Unstructured Networks: Search

Márk Jelasity

Outline

- Emergence of decentralized networks
- The Gnutella network: how it worked and looked like
- Search in unstructured networks
  - Random walk search in power law networks
  - Random walk search in random networks
  - Replication strategies
  - GIA: a prominent algorithm

Central index

- Index is stored on central servers: search is centralized
- Download is P2P
- For example, Napster
  - Works well, but
  - Not scalable
    - **Major investments needed if networks grows**
    - **Eg Google has 100,000+ servers already**
  - Not robust to attacks (legal and malicious)
- Incentive to go decentralized

First attempt to go decentralized: Gnutella

- Nullsoft (Justin Frankel)
- First client is spread via gossip...
  - AOL shuts down Nullsoft servers the day after the release
- Initially no attempt to control overlay topology
  - Emergent complex overlay
- Naive approach to search: flooding
- All communication (queries) are via flooding too

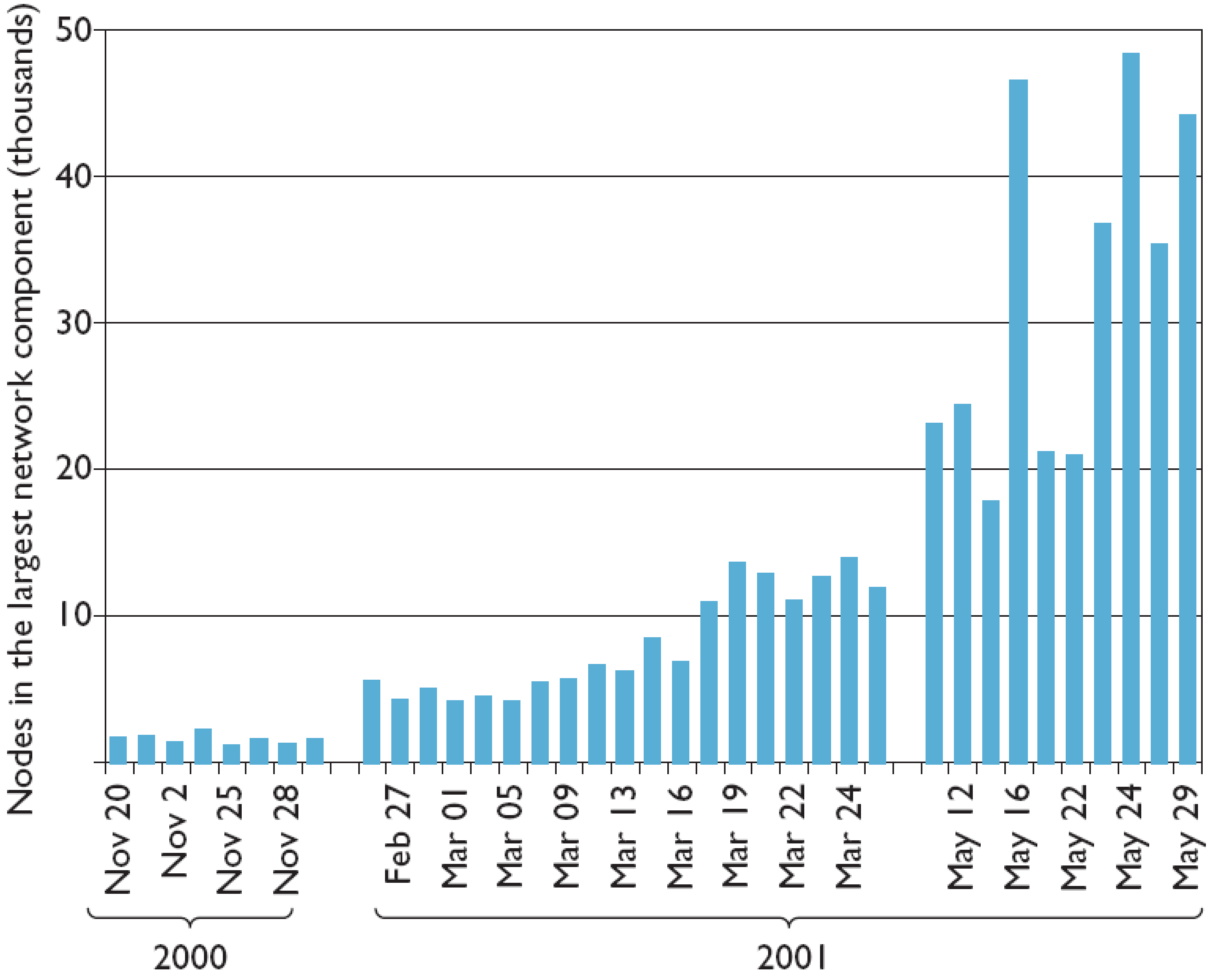
How Gnutella works?

- Gnutella protocol: flooding of queries
  - Ping, pong
    - **peer discovery at join and also continuously**
  - Query, query hit:
    - **Search hits are propagated back on the path of the search query**
- Join procedure
  - Find any member
  - Send ping message and collect pong messages

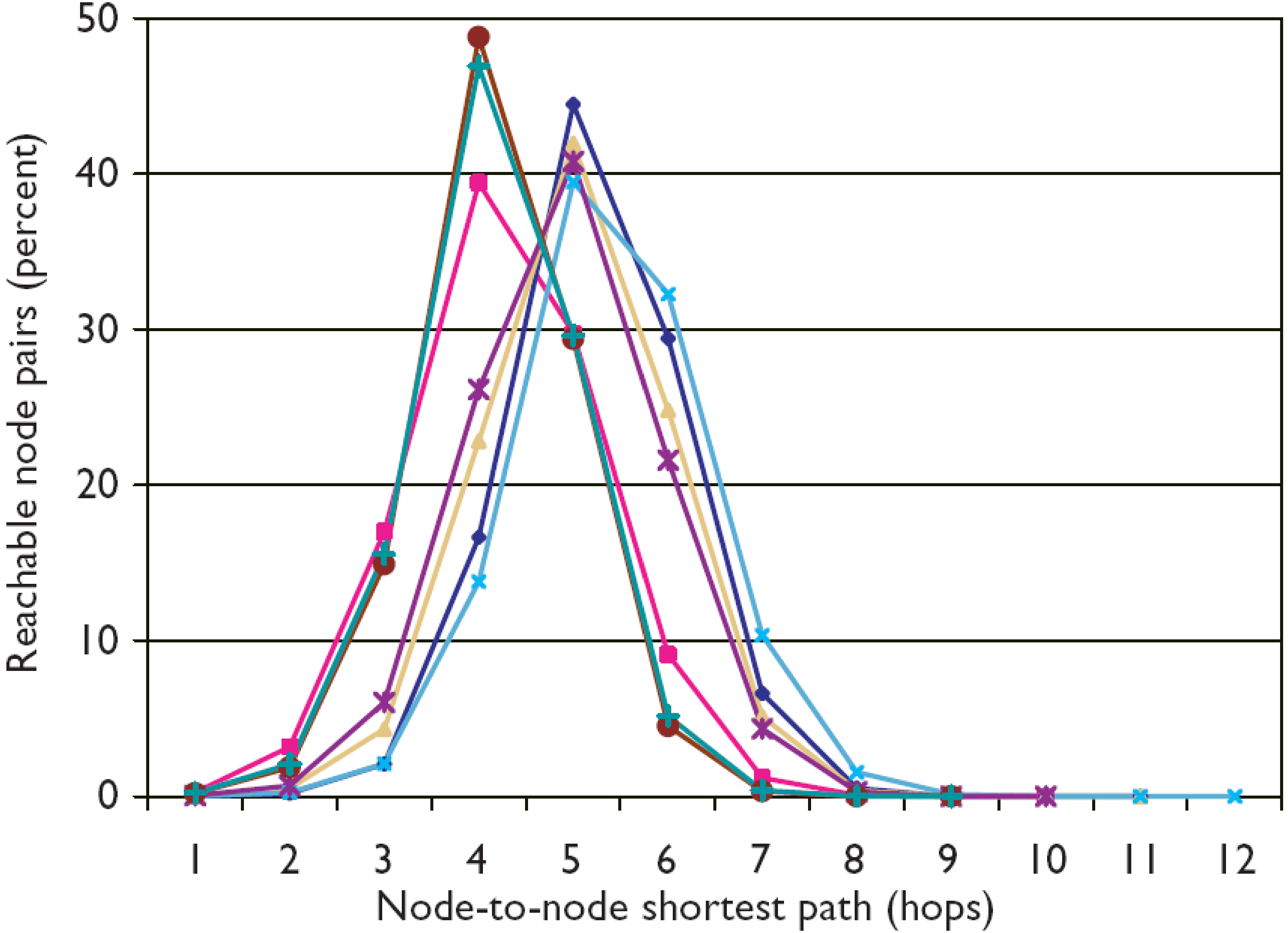
What is the Gnutella overlay looked like?

