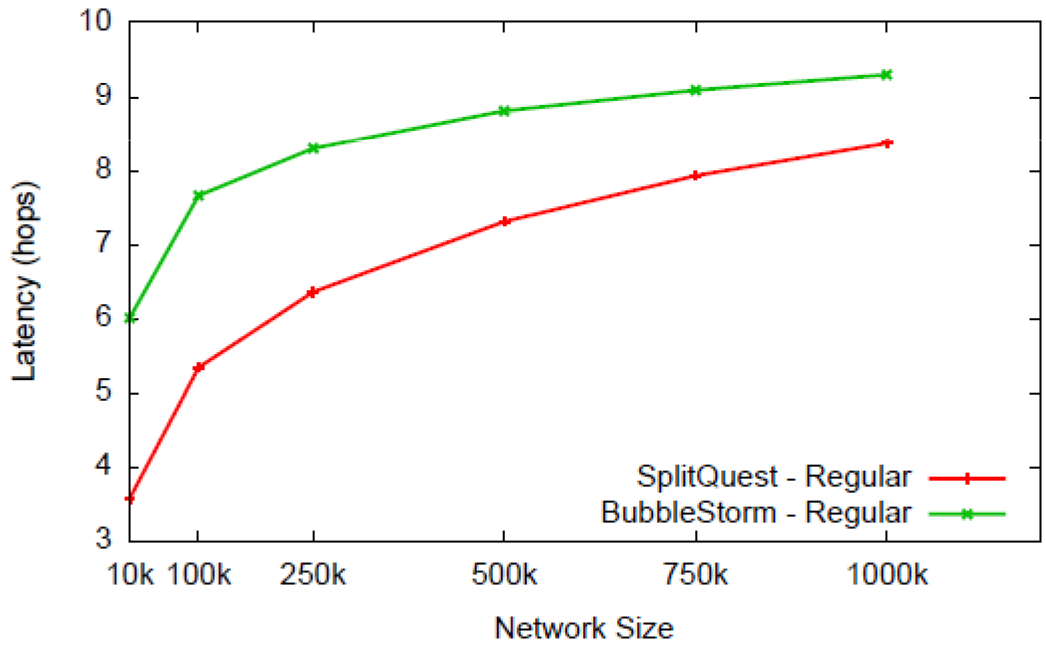
Preliminary Results

- Number of Hops: First Match Latency



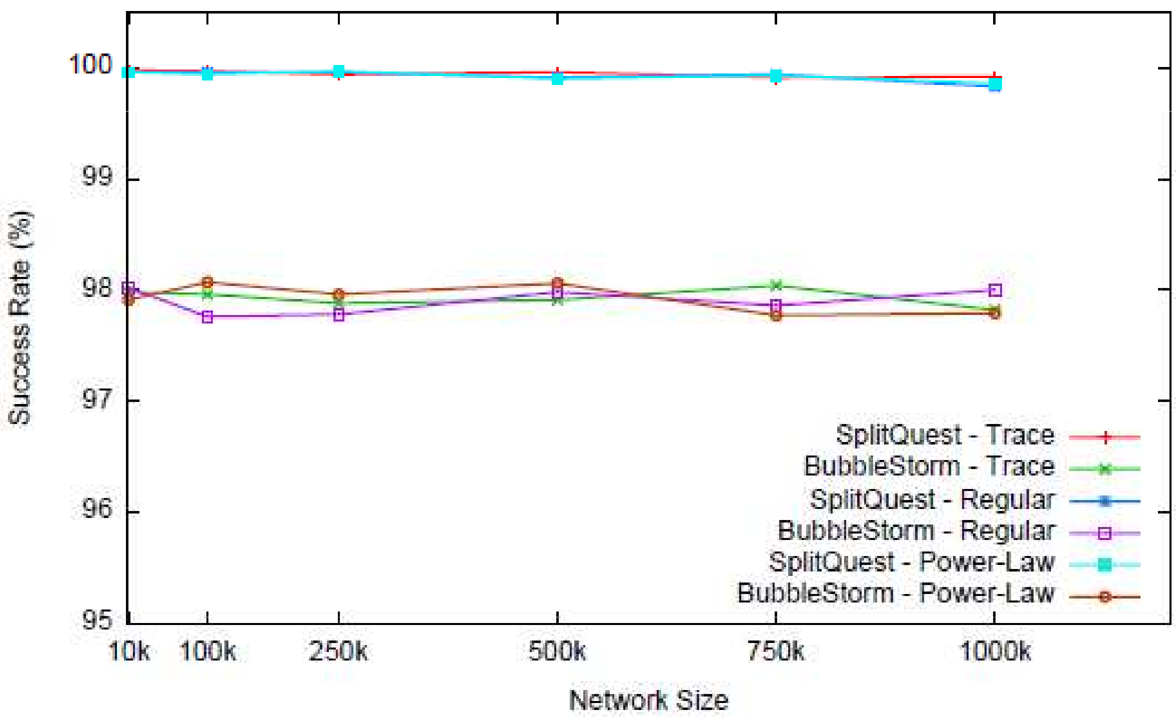
Preliminary Results

- Number of Hops: First Match Latency



Preliminary Results

- Success Rate: Churn



Conclusion

- SplitQuest appears promising
  - Fast
  - Supports complex queries
  - Avoids duplication of search messages

Future Work

- What is the impact of replicating metadata in more than one group?
- Can we allow groups of different sizes?
- How do probability distributions of node degrees and connections influence the three height?
- Can we have one single architecture that supports both DHT-like queries and complex queries efficiently?