Unstructured **P2P** Networks

Niels Olof Bouvin

Challenge

- Research Question: How can we efficiently (or just *reasonably*) search for and retrieve information in P2P networks?
- P2P networks are generally far too unstructured and transient to be crawled and indexed properly
- Most information would be *outdated* or *unavailable* by the time you were done…

Efficient Searching in Unstructured P2P Networks

- **Initial state:**
  - No knowledge about neither structure nor location
- **Goals for efficient searching**
  - Good results ~ find it if it's there
  - Good performance ~ low latency
  - Good network behaviour
    - no overloaded peers
    - no duplicate traffic

Overview

- **Characteristics of file sharing networks**
- The shape of the overlay network
- Alternatives to flooding
- Adaptive probabilistic search
- Super node architectures
- Gia
- Summary

(How) Unstructured P2P Networks?

- **These P2P networks have no fixed topology**
  - peers come and leave as they please (high "churn" rate)
  - peers may often have to self-determine how to connect to the network
  - all decisions must be taken locally and ad-hoc
- **Data distribution**
  - not only do the peers connect/leave arbitrarily, the data they hold is also unknown in advance
  - possible strength: redundancy of data (many copies across the network)

Background: Study of Unstructured P2P Networks

- (Saroin, Gummadi & Gribble 2002)
- Dimensions:
  - network bottlenecks (peers have different bandwidth available)
  - network latency
  - churn
  - level of sharing and co-operation
- Systems studied:
  - Napster & Gnutella

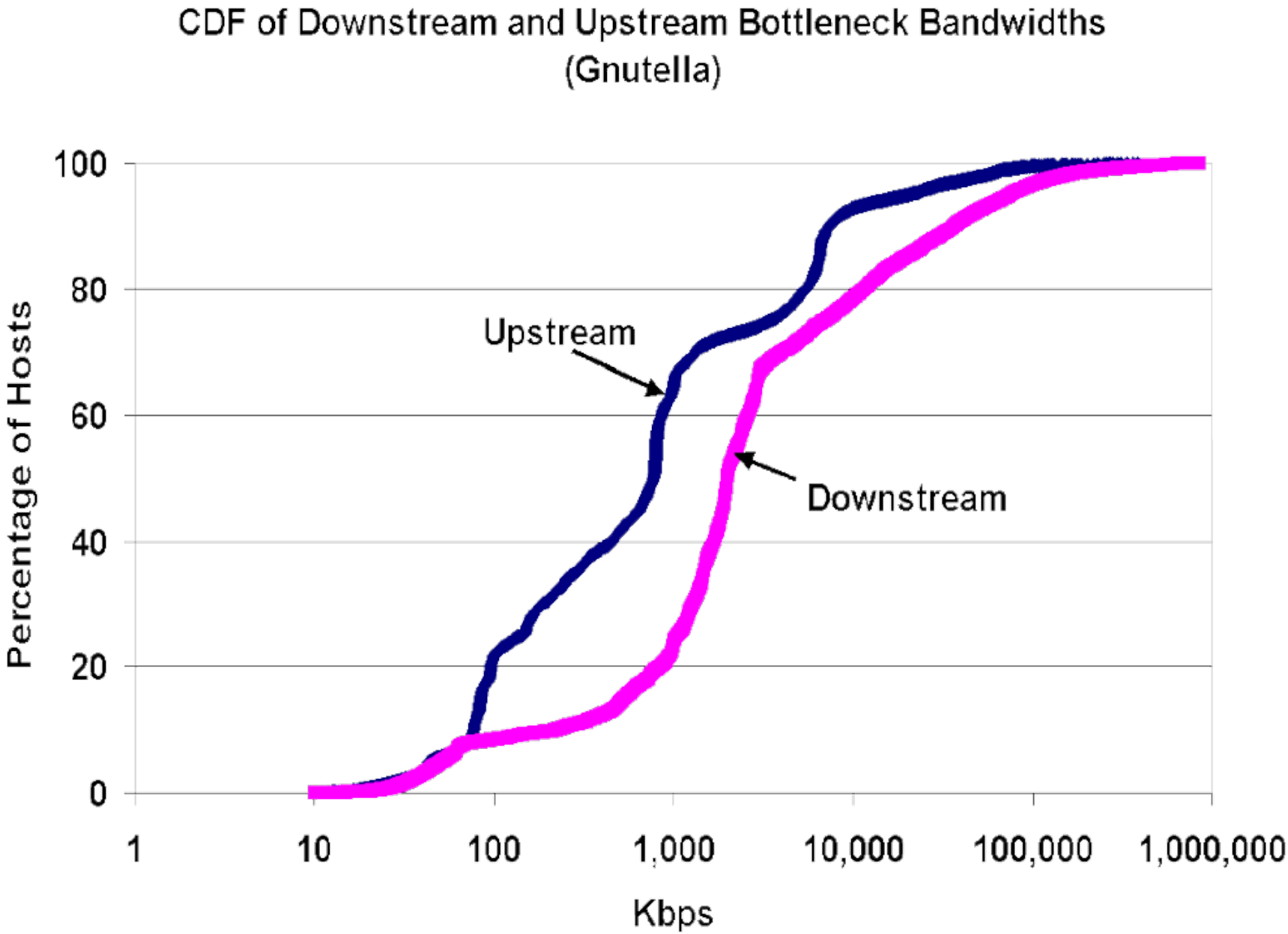
Research Questions: Bandwidth

- **Are peers sufficiently well-connected to serve successfully as servers?**
  - high bandwidth (upstream and downstream)
  - low latency
- **Badly connected peers are easily saturated**
  - this can potentially slow many down

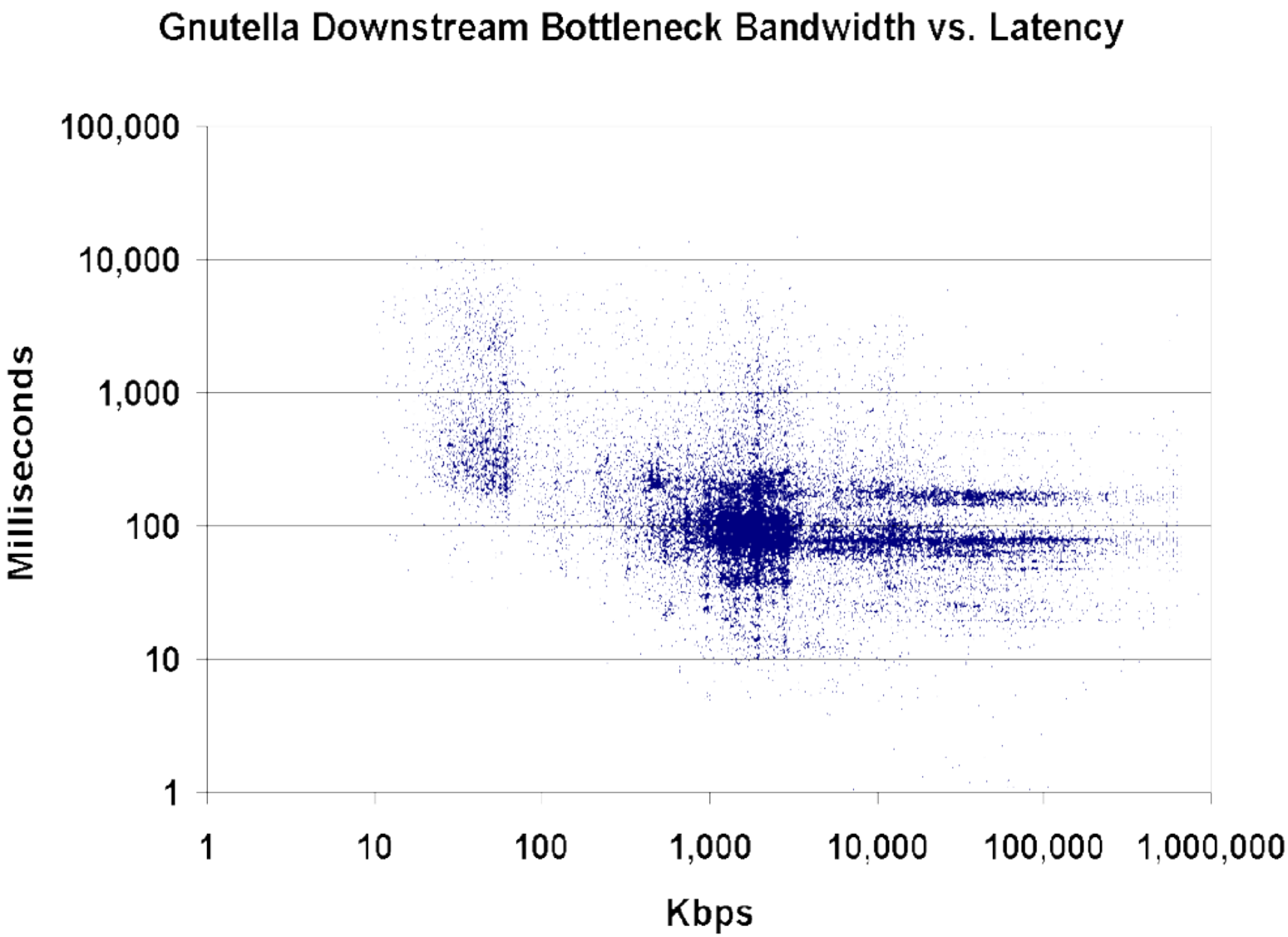
Method

- **Special purpose crawlers tracked users**
- **Napster**
  - no access to central databases, so data collected by searching for popular music
- **Gnutella**
  - aggressive ping/pong collected over time, charting an estimated 25-50% of the Gnutella network
- **Crawling apparently perceived as DoS attack led to premature end of Napster scanning**