- Measurements by Ripeanu et al.
- Distributed Gnutella crawler collecting snapshots of size in the order of 50,000 for a year
- They discover complex network structure and highly dynamical composition: churn
  - 40% spend less than 4 hours in the network
  - 25% spend more than 24 hours

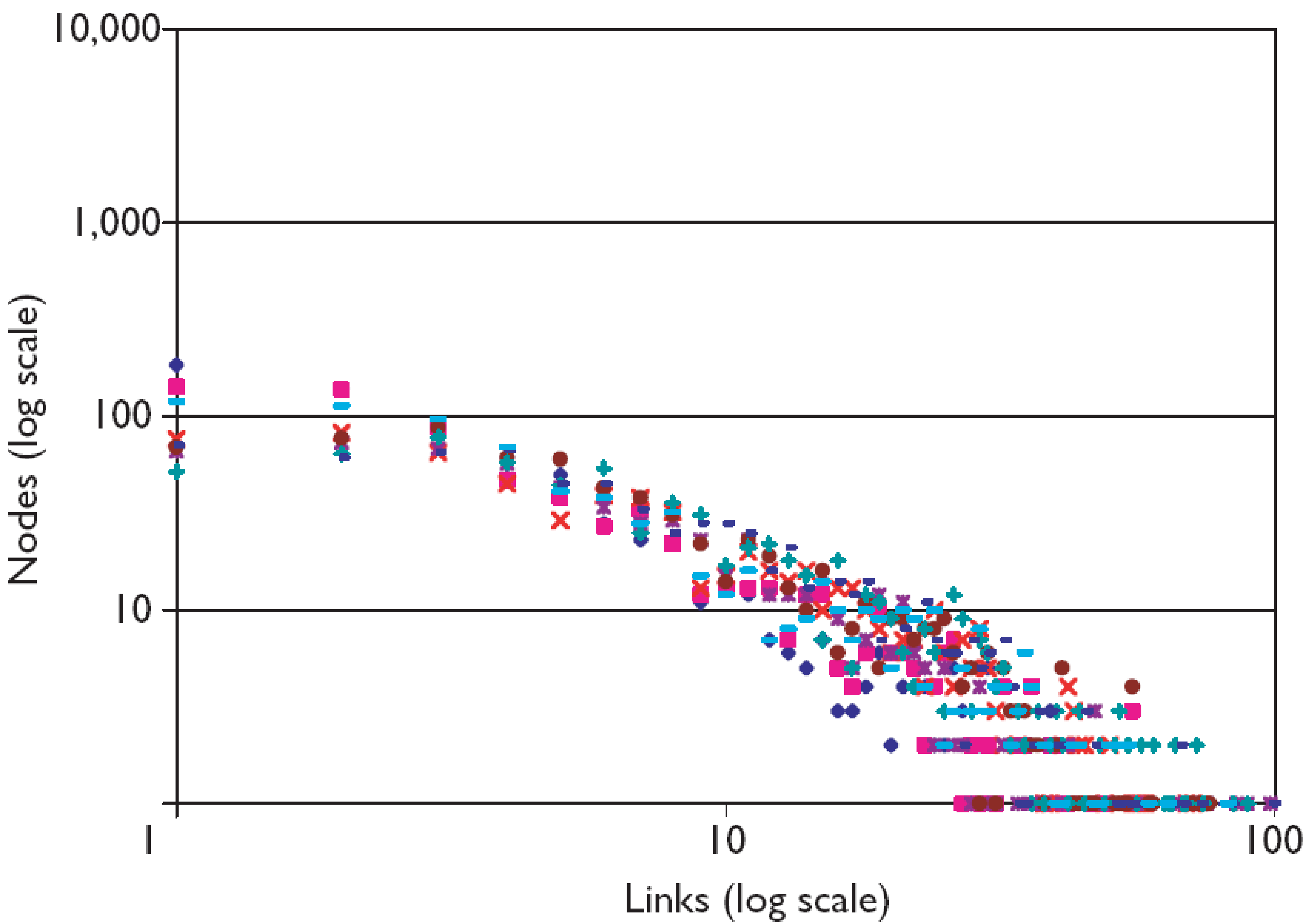
Growth of the network



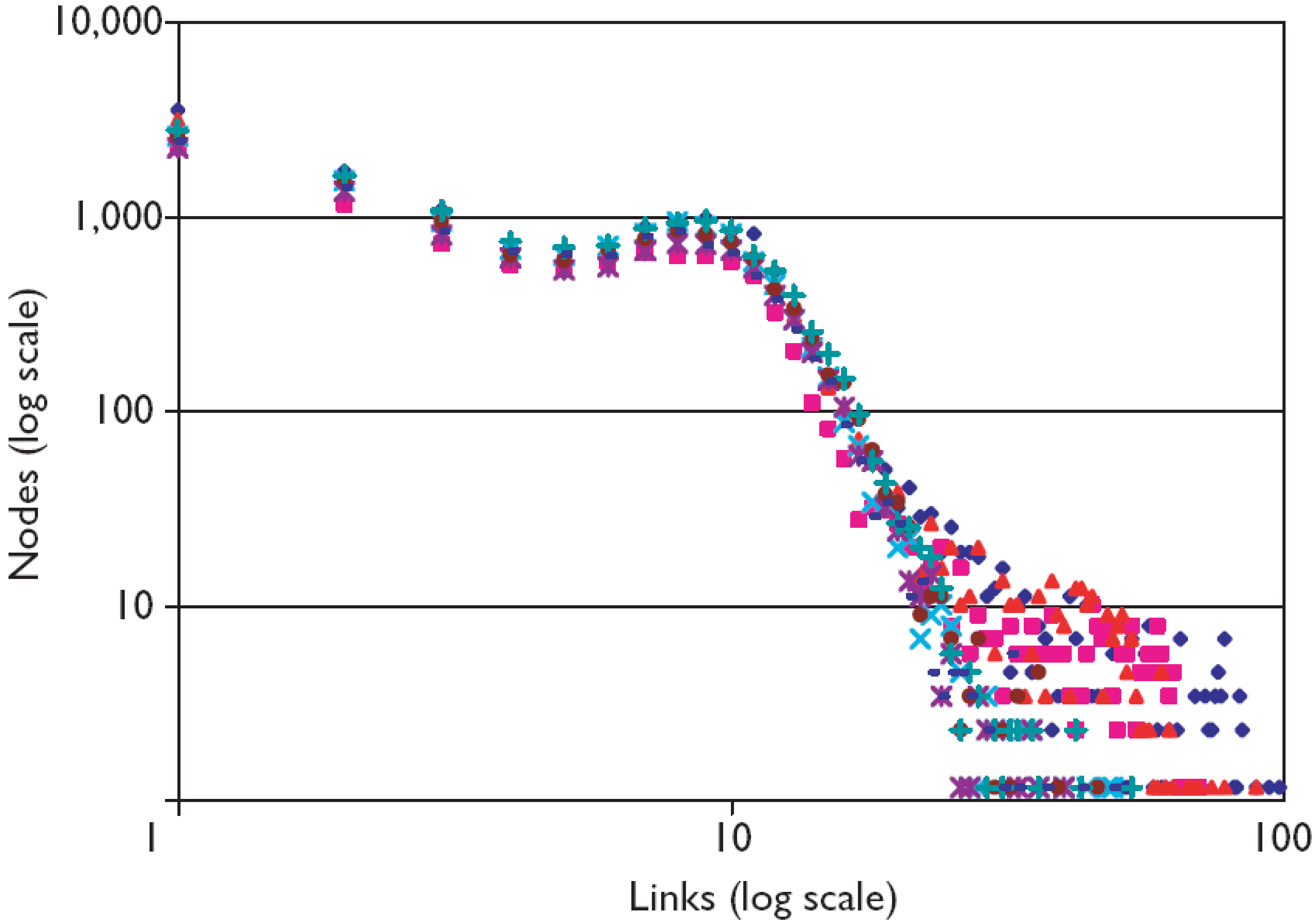
Path lengths



Degree distribution 2000 November

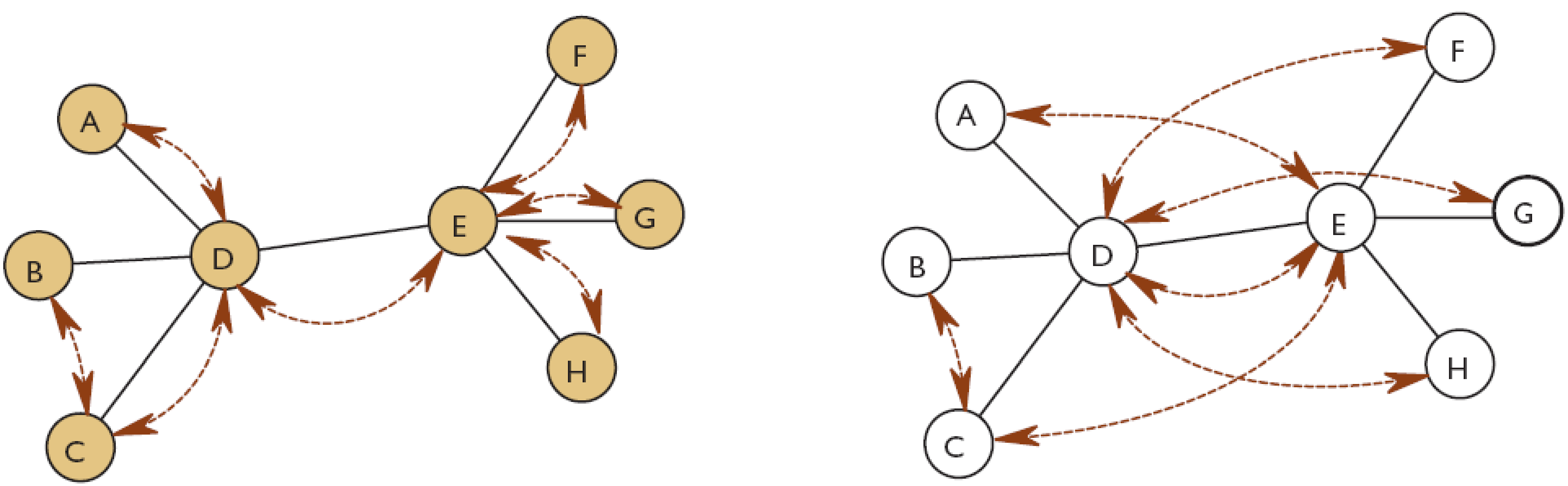


Degree distribution 2001 May



Underlying topology

- We have seen the that Internet is also power law
- Is there correlation between the overlay and the Internet?
- Ripeanu et al find that there is none



Search: flooding

- The default search model is flooding
  - Query is sent with a TTL, typically TTL=7
  - Query hits are propagated back on the path of the query
- Serious problems
  - Extremely wasteful with bandwidth
    - **A large (linear) part of the network is covered irrespective of hits found**
    - **Enormous number of redundant messages**
    - **All users do this in parallel: local load grows linearly with size**

Questions

- Does the scale-free topology has an effect on search protocols
  - Can we exploit it, or is it a disadvantage
  - What is the optimal search protocol for it
- In general, what search protocols can we come up with in an unstructured network
- What other techniques can we apply
  - Controlling topology to allow for better search
  - Controlling placement of objects (replication)

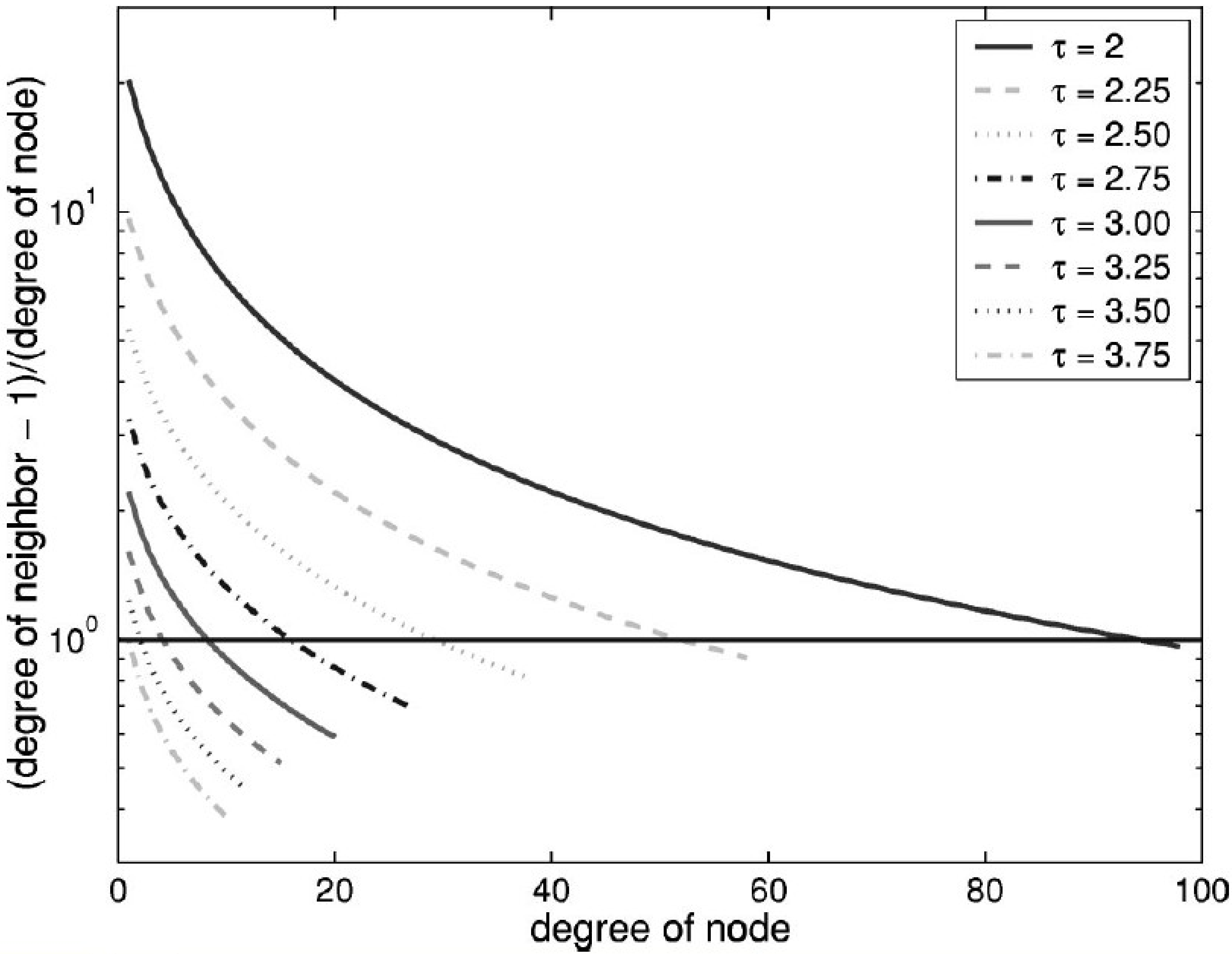
Search in scale-free networks

- Basic observations
  - In certain models if degree distribution is p_k then the distribution of the degree of a neighbor is proportional to kp_k (very important observation)
  - Nodes can easily store index of objects stored by their neighbors
- So in scale-free: high degree nodes are easy to find by (biased) random walk
- And high degree nodes can store the index about a large portion of the network
- Hint: a bit like the star topology

