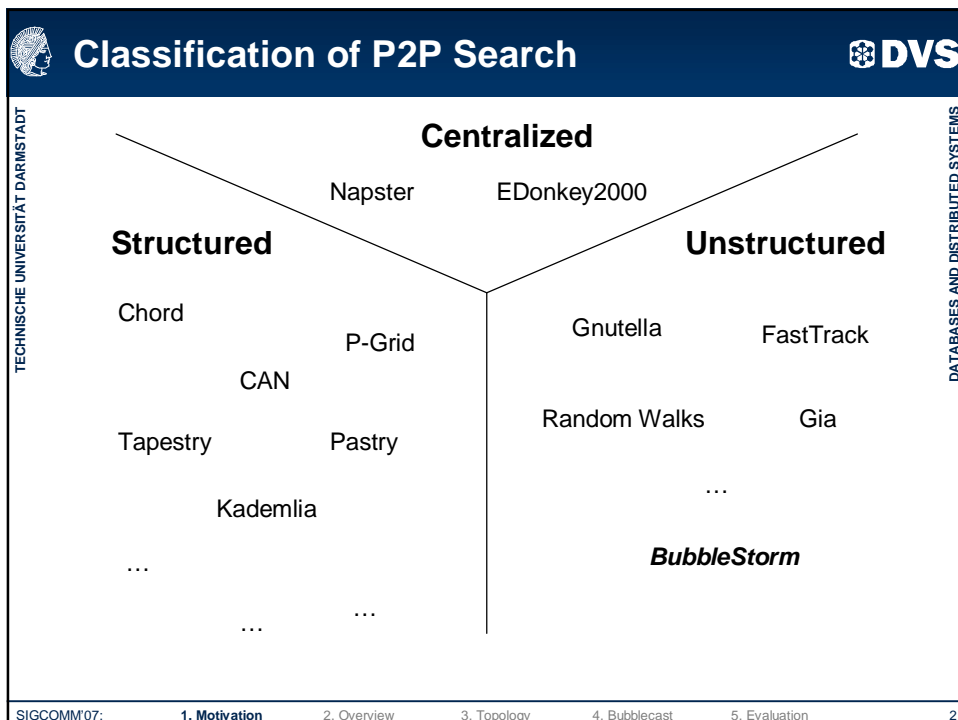


BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search

Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, Alejandro P. Buchmann
 Databases and Distributed Systems Group
 Technische Universität Darmstadt
 Germany



www.dvs1.informatik.tu-darmstadt.de





§ Centralized

- § Classic client/server architecture
- § Single point of failure
- § Maintenance costs

§ Unstructured

- § Place data somewhere; find it later somehow
- § Accounts for most deployed systems

§ Structured

- § Place data cleverly to make finding it easier
- § Extensively researched
- § Little real-world impact (so far)



Why unstructured search?



Query processing is independent from routing

§ This simplifies application development:

§ Implementing a query language locally

⇒ distributed implementation “for free”

§ Reuse existing libraries for query languages

§ SQLite, XPath, Lucene, ...

§ No need to invent a new algorithm per query language



Any selective query can be supported

§ One operation in unstructured systems can perform

§ a full-text search

§ range restriction on file size

§ hierarchical type selection

§ Structured systems break queries into small pieces

§ e.g. DHTs must transform the query into key-value

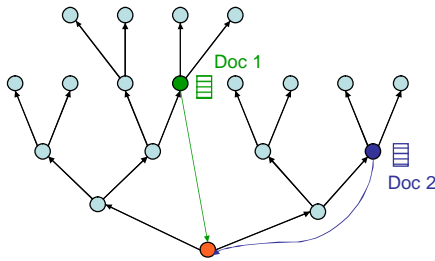
⇒ Cannot simply compare the cost of operations