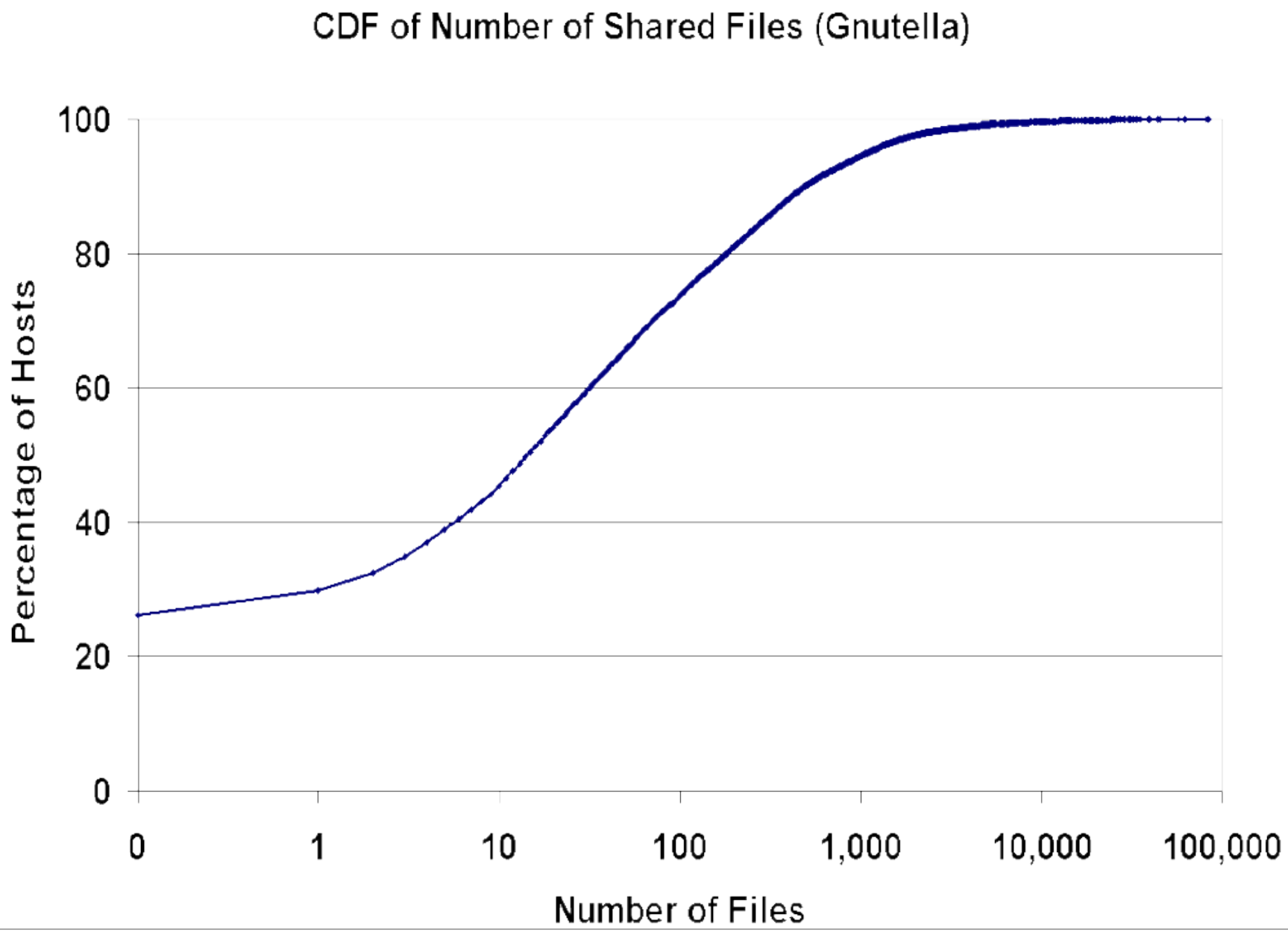
Asymmetry of Bandwidth



Bandwidth/Latency



Free-riding


CDF of Number of Shared Files (Gnutella)
Percentage of Hosts
100
80
60
40
20
0
0
1
10
100
1,000
10,000
100,000
Number of Files

Summary of Findings

- **Peers are heterogeneous**
  - different bandwidth
  - different latency
- **Users are lazy**
  - different contributions to the network
- **Therefore P2P systems should**
  - delegate tasks according to ability
  - reward altruistic behaviour

Overview

- Characteristics of file sharing networks
- **The shape of the overlay network**
- **Alternatives to flooding**
- **Adaptive probabilistic search**
- **Super node architectures**
- **Gia**
- **Summary**

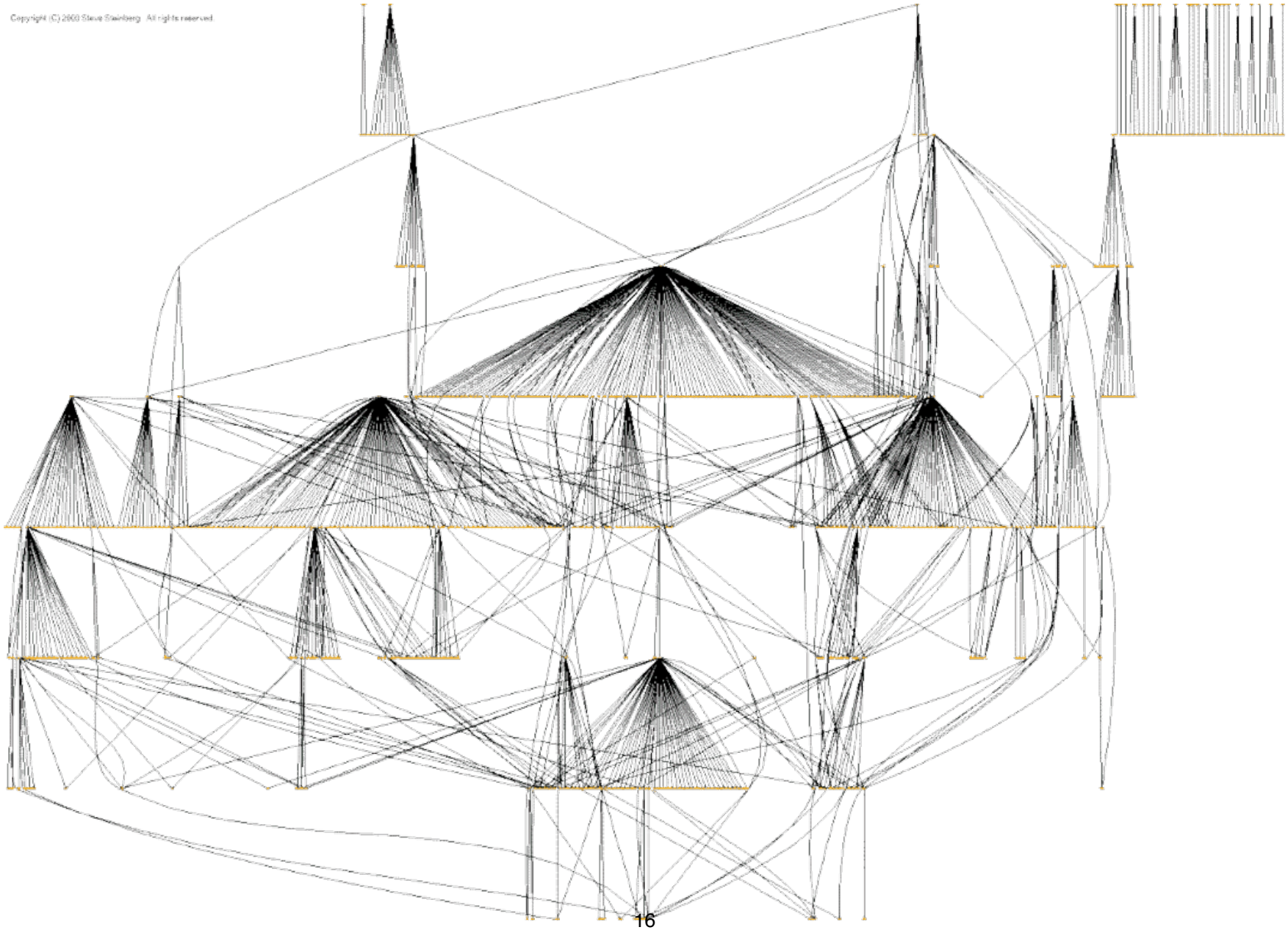
The Structure of an Overlay Network

- Gnutella network graphs are not random
- Gnutella networks have been shown to follow a power-law distribution
  - peers prefer to connect to well-connected peers
- Such networks are highly resilient against random breakdown