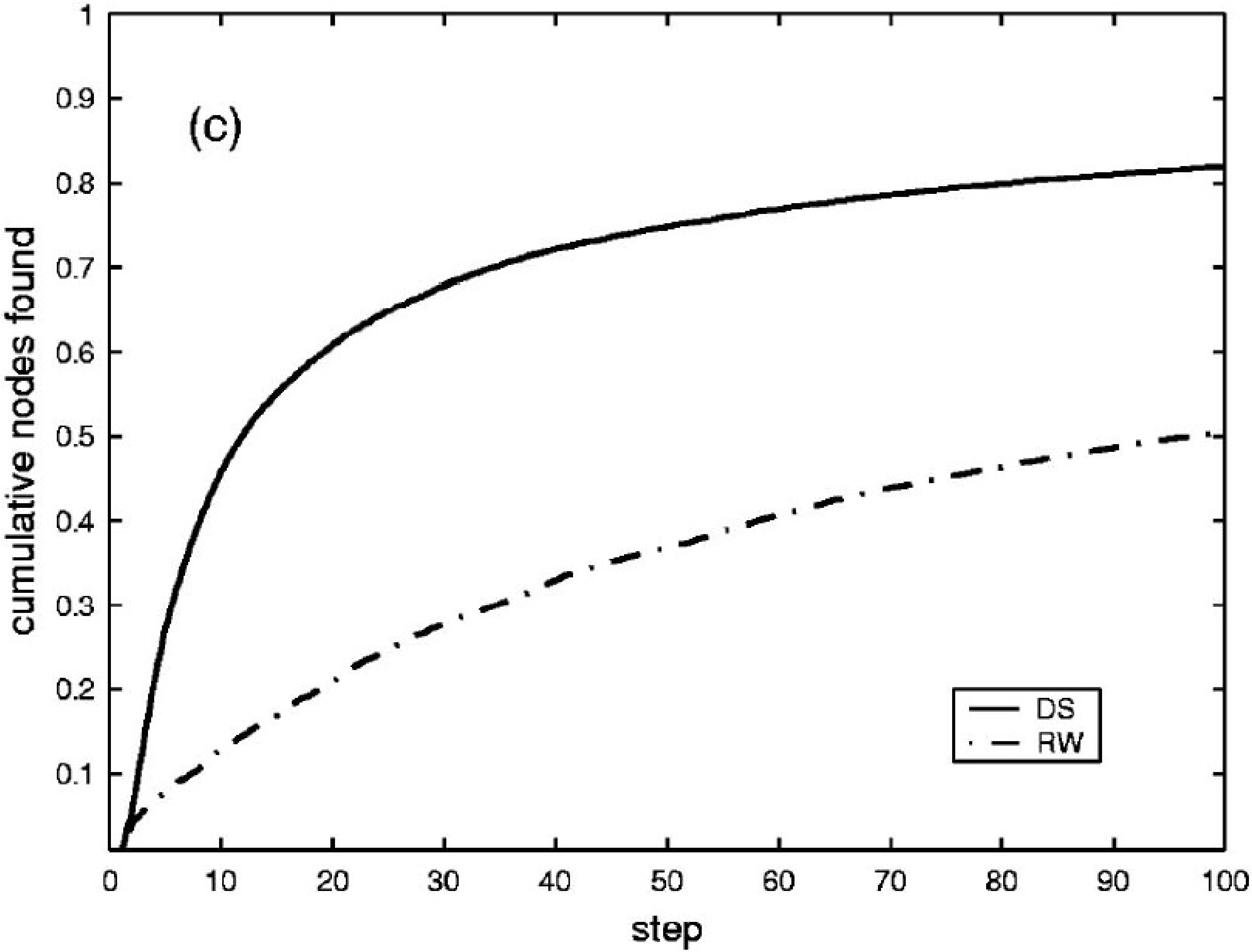
Search in scale-free networks

- Proposed algorithm variants
  - Random walk (RW)
    - **avoiding the visit of last visited node**
  - Degree-biased random walk (DS)
    - **Select highest degree node, that has not been visited**
    - **This first climbs to highest degree node, then climbs down on the degree sequence**
    - **Provably optimal coverage**
- Examined networks
  - Scale-free network with γ=2.1, abrupt cutoff
  - ER graphs
  - Different sizes, but N=10,000 if not specified