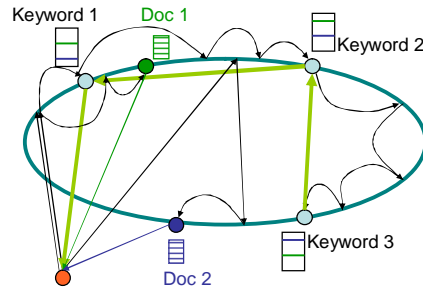
Example: Full text search



Unstructured Overlay



DHT with inverted index



- § One op. fetches documents
- § May contact more peers
- § Latency favours the unstructured approach
- § Relative bandwidth requirements highly parameter dependent
- § Perform multiple lookups
- § Transfer word lists (not via DHT)
- § Fetch documents



Not everything is uniform

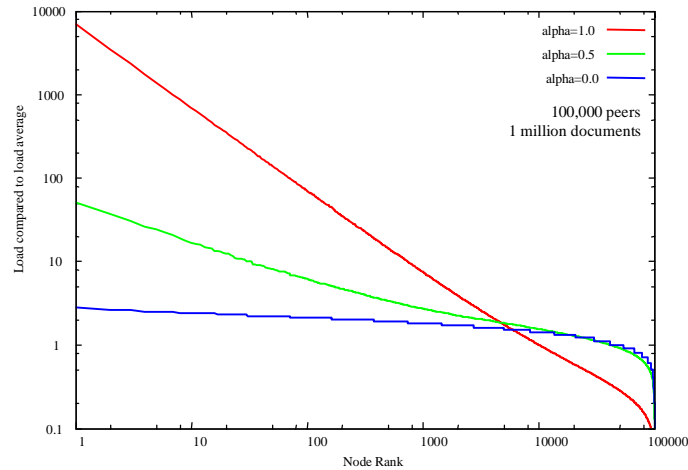


Natural load balancing

- § P2P applications often handle Zipfian loads
 - § Human text has $\alpha=1$
 - § YouTube has $\alpha=0.5$
- § An unstructured request can be served by any peer
- § Heterogeneity is accommodated by irregular degree
- § In comparison, adding a keyspace creates hot spots



Example: Zipfian Load



- § For natural languages (e.g. full text search) in a keyspace:
 - § Expected load on most the loaded peer is 7000x average
 - § The loaded peer probably has only average capacity



BubbleStorm Intuition



- § Replicate both queries and data
 - § $O(\sqrt{n})$ copies each (hidden constants unequal)
- § Data and queries rendezvous in the network



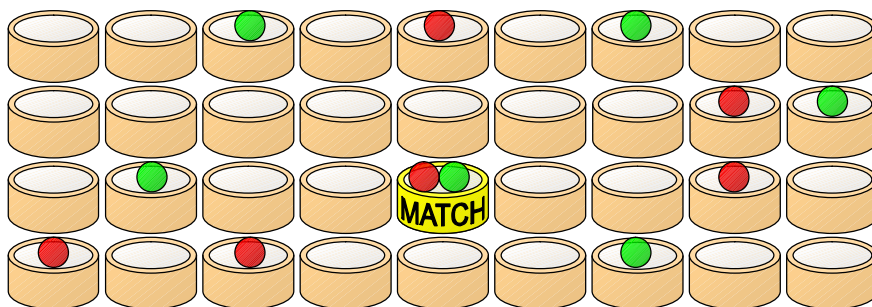


BubbleStorm: Random Replication



- § Place data replicas on random nodes
- § Nodes evaluate query replicas on all stored data
 - Where both data and query go, matches are found

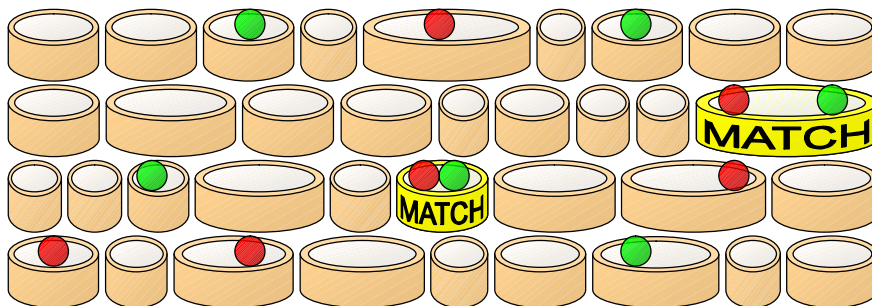
- § Collisions result from the birthday paradox