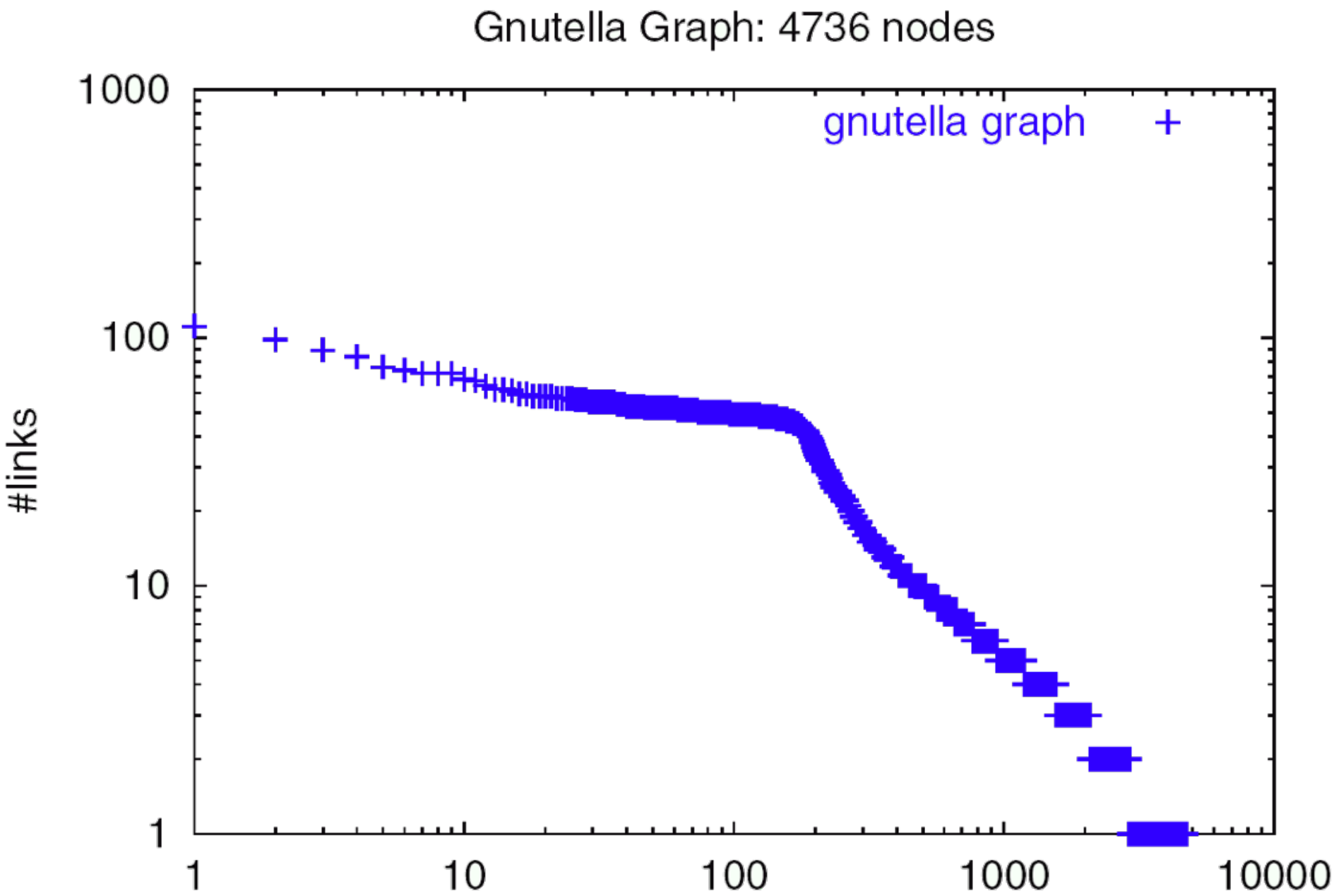
Power-Law Networks

- **A power-law network is a network, where the probability that a node has k neighbours is** $P(k) \sim k^{-\gamma}$
- **Thus,**
  - a few peers have many neighbours
  - most peers have few neighbours
- **Power-law is found in many networks, e.g., WWW**

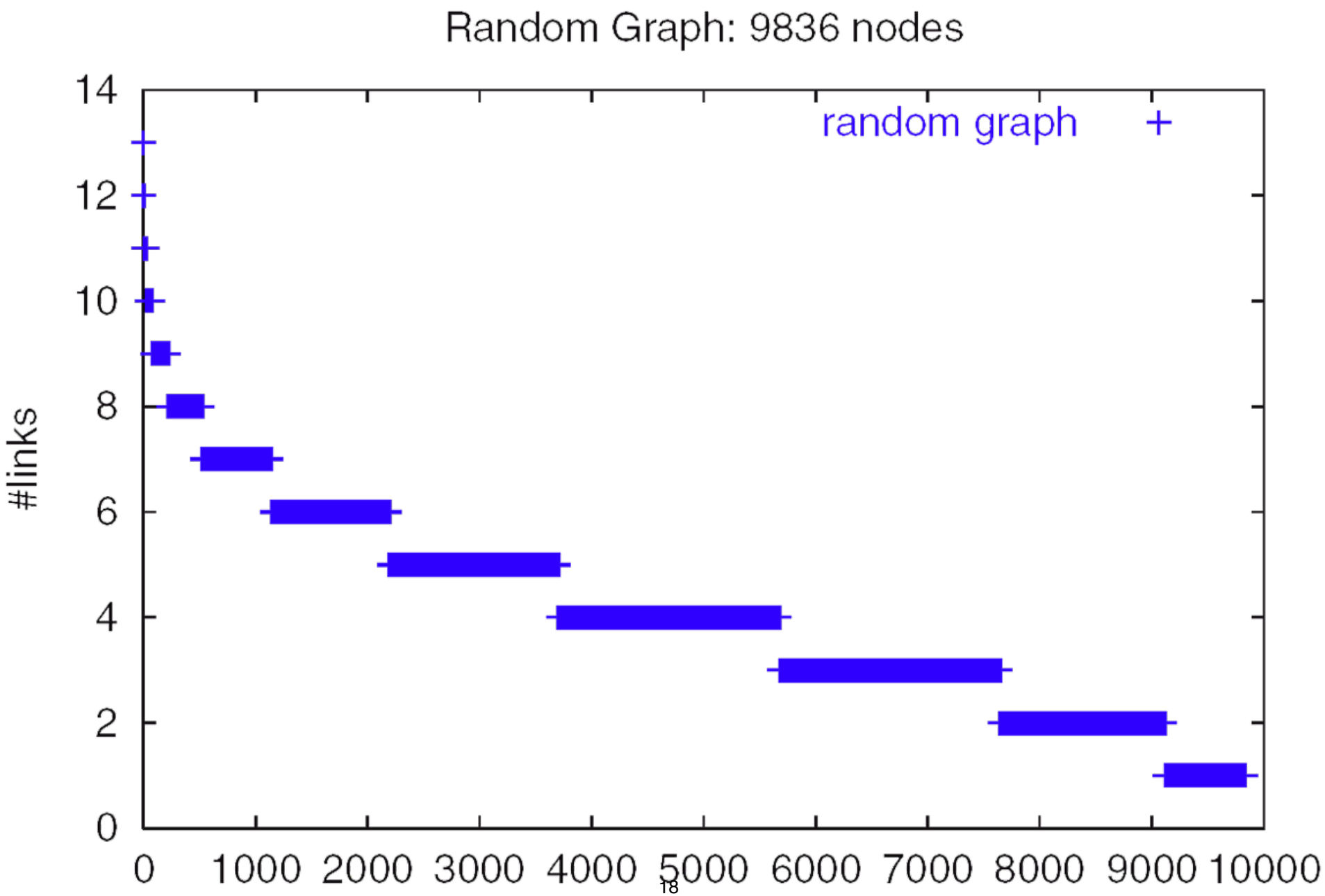
A Gnutella Network Graph



Gnutella–Node Degree/#Links



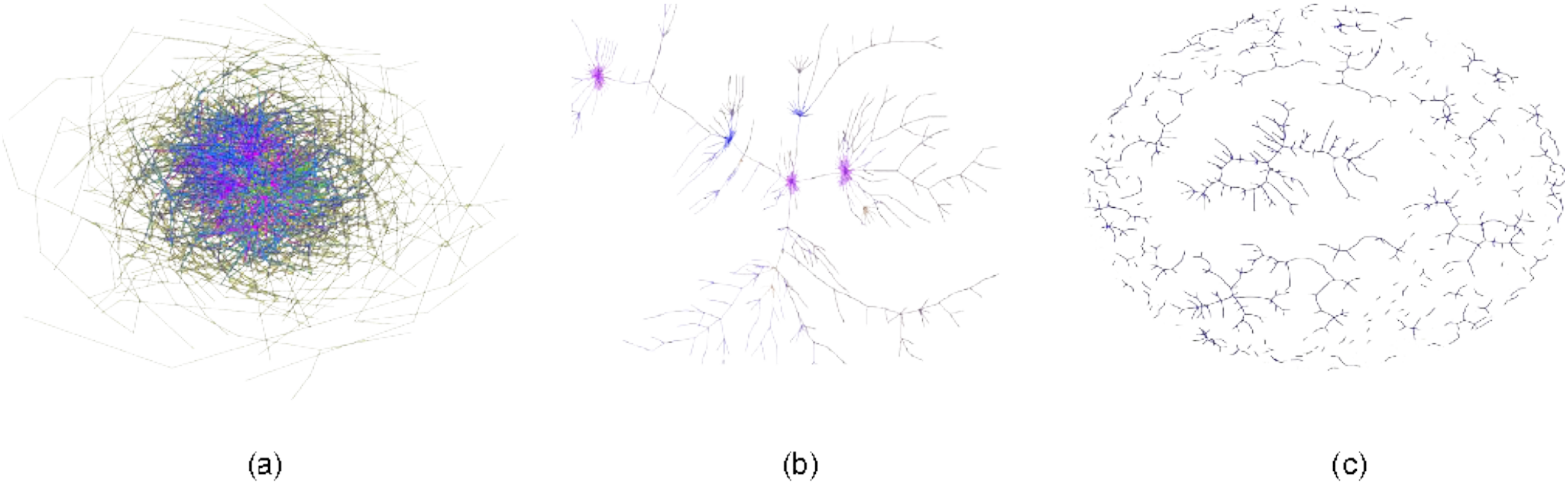
Random–Node Degree/#Links



Power-Law Overlay Networks in P2P

- **You are never far from a well-connected peer**
  - so you are never far from the rest of the network
- **If attacks and other mishaps are *random*, they will most likely affect the less-connected peers without too much inconvenience for the rest**
- **If attacks are *targeted*, the entire network can be taken out**

Random & Targeted Attacks against Gnutella Graph



(a) original graph; (b) random attack; (c) targeted attack

Overview

- Characteristics of file sharing networks
- The shape of the overlay network
- **Alternatives to flooding**
- **Adaptive probabilistic search**
- **Super node architectures**
- **Gia**
- **Summary**