Climbing up the degree sequence

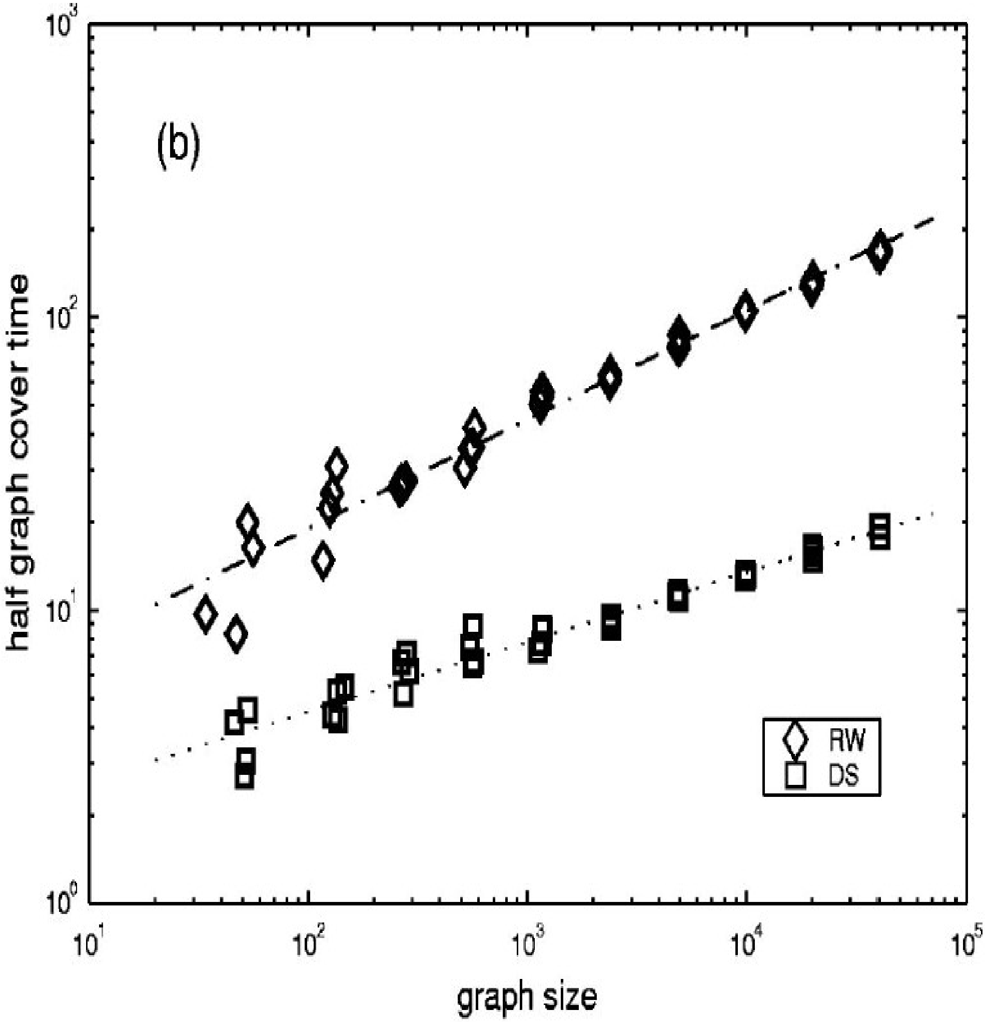


Speed of coverage

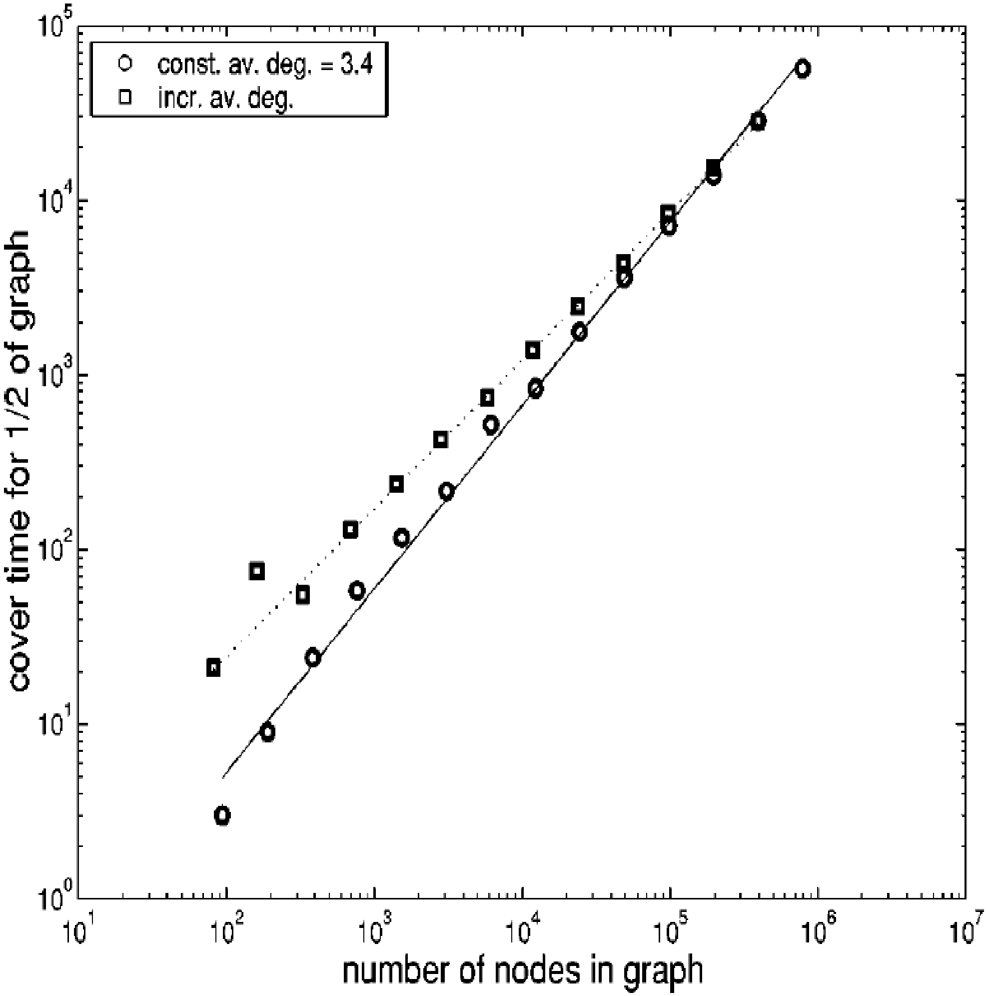


Half graph cover time

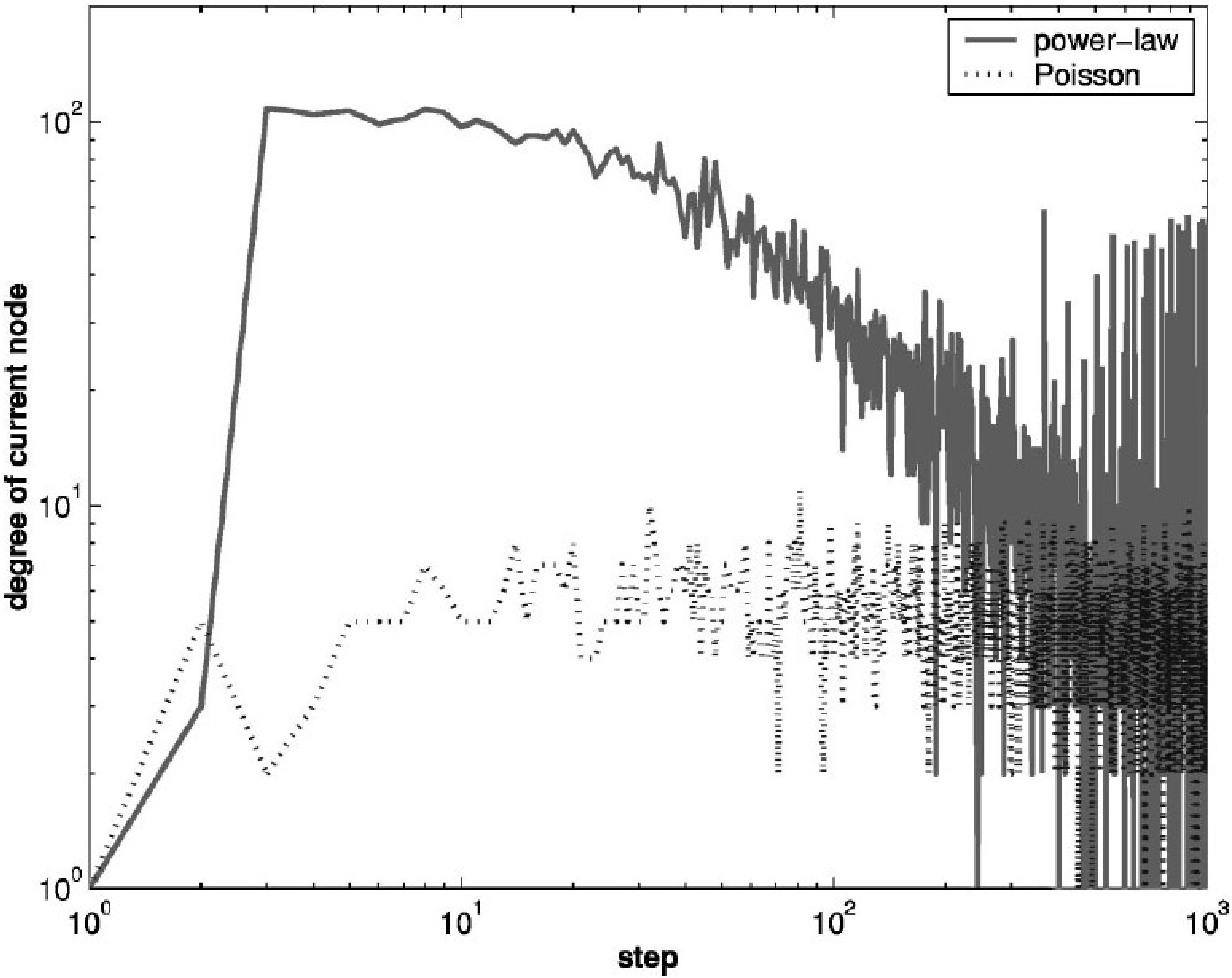
Scale free graph



ER graph



Visited node degrees



Conclusions

- Advantages
  - Takes advantage of scale-free distribution and speeds up search relative to ER graphs
  - Search time complexity is sublinear
- Disadvantages
  - Difficulty with rare objects (but this is a common problem of unstructured search)
  - Places very high load on high degree nodes
- Keeping this in mind, let's look at other topologies and see if they are better

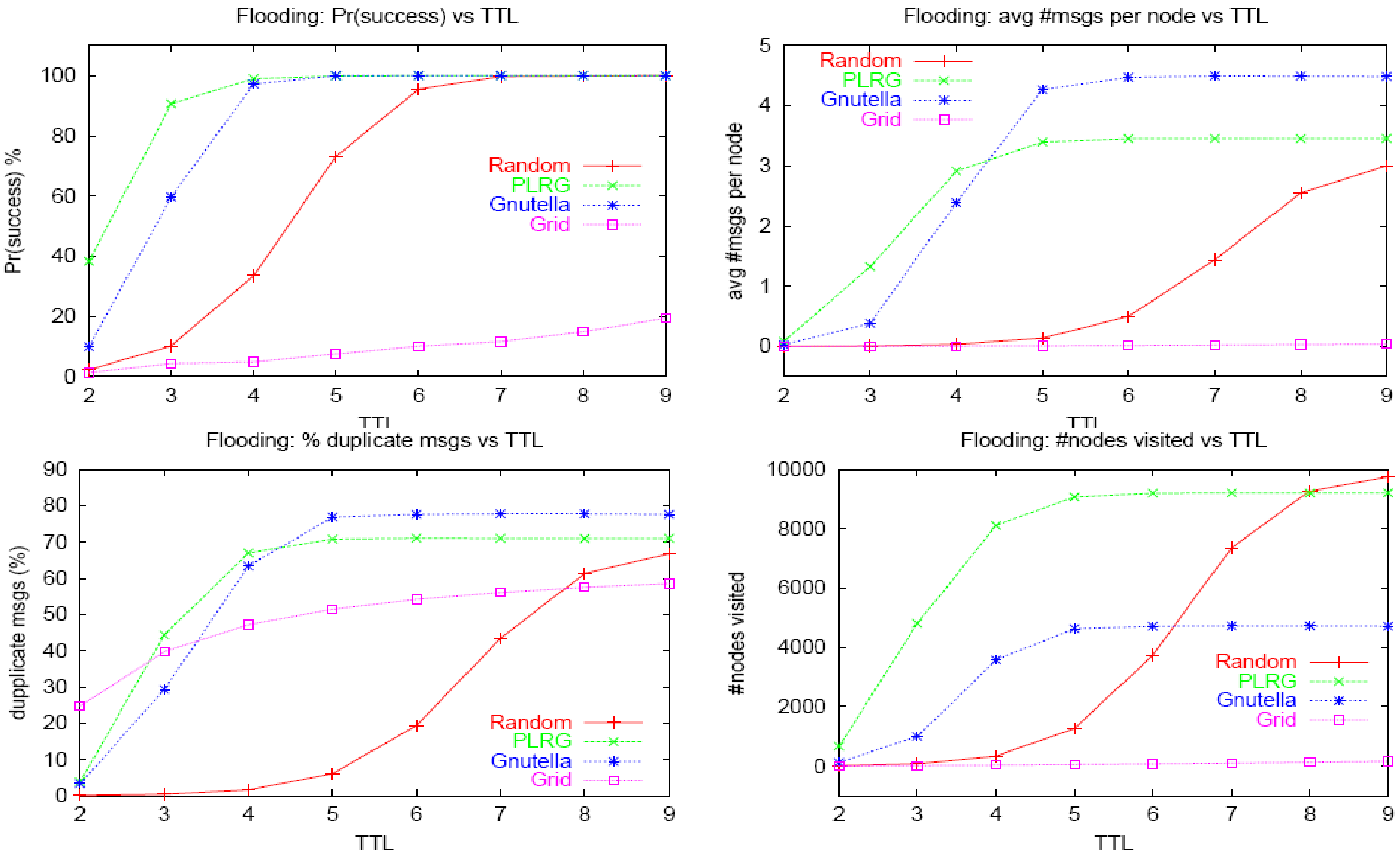
More search algorithms

- Expanding ring
  - Flooding with increasing TTL until result is found
  - The point is to avoid a fixed TTL
- K-walker
  - K independent random walks, to avoid message duplication in flooding and expanded ring
    - **With checking: in every 4 steps all walks check back if they need to go on or not**
    - **With state keeping: to implement self-avoiding walks**

Evaluation of search algorithms

- So far simplified model
  - ignored query and replication distribution, focused on coverage
- Three main components
  - Overlay network, Query modeling, Replication strategies
- Overlay networks
  - ER graph, avg. degree 4, N=10000
  - Power law (scale-free) graph, N=10000
  - Gnutella snapshot 2000 Oct, N=4000
  - 2-dim 100x100 grid

Problems with flooding



Evaluation of search algorithms

- Query distributions
  - q_i: the proportion of queries for object i
  - Uniform: all objects receive the same amount of queries
  - Power law: a few objects are very popular, many objects are not so much (heavy tail)
- Replication plays a role too
  - Spread copies of objects to peers: more popular objects can be found easier
  - File-sharing networks show an emergent replication behavior

Evaluation of search algorithms

- Object replication
  - Replication of object i typically proportional to q_i
  - Uniform: all objects receive the same amount of copies
  - Proportional: proportional to q_i
  - Square-root: proportional to square-root q_i
    - **Can be proven to be optimal in certain cases (see later)**
- Meaningful combinations of query/replication
  - uniform/uniform, power-law/proportional, power-law/square-root