BubbleStorm: Exploiting Heterogeneity





- § Peers have different capacities
 - Faster peers receive more traffic
- § This is beneficial!
 - Contribution is squared





BubbleStorm Components



- § Random Topology 
 - § Allows efficient sampling of peers at random
- § Topology Measurement
 - § Computes network size and statistics
 - § See PODC'07 brief announcement
- § Bubblecast 
 - § Replicates queries/data onto peers quickly
- § Bubble Maintainer
 - § Preserves the correct number of replicas

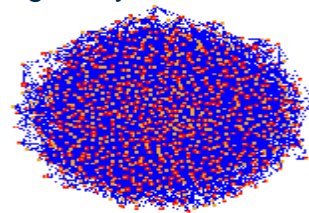
 Covered in this talk



Random Multigraph Topology



- § Random graphs support the birthday paradox
 - § Exploring an edge leads to a randomly sampled peer
 - ⇒ creation of random node subset (bubble) is cheap
- § Node degree is chosen proportional to bandwidth
 - § As random walks (and bubblecasts) follow edges with equal probability
 - Utilization will be balanced for heterogeneity



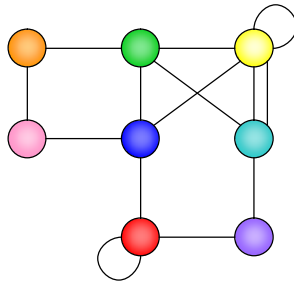


Random Multigraph Topology

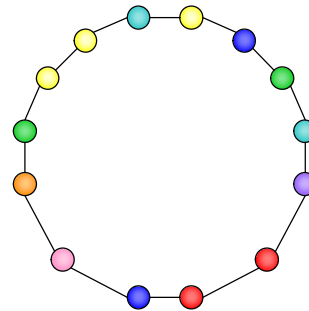


TECHNISCHE UNIVERSITÄT DARMSTADT

Topology



Eulerian Cycle



DATABASES AND DISTRIBUTED SYSTEMS

- § The topology is a random permutation of its edges
- § It is modified only when peers join or leave

SIGCOMM'07:

1. Motivation

2. Overview

3. Topology

4. Bubblecast

5. Evaluation

15

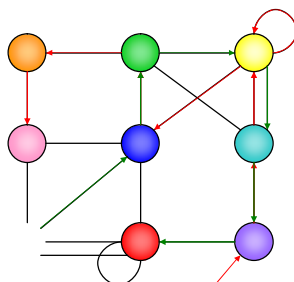


Join Algorithm



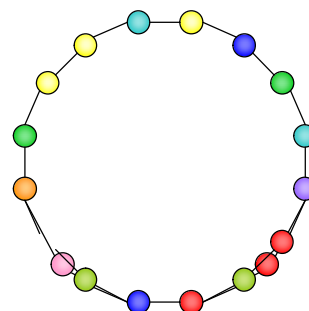
TECHNISCHE UNIVERSITÄT DARMSTADT

Topology



Joining Node

Eulerian Cycle



DATABASES AND DISTRIBUTED SYSTEMS

- § Contact bootstrapping node
- § Random walk finds a random edge
- § Split the edge and insert in between
- § Multiple joins are executed in parallel or iteratively

SIGCOMM'07:

1. Motivation

2. Overview

3. Topology

4. Bubblecast

5. Evaluation

16

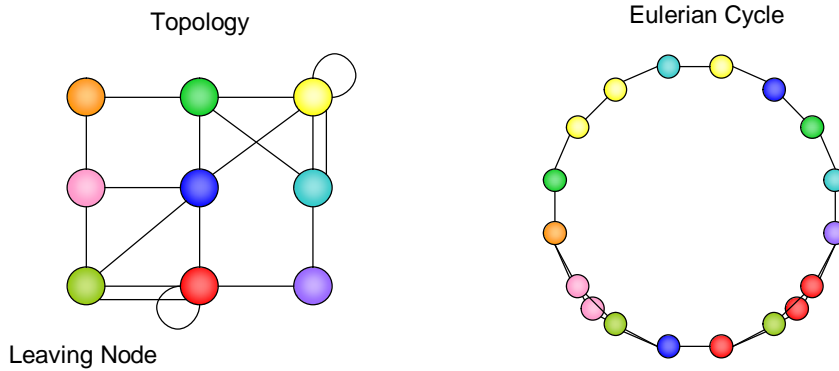


Leave Algorithm



TECHNISCHE UNIVERSITÄT DARMSTADT

DATABASES AND DISTRIBUTED SYSTEMS



- § Leave splices two neighboring edges together
- § Join and leave do not change degree of neighbors

SIGCOMM'07:

1. Motivation

2. Overview

3. Topology

4. Bubblecast

5. Evaluation

17

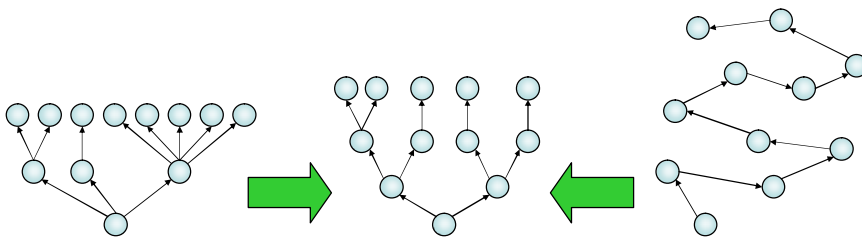


Bubblecast Motivation



TECHNISCHE UNIVERSITÄT DARMSTADT

DATABASES AND DISTRIBUTED SYSTEMS



Flooding

- + low latency
- + reliable
- imprecise node count
- unbalanced link load

Bubblecast

- + low latency
- + reliable
- + precise node count
- + balanced link load

Random Walk

- high latency
- unreliable
- + precise length
- + balanced link load

node counter (not hops)
fixed branch factor

branch in every step

SIGCOMM'07:

1. Motivation

2. Overview

3. Topology

4. Bubblecast

5. Evaluation

18

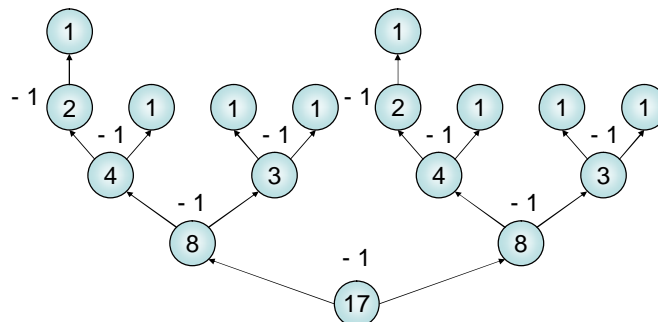


Example Bubblecast Execution



A counter specifies the number of replicas to create

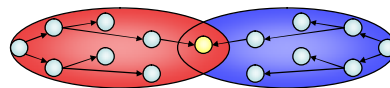
- § Decrement the counter for matching locally
- § Split the counter between two neighbors
- § Counters are always integral
- § Forwarding terminates when counter reaches 0
- § Final routing depth differs by at most one hop



Bubblecast Properties



- § Used for query *and* data replication
- § Fixed branch factor balances load
 - § Same stationary distribution as a random walk
- § Counter for edges crossed, not hops
 - § Precisely controls replica count
- § Logarithmic routing depth
 - § Slightly deeper than flooding
- § Message loss reduces replication by $\log(\text{size})$
- § Samples random nodes
 - ... due to random topology