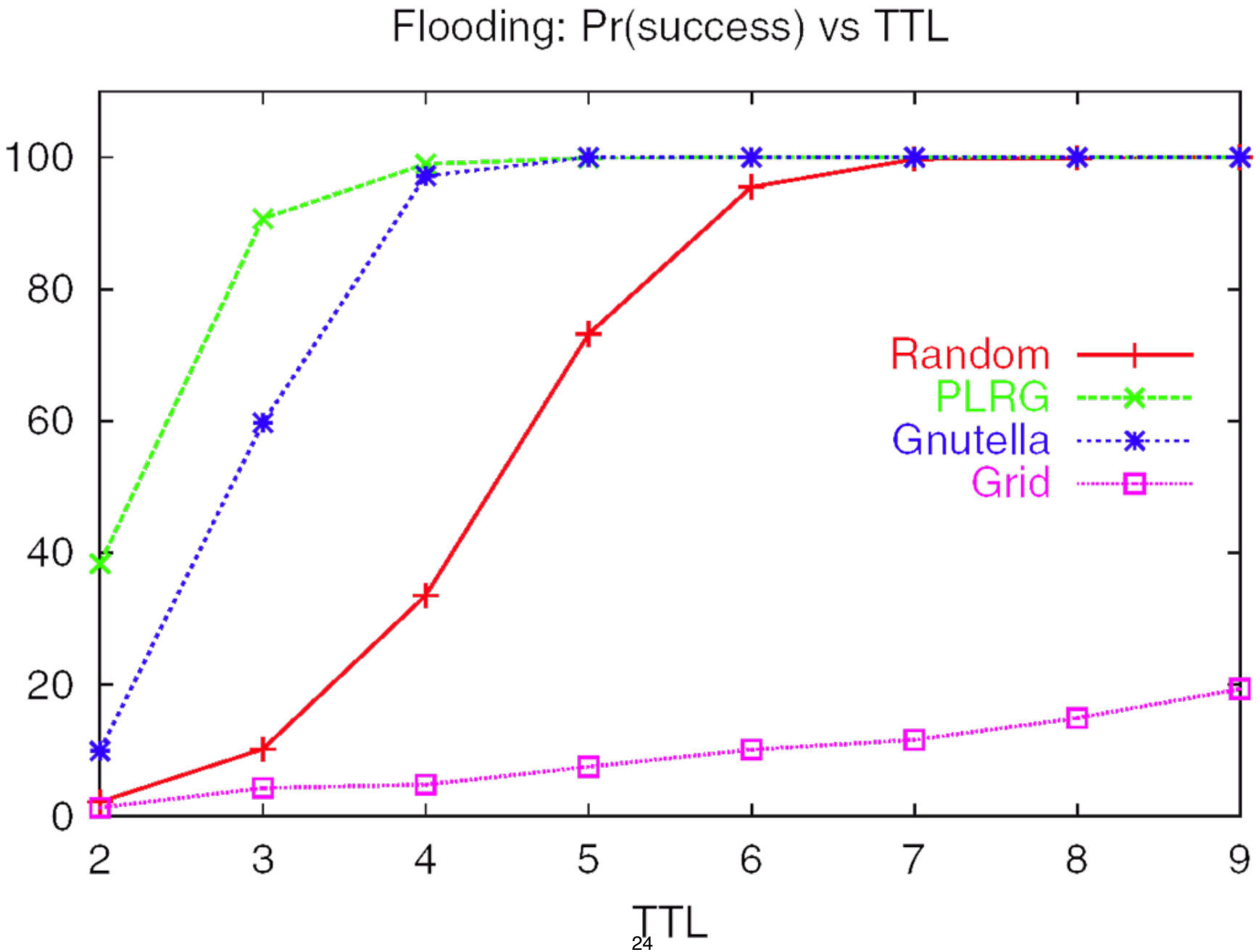
k-Walker Search in Unstructured P2P Networks

- **Aim**
  - Investigate how bad flooding is and demonstrate superior searching methods
  - Investigate how the distribution of replicates affects searching (but we will not go into that)

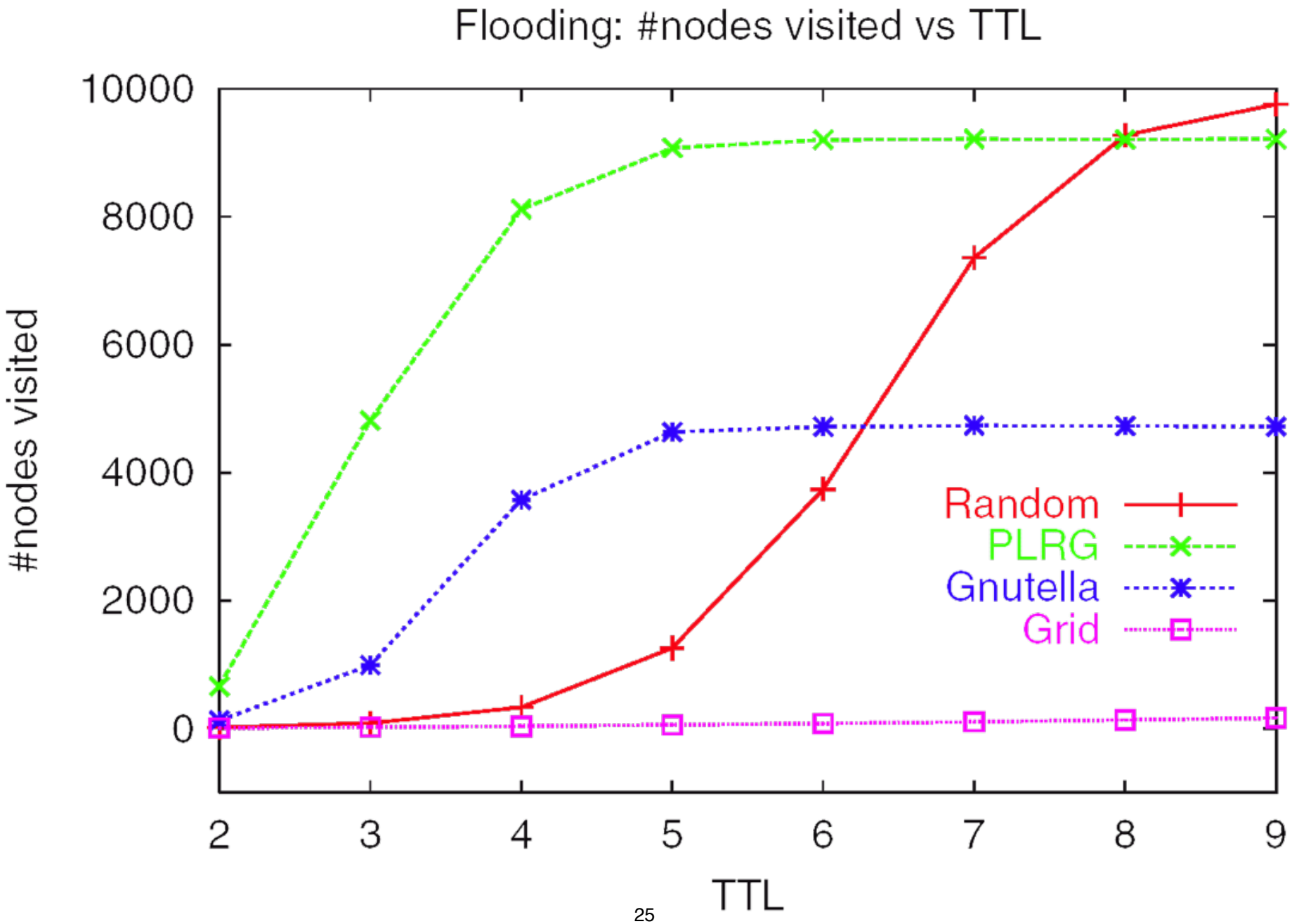
Time to Live Considered Harmful

- **Naïve Gnutella searching consists of spreading queries by flooding until TTL is exhausted**
  - if a hit is found, the flooding continues regardless everywhere else
  - if a hit is not found, a tremendous amount of messages have been sent for no good reason
  - high TTL generates a lot of traffic
  - low TTL may not locate the desired resource
- **But, as a query quickly covers a large portion of the network neighbourhood, the delay between issuing a query and receiving results is quite low**