Some results

distribution model		50 % (queries for hot objects)				100 % (all queries)			
query/replication	metrics	flood	ring	check	state	flood	ring	check	state
Uniform / Uniform	#hops	3.40	5.77	10.30	7.00	3.40	5.77	10.30	7.00
	#msgs per node	2.509	0.062	0.031	0.024	2.509	0.061	0.031	0.024
	#nodes visited	9220	536	149	163	9220	536	149	163
	peak #msgs	6.37	0.26	0.22	0.19	6.37	0.26	0.22	0.19
Zipf-like / Proportional	#hops	1.60	2.08	1.72	1.64	2.51	4.03	9.12	6.66
	#msgs per node	1.265	0.004	0.010	0.010	1.863	0.053	0.027	0.022
	#nodes visited	6515	36	33	47	7847	396	132	150
	peak #msgs	4.01	0.02	0.11	0.10	5.23	0.20	0.17	0.14
Zipf-like / Square root	#hops	2.23	3.19	2.82	2.51	2.70	4.24	5.74	4.43
	#msgs per node	2.154	0.010	0.014	0.013	2.308	0.031	0.021	0.018
	#nodes visited	8780	92	50	69	8983	269	89	109
	peak #msgs	5.88	0.04	0.16	0.16	6.14	0.12	0.17	0.16

ER graph

distribution model		50 % (queries for hot objects)				100 % (all queries)			
query/replication	metrics	flood	ring	check	state	flood	ring	check	state
Uniform / Uniform	#hops	2.37	3.50	8.95	8.47	2.37	3.50	8.95	8.47
	#msgs per node	3.331	1.325	0.030	0.029	3.331	1.325	0.030	0.029
	#nodes visited	8935	4874	147	158	8935	4874	147	158
	peak #msgs	510.4	132.7	12.3	11.7	510.4	132.7	12.3	11.7
Zipf-like / Proportional	#hops	1.74	2.36	1.81	1.82	2.07	2.93	9.85	8.98
	#msgs per node	2.397	0.593	0.011	0.011	2.850	0.961	0.031	0.029
	#nodes visited	6969	2432	43	49	7923	3631	136	145
	peak #msgs	412.7	58.3	4.9	5.1	464.3	98.9	12.7	11.7
Zipf-like / Square root	#hops	2.07	2.94	2.65	2.49	2.21	3.17	5.37	4.79
	#msgs per node	3.079	0.967	0.014	0.014	3.199	1.115	0.021	0.020
	#nodes visited	8434	3750	62	69	8674	4200	97	103
	peak #msgs	496.0	93.7	6.3	6.3	499.6	111.7	8.9	8.4

power-law graph

Notes for the experiments

- Parameters
  - 100 objects, avg replication ratio 1%
  - ER graph: TTL for flooding is 8, “check” and “state” are 32-walkers, γ=1.2 for query distribution
  - Power-law graph: same, but TTL=5
- Algorithms
  - Check: 32-walker with checking for termination
  - State: same as 32-walker, but also self-avoiding

Conclusions

- Fixed TTL must be avoided, be adaptive instead
- Avoid exponential spreading of queries
  - Note that this assumes that each object is replicated enough, otherwise search takes too long
- Message duplication must be avoided
  - ER random graph is best for this
  - So now: is scale-free good or bad?
- Square-root replication is optimal
  - How about dynamic methods for achieving that?

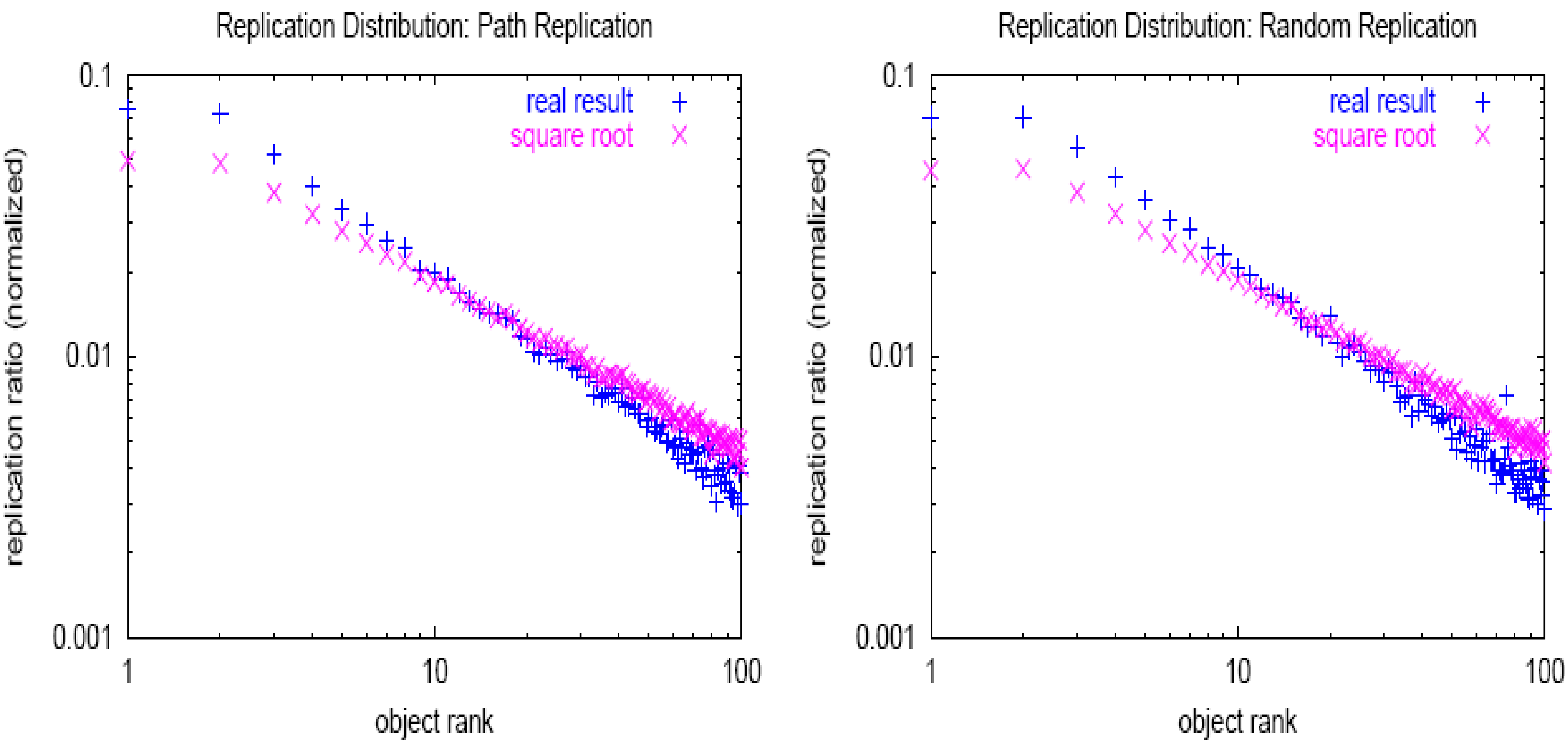
Replication strategies

- Average search size
  - The uniform and proportional strategies result in the same avg search size (avg number of random probes to find an object)
  - Avg search sizes for individual objects differ with the proportional strategy
  - Square-root can reduce avg search size
- Utilization ratio
  - Avg utilization ratio is 1 if we run each search until success
  - Variance is quite different with different strategies