§ BubbleStorm costs roughly $c\sqrt{n}$ bandwidth / op. to provide exhaustive search with $P(\text{failure}) < e^{-c^2}$

c	1	2	3	4
P(success)	63.21%	98.17%	99.99%	99.99999%

§ The full equation ($e^{-c^2 + c^3 HY}$) is complicated by

- § Heterogeneous peer capacity (H)
- § Dependent sampling (due to repeated withdrawals)
- § Unequal query and post traffic (Y ; BS optimizes this)

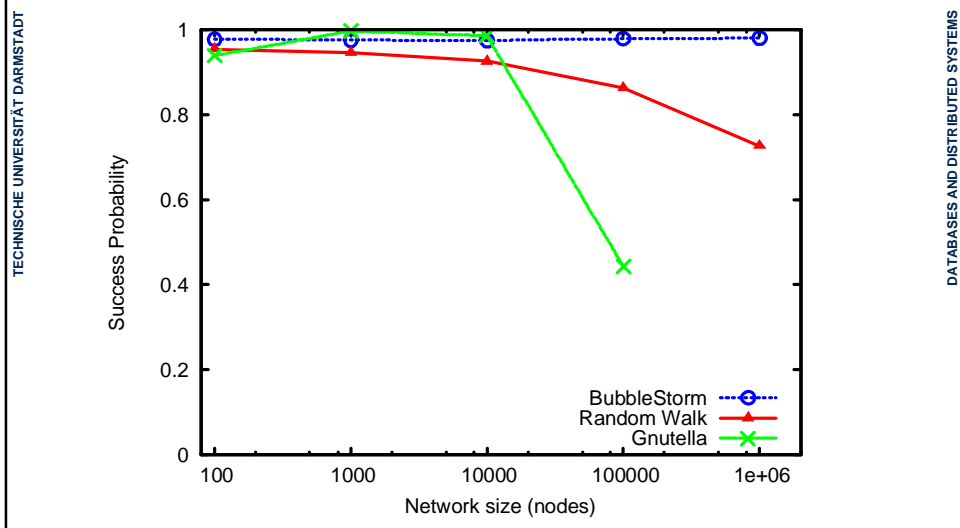
Full details in the paper



Simulation parameters

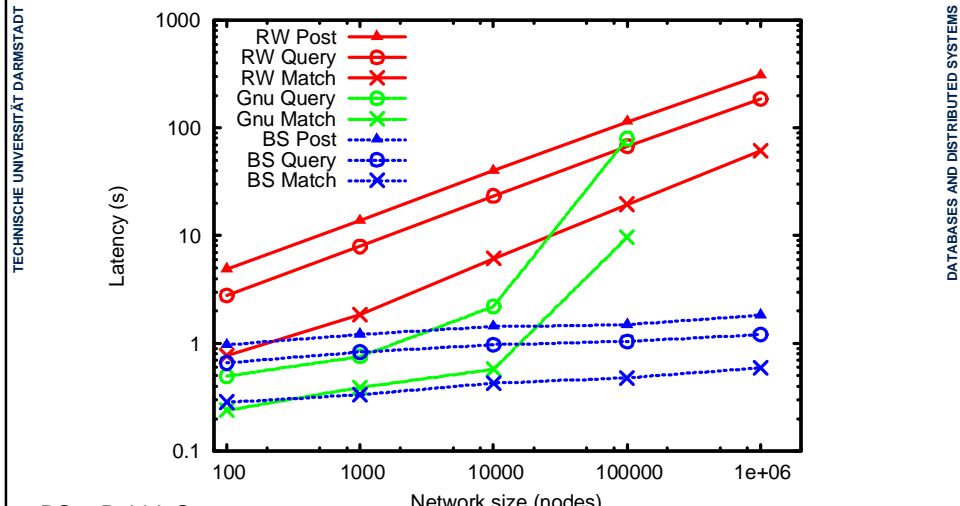
- § 1 million peers with heterogeneous upstream:
 - § 60% 16kB/s, 25% 32kB/s,
 - § 10% 128kB/s, 5% 1.2MB/s
- § 100B query every 5 user minutes (80/20 injection)
- § 2kB meta data stored every 30 user minutes
- § Exponential lifetime, mean 60 minutes
- § 10% of leaves are crash failures
- § Target reliability is 98.2% ($c=2$)