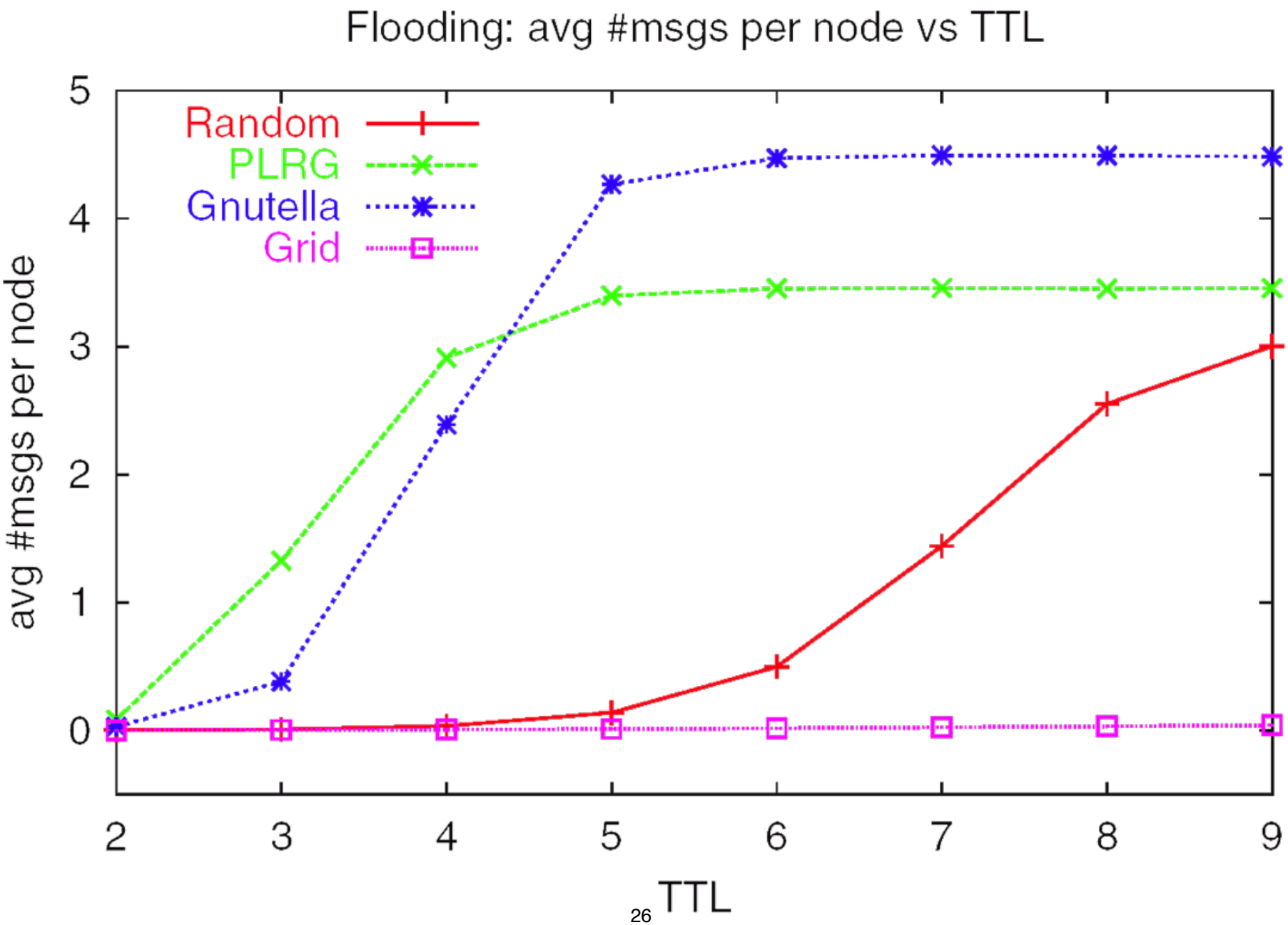
Successful Search/TTL



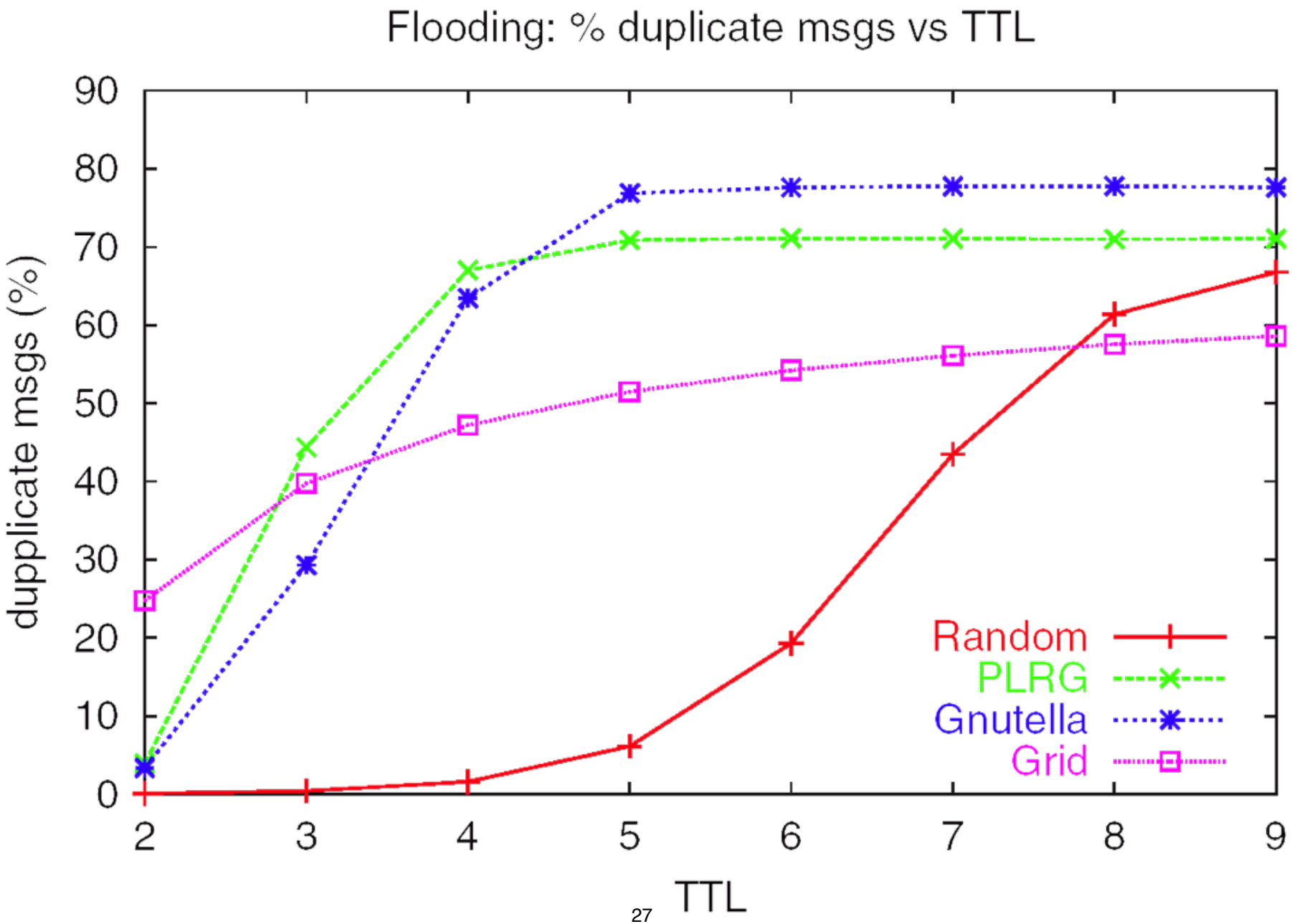
#Nodes Visited/TTL



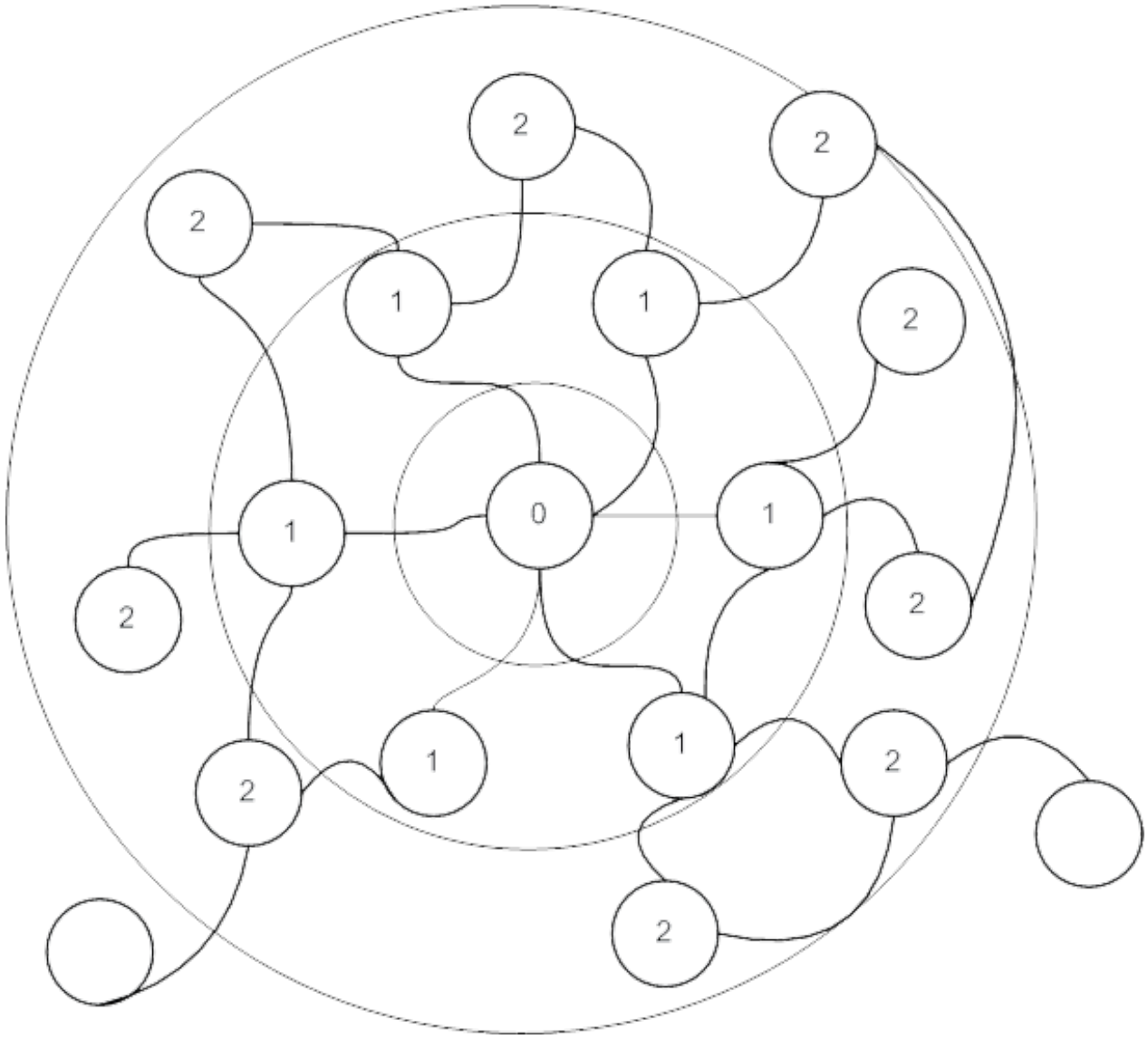
Average #Messages per Node/TTL



%Duplicate Messages/TTL



Flooding Alternative: Expanding Ring

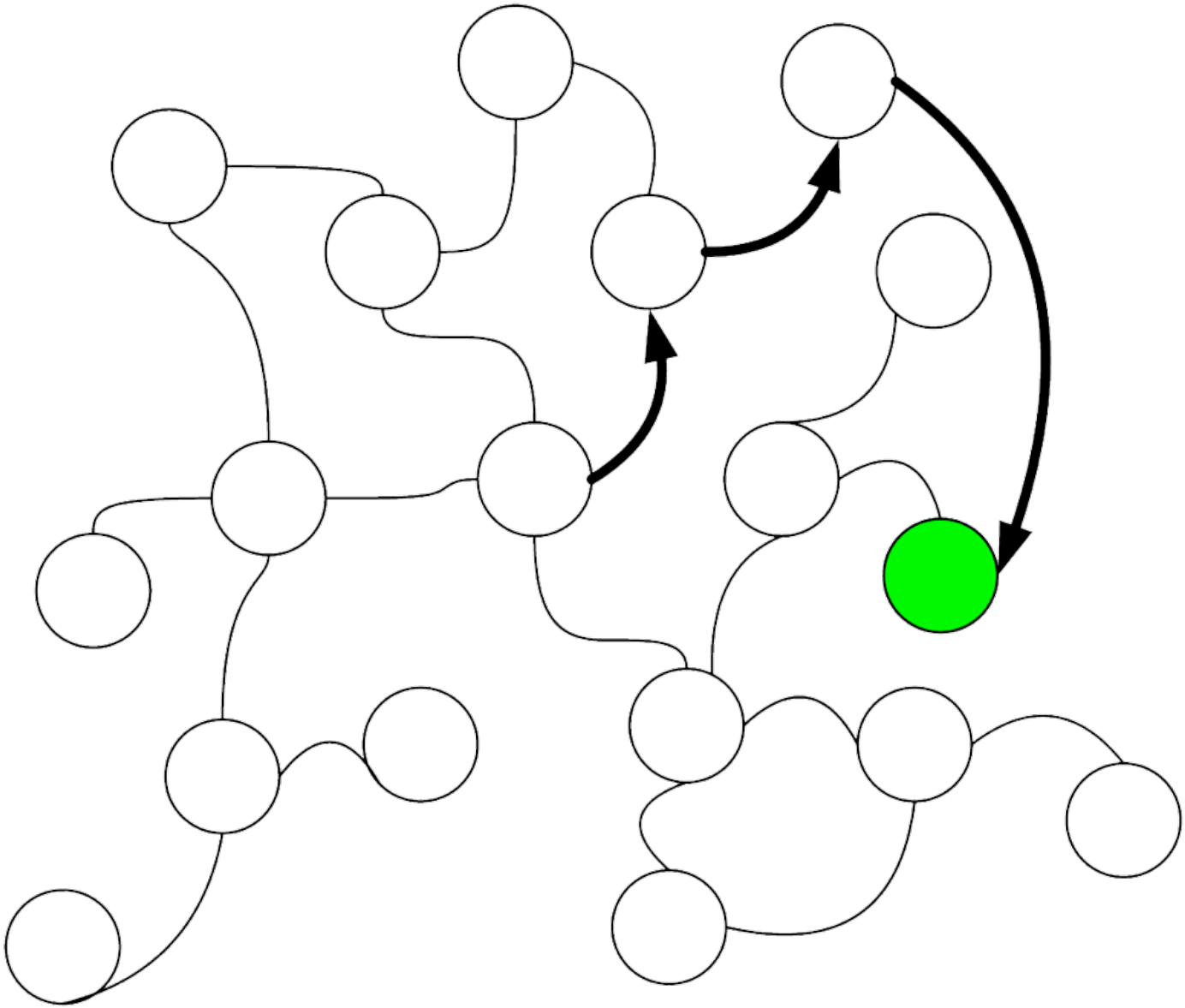


Start with small values of TTL, and increase TTL until sufficient number of hits are found

Expanding Ring

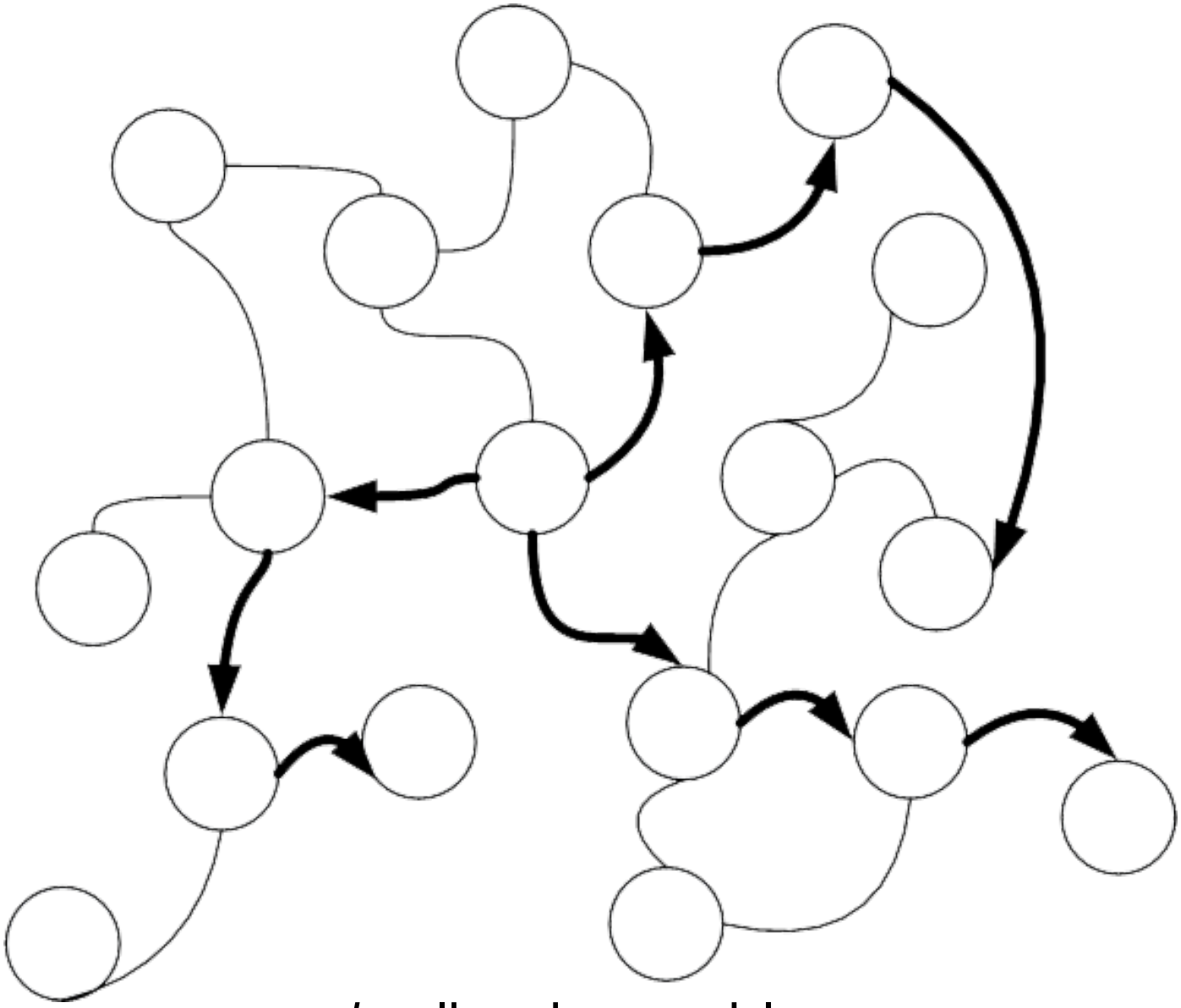
- **Advantages**
  - ultimately as successful as ordinary flooding
  - if a resource is nearby, it is located at a lower overall cost
- **Disadvantages**
  - if the resource is not found, *more* messages are generated than ordinary flooding!
  - successive searches mean longer delays

Random Walker



Depth-first search: A query transverses the network randomly until a match is found

k Random Walkers



k walkers decrease delay

Random Walkers

- **Advantage**
  - much more efficient in term of overall traffic
- **Disadvantage**
  - longer delays
- **Typical configuration**
  - TTL = 1024; $32 \leq k \leq 64$
- **Variations**
  - walker checks periodically source for sufficient success (every fourth step)
  - nodes maintain state and do not forward the same query to the same neighbour twice

Results

distribution model		50 % (queries for hot objects)			
query/replication	metrics	flood	ring	check	state
Uniform / Uniform	#hops	2.39	3.40	7.30	6.11
	#msgs per node	4.162	0.369	0.051	0.045
	#nodes visited	4556	933	141	151
	peak #msgs	64.9	6.4	1.3	1.2
Zipf-like / Proportional	#hops	1.60	2.18	1.66	1.66
	#msgs per node	2.961	0.109	0.021	0.021
	#nodes visited	3725	357	49	60
	peak #msgs	43.8	2.0	0.7	0.8

Gnutella graph

Conclusions

- **Results**
  - Random walkers scale much better than flooding – especially with regards to message duplication