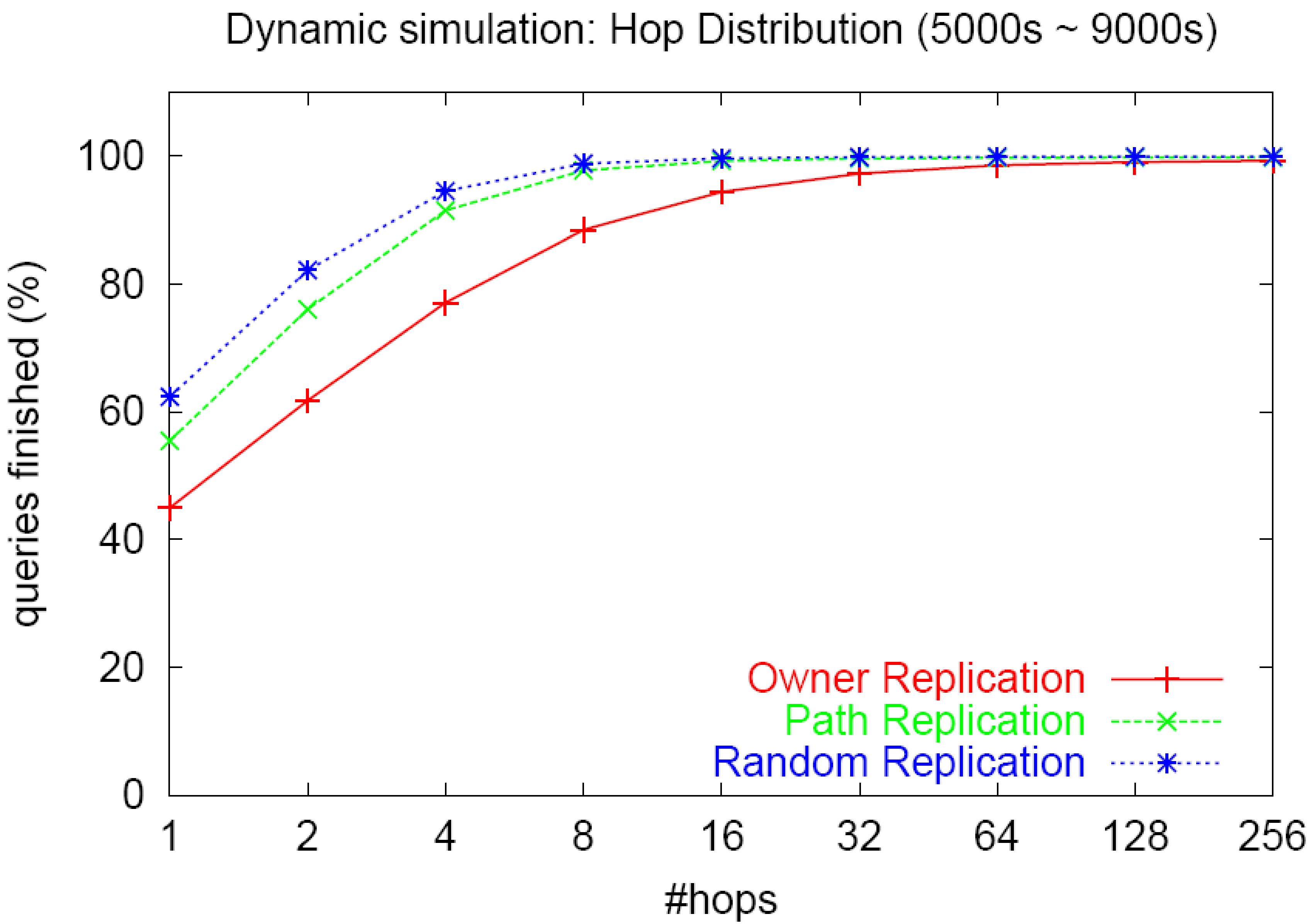
Achieving good replication

- Owner replication
  - Results in proportional replication
- Path replication
  - Results in square root replication
- Random replication
  - Same as path replication, only using the given number of random nodes, not the path
- Removal strategy
  - Must be random or based on fixed time

Achieved replication distribution



Performance of different replications



GIA: motivation

- Unstructured networks are good
  - Fault tolerant, robust
  - Support arbitrary keyword queries
- Flooding is not good
- Random walks are better but not perfect
  - They are too blind without some help, such as biased walk (see scale-free nets)
  - Load balancing can be a problem esp in heterogeneous networks under high query load

GIA motivation

- Major problem seems to be poor load balancing
- So let us now make they query “throughput” of the system the main evaluation criterion
  - Load balancing is the major thing to optimize here
- We know networks are heterogeneous
- This means we must make sure nodes process queries proportional to their bandwidth
  - Topology: Let's adapt the topology so that all nodes have the right amount of neighbors
  - Flow control: Let's cleverly limit the number of forwarded queries to neighbors

Components

- One hop replication
  - Pointers to objects are replicated on neighbors
- Topology adaptation
  - Put most nodes within short reach of high capacity nodes
- Flow control
- Search protocol
  - Random walk biased towards high capacity (not high degree) nodes
  - Note that without topology adaptation, capacity and degree do not necessarily correlate

Topology adaptation

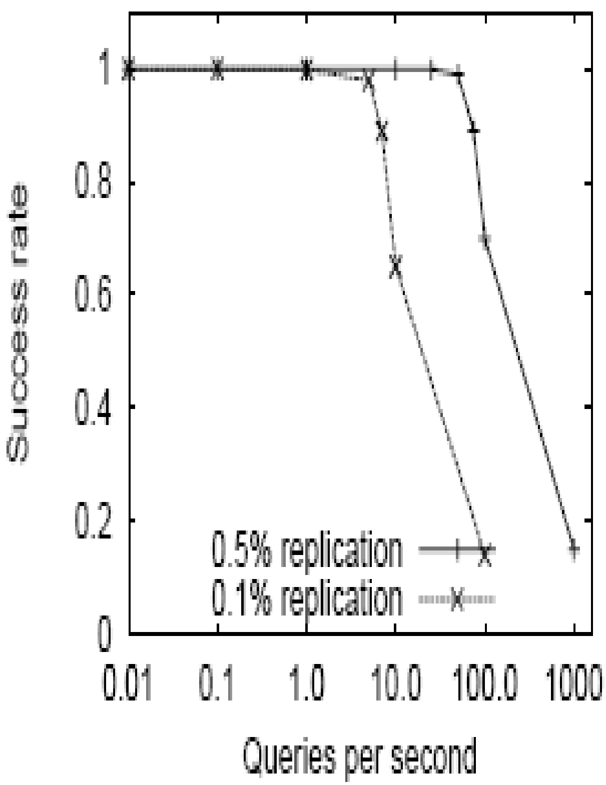
- All nodes keep trying to improve their neighbor set until possible (satisfaction function)
  - Candidates in “host cache”
  - Using candidates, we continuously want to
    - **increase the capacity of our neighbors**
    - **decrease the number of neighbors of our neighbors**
- Topology is undirected: handshake mechanism
  - We need to ask nodes to accept us as a neighbor
  - They might need to drop neighbors

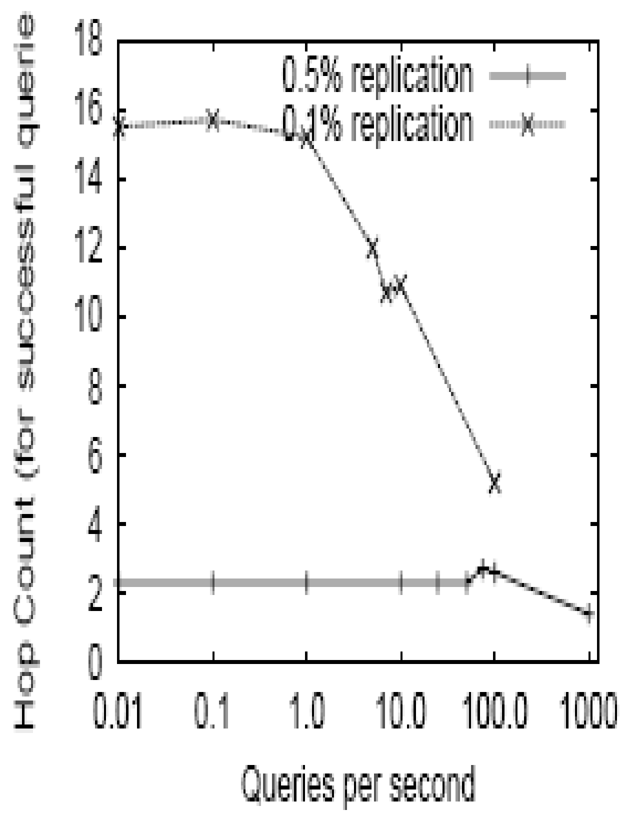
Flow control

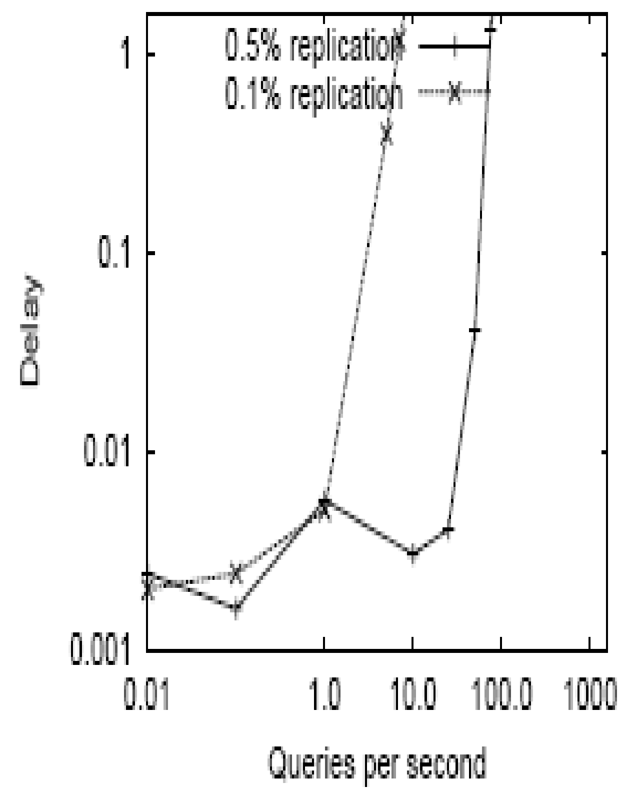
- Nodes assign tokens to their neighbors proportional to their capacity
- More tokens are assigned to higher capacity nodes (incentive to be honest when reporting capacity)
- Search protocol
  - Picks highest capacity neighbor to forward query, for which there is a token available