TECHNISCHE UNIVERSITÄT DARMSTADT **Apples to Apples Performance: Success** **DVS**

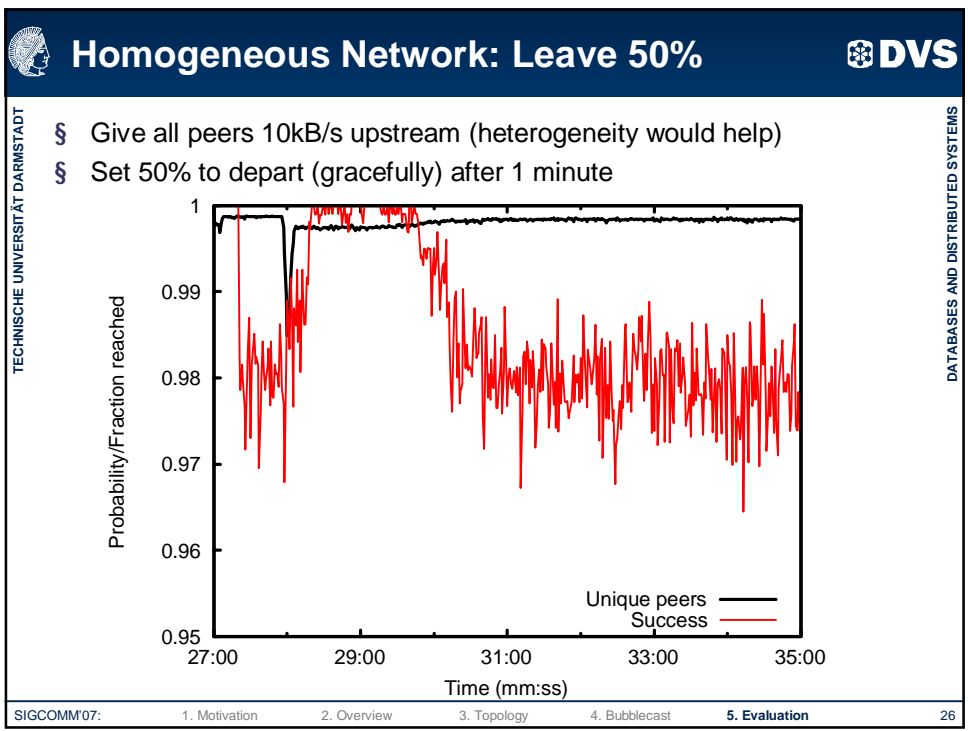
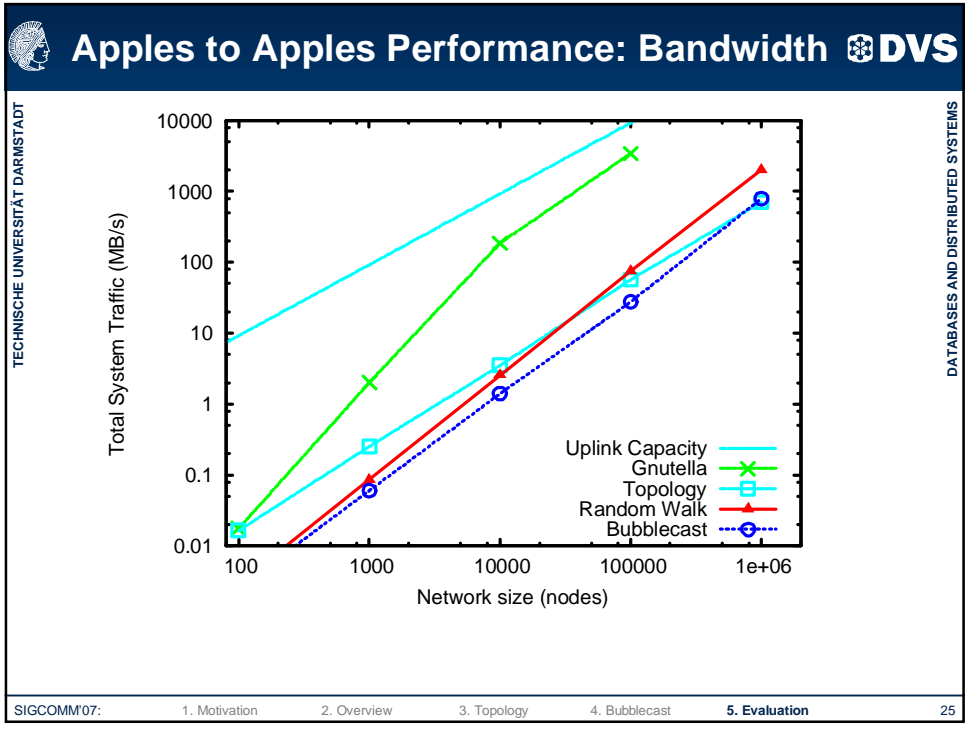