  - User perceived delays are increased
  - Blindly using TTL is inefficient – queries should check back periodically
- **However**
  - Simulation assumes a stable network
  - Content/traffic may not be Zipf-distributed after all (Gummadi et al. 2003)

Overview

- Characteristics of file sharing networks
- The shape of the overlay network
- Alternatives to flooding
- **Adaptive probabilistic search**
- **Super node architectures**
- **Gia**
- **Summary**

Adaptive Probabilistic Search

- **The nodes and walkers so far have been stupid**
  - (or rather not employing all available knowledge)
- **Lv et al.'s nodes only remembered where they sent walkers**
  - other than that, destinations were chosen at random
- **What if nodes made informed choices when sending walkers on their merry way?**
- **Danger: It takes extra work to maintain knowledge!**

Blind or Informed Search Methods

- **Blind methods**
  - No information is kept about the whereabouts of resources
  - Easy, but inefficient
  - No worse (or, just as bad) if churn is high
- **Informed methods**
  - Nodes maintain indices over known files and their location
  - More efficient, but indices must be kept updated
  - Expensive if the churn rate is high, as indices must be continually refreshed

Probabilistic Search

- **All nodes maintain indices for**
  - each resource requested (either locally or forwarded) per neighbour
  - a value reflecting the probability of a neighbour to be chosen for a future request for this object
- **Queries are performed with *k* walkers**
- **Each node forwards walkers based on the probability distribution in its index**
- **Walkers remember visited nodes (or nodes remembers routing)**
- **Duplicates are discarded**
- **Walkers terminate on success or TTL**

Adaptive Probabilistic Search?