Performance measures

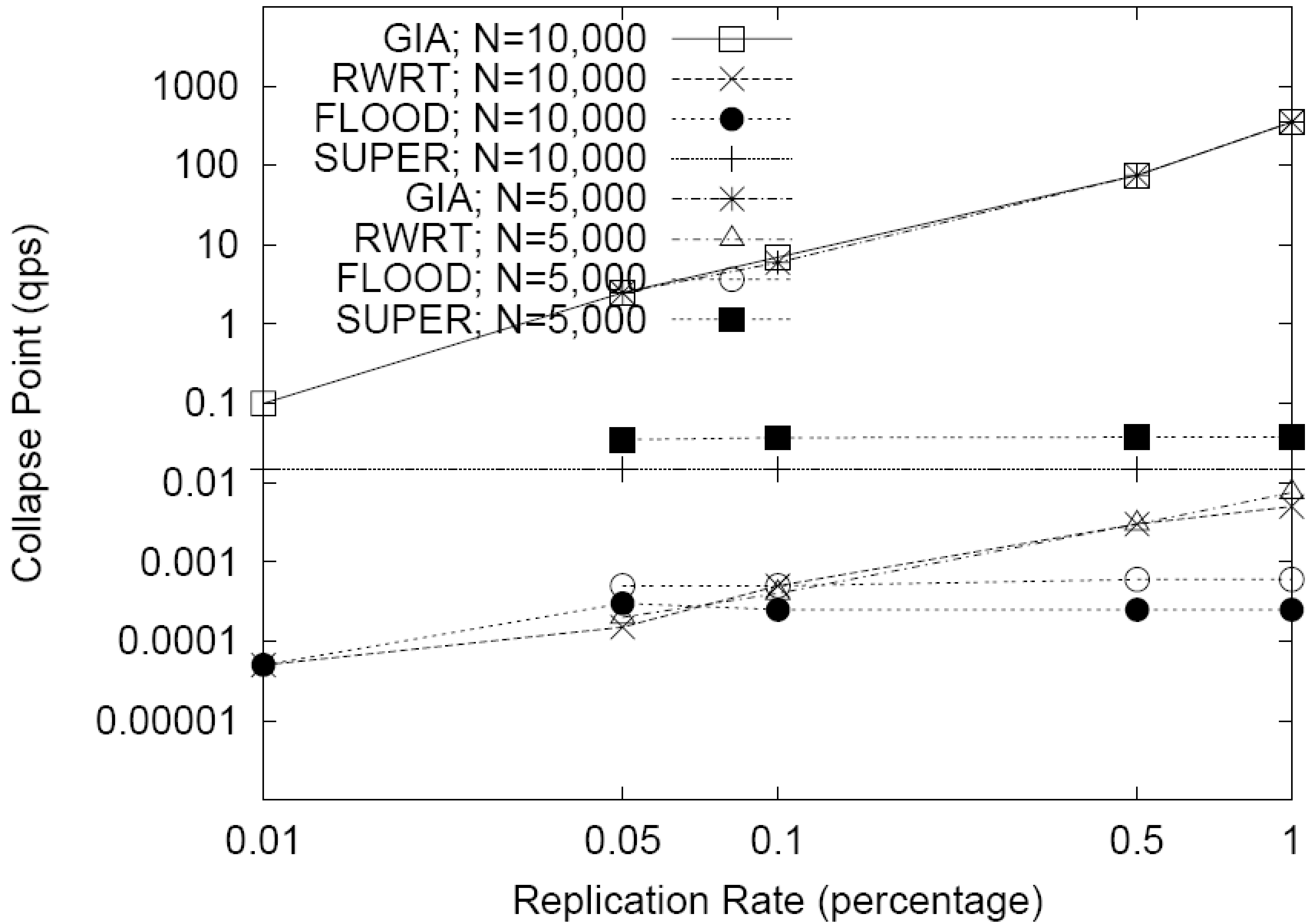
- Main focus is system load, and metrics as a function of that
- Behavior is captured by “collapse point”: success rate passes 90%



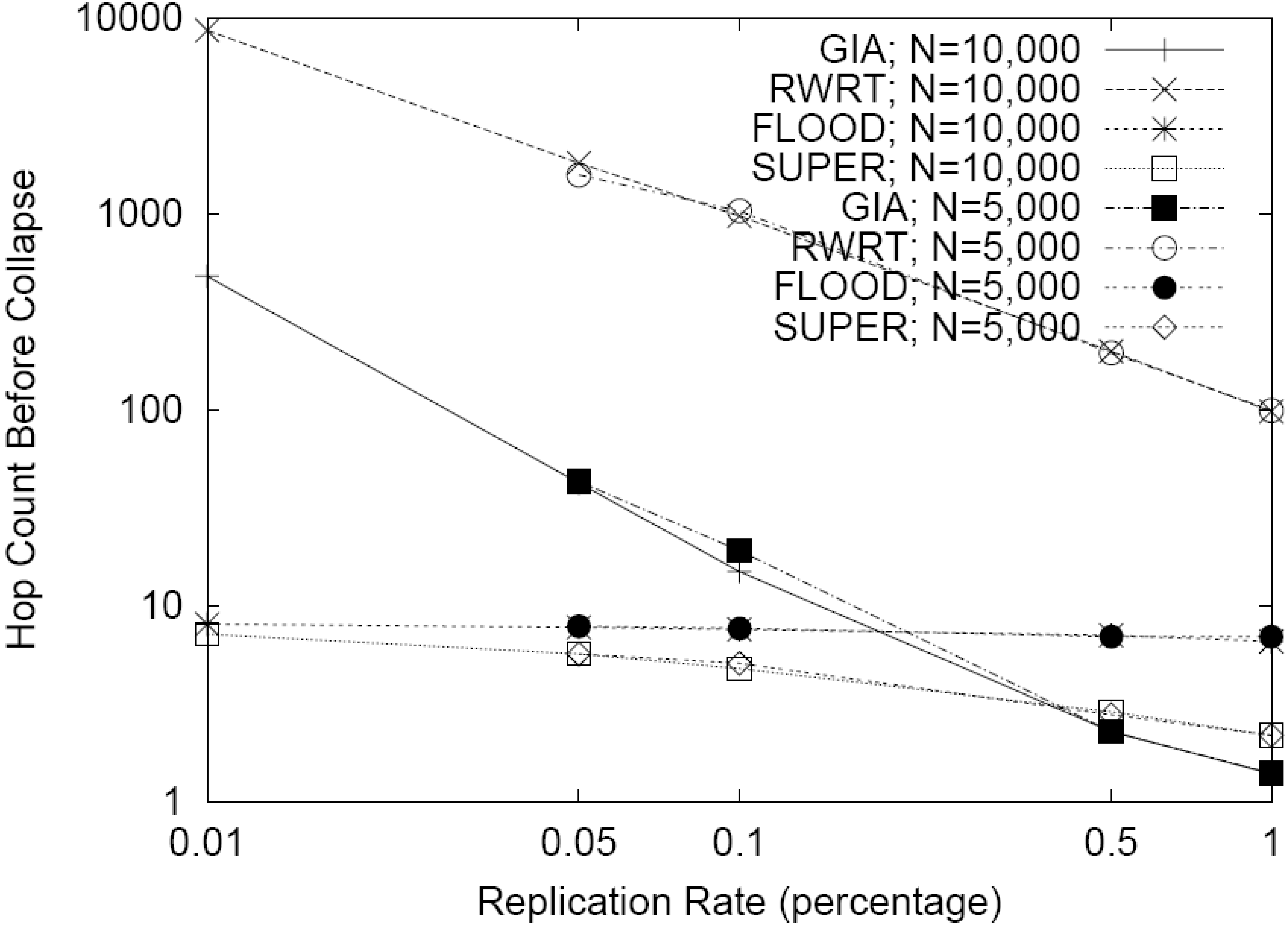




Results: collapse points



Results: hop count before collapse



Factor analysis of components

- 10,000 nodes, 0.1% replication
- Only all components together achieve the desired effect

Algorithm	Collapse Point	Hop-count
GIA	7	15.0
GIA – OHR	0.004	8570
GIA – BIAS	6	24.0
GIA – TADAPT	0.2	133.7
GIA – FLWCTL	2	15.1

Algorithm	Collapse Point	Hop-count
RWRT	0.0005	978
RWRT + OHR	0.005	134
RWRT + BIAS	0.0015	997
RWRT + TADAPT	0.001	1129
RWRT + FLWCTL	0.0006	957

Summary

- Major components are
  - Search algorithm
  - Overlay topology
  - Replication strategies (pointer and object)
  - Flow control
- All of these can (and should) be adapted cleverly!
- At least topology and replication can be emergent as well (that is, influenced by aggregate user behavior)
- Problem of poor performance on rare files still exists

Some refs

- Papers this presentation used material from

  - Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, Nick Lanham, and Scott Shenker. Making gnutella-like p2p systems scalable. In Proceedings of ACM SIGCOMM 2003, pages 407–418, 2003.
  - Qin Lv, Pei Cao, Edith Cohen, Kai Li, and Scott Shenker. Search and replication in unstructured peer-to-peer networks. In Proceedings of the 16th ACM International Conference on Supercomputing (ICS'02), 2002.
  - Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Mapping the gnutella network. IEEE Internet Computing, 6(1):50–57, 2002. (doi:10.1109/4236.978369)
  - Lada A. Adamic, Rajan M. Lukose, Amit R. Puniyani, and Bernardo A. Huberman. Search in power-law networks. Physical Review E, 64:046135, 2001.

- The course website
  - http://www.inf.u-szeged.hu/~jelasity/p2p/