Search success remains unaffected by increasing the network size

TECHNISCHE UNIVERSITÄT DARMSTADT **Apples to Apples Performance: Latency** **DVS**



BS = BubbleStorm
 Gnu = Gnutella
 RW = Ferreira P2P'05
 Post = Data replicated
 Query = Query completed
 Match = First hit found

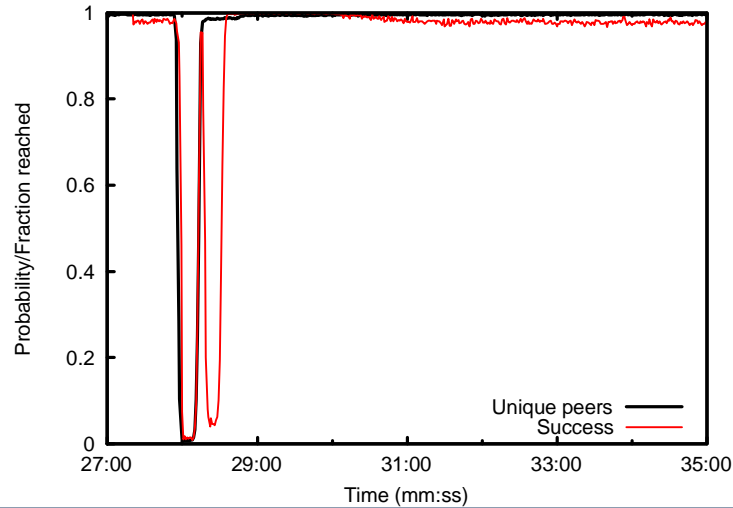




Homogeneous Network: Crash 50%



- § Crash 50% of peers after 1 minute
- § Echo effect: posted data is missing



Current and Future Work



- § Replica preservation in persistent bubbles
 - § Sustain bubble sizes under churn
 - § Scale bubble sizes with network size / composition
- § Update content
 - § Non-destructive, versioned updates
 - § Delete with death certificates
- § Release implementation



BubbleStorm Properties in Recap



- § Unstructured
 - § Queries may be in any language
- § Heterogeneous
 - § Exploits peers with varied capacity
- § Load Balanced
 - § Stationary utilization distribution is flat
- § Resilient
 - § Survives 50% crash fail and 90% leave
- § Exhaustive
 - § All matches of an operation can be retrieved
- § Probabilistic
 - § Success is a tunable guarantee



When does BubbleStorm fit?



- § Complex query languages
 - § Keyword search and beyond
- § Zipfian load with large α
 - § Partitioning data will create an all-pairs sub-problem
- § Mostly static data
 - § Allows us to trade post traffic for search traffic
- § Highly volatile networks
 - § Unstructured topology recovers quickly

Thanks for listening!

Questions