- How are the probabilities established and maintained?
- We could back-trace along the search route and update every time a query terminates
- Thus, the network would adapt over time
- Might we save some network traffic by adjusting the probabilities in advance?

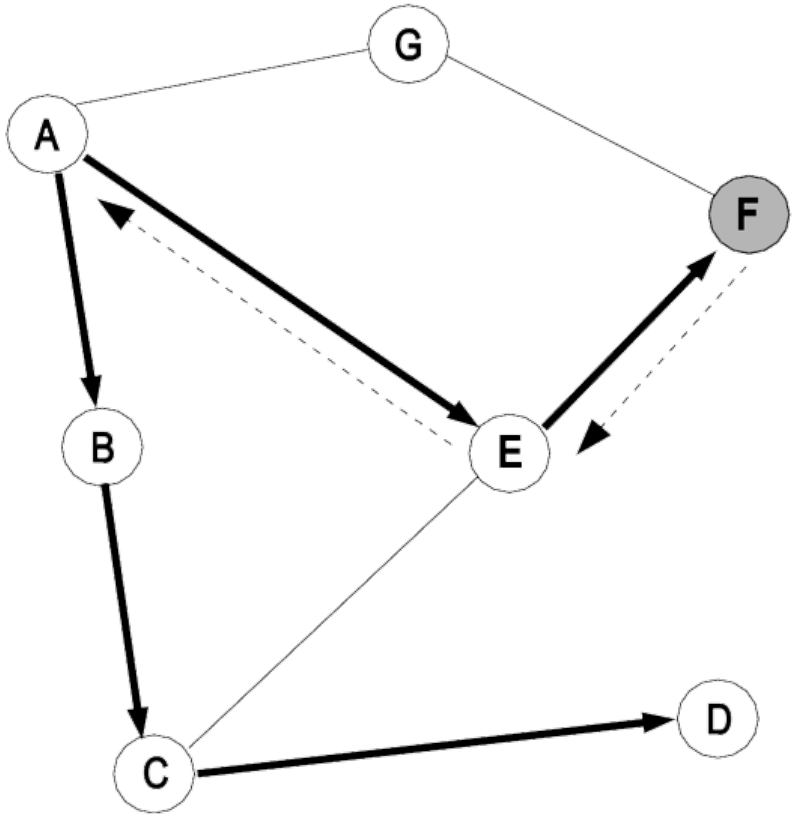
Pessimistic Approach

- **Requester assumes failure**
  - after choosing a destination node, the node decreases the matching probability for that node/resource combo
  - this is repeated by each node along the route
- **if the walker fails, nothing is done**
- **if the walker succeeds, it back-traces and the nodes' associated probabilities are increased**

Optimistic Approach

- **Requester assumes success**
  - after choosing a destination node, the node increases the matching probability for that node/resource combo
  - this is repeated by each node along the route
- **If the walker succeeds, do nothing**
- **If the walker fails, it back-traces and the nodes' associated probabilities are decreased**

APS – Pessimistic Example



Indices	Initially	After walkers finish	After the updates
A→B	30	20	20
B→C	30	20	20
C→D	30	20	20
A→E	30	20	40
E→F	30	20	40
A→G	30	30	30

Optimistic or Pessimistic?

- **The expensive part of APS is updating probabilities along the nodes – this should therefore be minimised**
  - If *more than half* of all queries are successful, you should be *optimistic*
  - If *less than half* of all queries are successful, you should be *pessimistic*
- **This knowledge becomes available over time, as all searches passing by can be monitored**

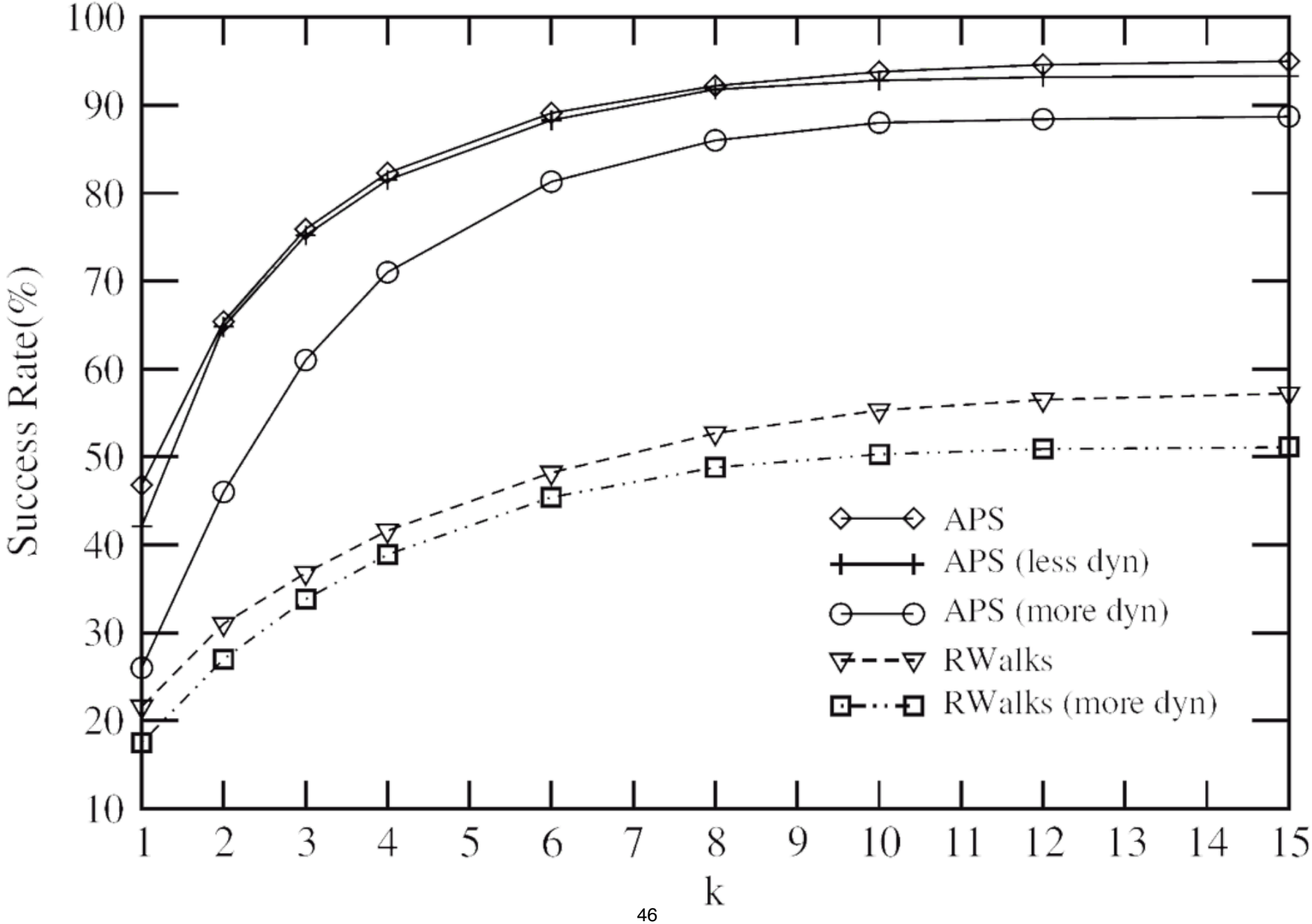
Algorithmic Variations

- **Swapping-APS (s-APS)**
  - be pessimistic for unpopular resources and optimistic for popular stuff
- **Weighted-APS (w-APS)**
  - adjust probabilities inverse proportional with distance to resource (if path is carried by walker)
  - prefer close resources over remote resources
- **These variations incur no increased network overhead**

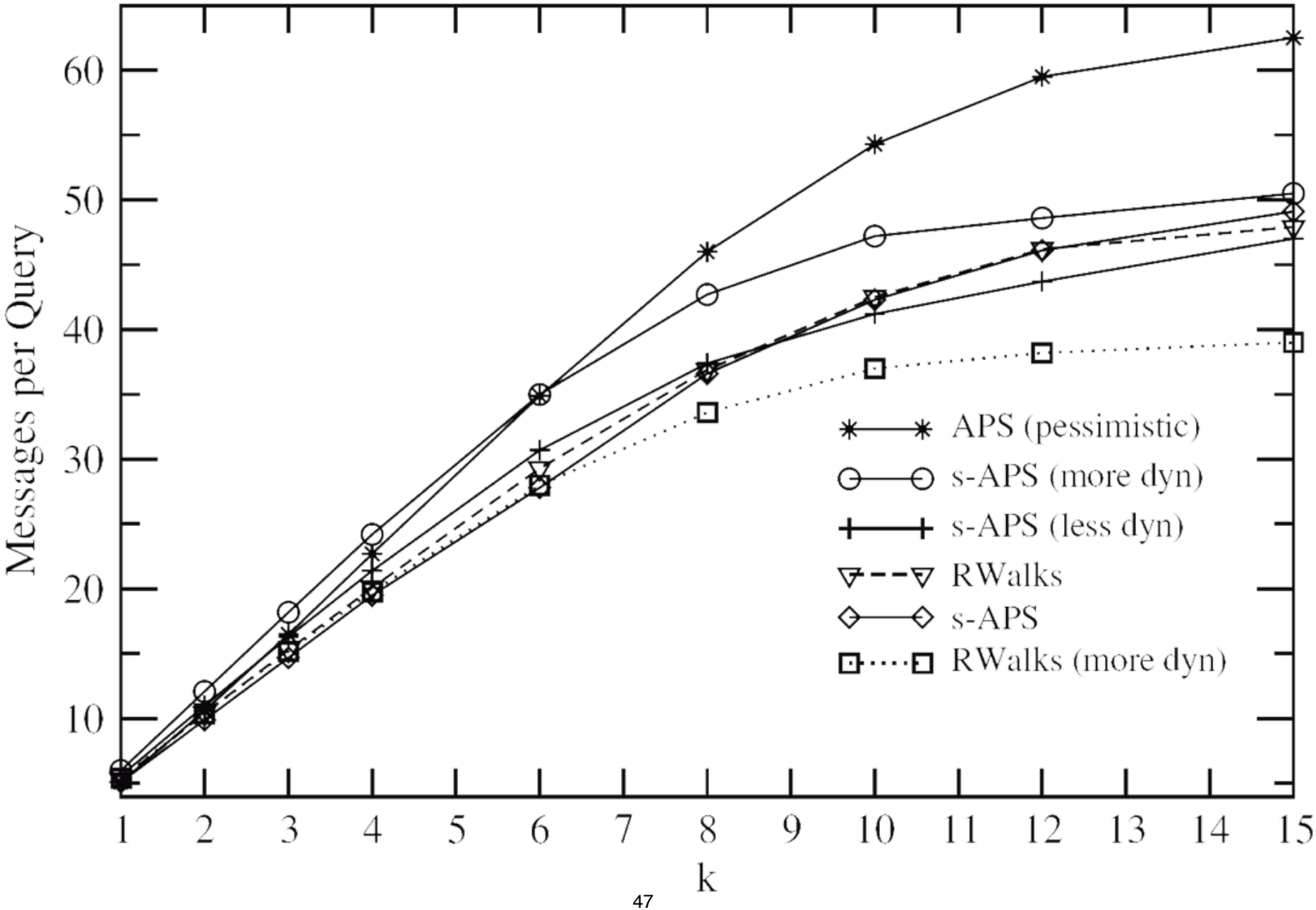
Simulation Configuration

- 10000 nodes
- 10 neighbours
- 1-15 walkers
- TTL = 6
- 100 objects
- Churn simulated with leaving and joining nodes

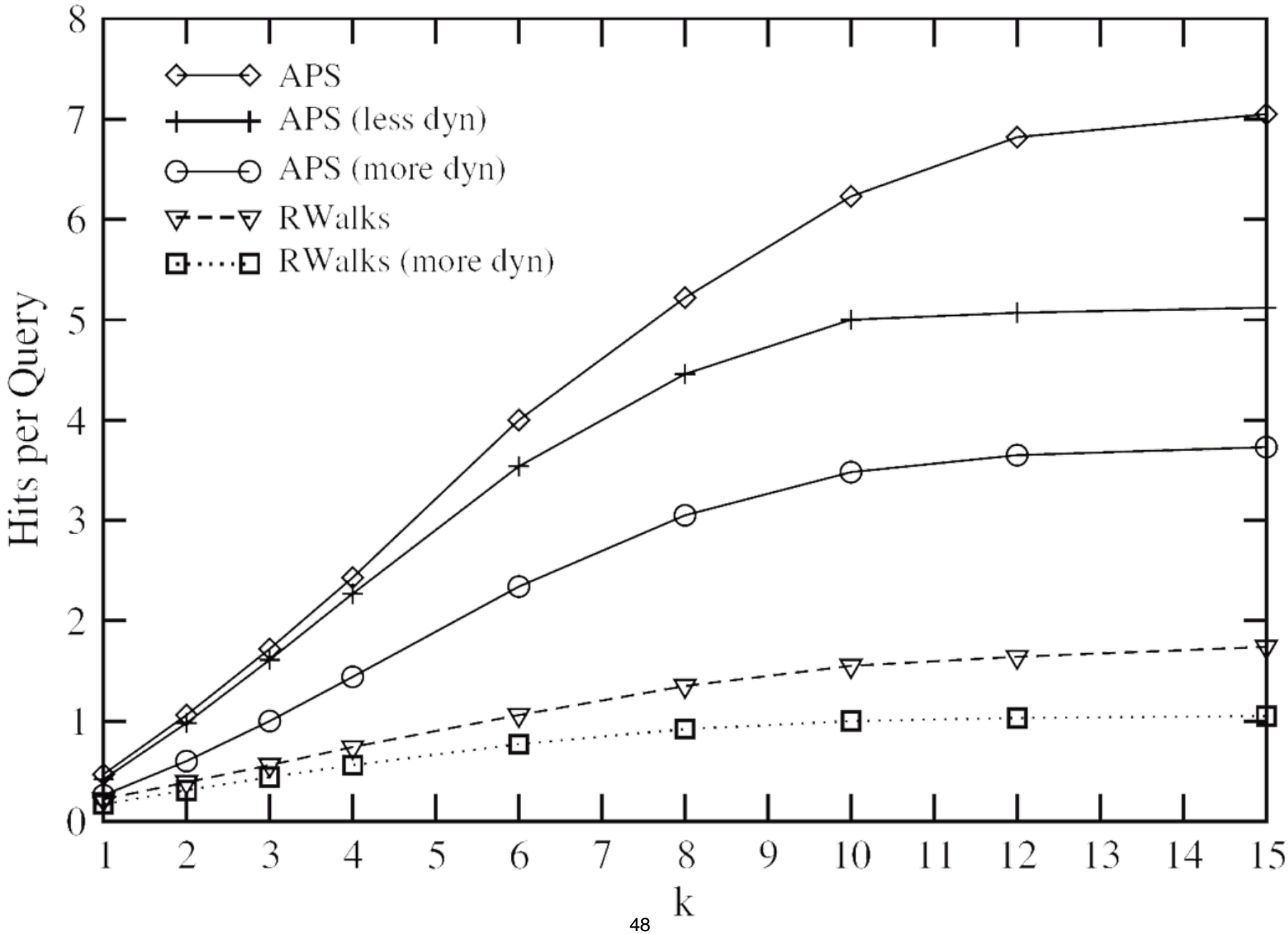
Results–Success Rate



Results–Messages per Query



Results–Hits per Query



Conclusions

- **Improved version of random walker**
  - higher success rate
  - generates more traffic
- **Resilient to transience**
  - adapts to new peers and their resources
  - new peers are included – it is, after all, probabilistic
  - new peers may not know a lot, but their neighbours will
- **Other possible directions for improvements:**
  - direct learning from neighbours worth the overhead?

Overview

- Characteristics of file sharing networks
- The shape of the overlay network
- Alternatives to flooding
- Adaptive probabilistic search
- **Super node architectures**
- **Gia**
- **Summary**

Super Nodes

- So far, you have seen unstructured P2P systems where peers are assumed to be, more or less, equal
- As shown by Saroin, Gummadi & Gribble, peers have different capabilities
  - some are much more capable than others
- Why not exploit these more powerful nodes?

The Problem with Flooding

- **Flooding is mainly disastrous because of high TTL**
- **Huge amounts of packets overwhelm the network and the weaker peers**
- **In Gnutella, this traffic consists of**
  - peer discovery (Ping and Pong)
  - queries (Query and QueryHit)

Roles in a Super Node Network

- **Super nodes**
  - have excellent connectivity
  - register leafs
  - discover other super nodes
  - index all resources held by leafs
  - answer and forward queries to other super nodes
- **Leafs**
  - less powerful peer
  - attach to a super node
  - download and upload resources

Conclusions

- **Pro**
  - reduced overall traffic
  - improved performance
  - better user experience (leafs' bandwidth used for transferring files)
- **Con**
  - if still based on flooding, just reducing a bad solution
    - ("New and Improved: Now with 80% **less** poison!")
  - how are super nodes elected, and what do they gain?
  - targeted attacks become a concern

Overview

- Characteristics of file sharing networks
- The shape of the overlay network
- Alternatives to flooding
- Adaptive probabilistic search
- Super node architectures
- **Gia**
- **Summary**

An Active Topology Adaption with Biased Random Walkers

- Gia: A system combining
  - ***topology adaption*** – peers should connect to strong and well-connected peers able to handle the traffic
  - ***active flow control*** – if a peer is overloaded it should be not bothered until it is ready again
  - ***one-hop replication*** of indices – every peer knows what its neighbours store
  - ***biased random walking*** – queries seek towards high capacity peers

Gia Terms

- **Capacity**
  - ability to handle messages/time – i.e., bandwidth, CPU power, storage capacity...
- **Satisfaction**
  - 0..1: degree to which a peer's own capacity is matched by the sum of its neighbours' capacities/degree

Topology Adaption

Let C_i represent capacity of node i
if $num\_nbrs_X + 1 \leq max\_nbrs$ **then** {we have room}
  ACCEPT Y; return

{we need to drop a neighbor}
$subset \leftarrow i \ \forall \ i \in nbrs_X$ such that $C_i \leq C_Y$
if no such neighbors exist **then**
  REJECT Y; return
candidate $Z \leftarrow$highest-degree neighbor from $subset$

if $(C_Y > max(C_i \ \forall \ i \in nbrs_X)\ )$ {Y has higher capacity}
or $(num\_nbrs_Z > num\_nbrs_Y + H)$ {Y has fewer nbrs}
 then
  DROP Z; ACCEPT Y
else
  REJECT Y

Add neighbours if we need them. Replace if there's someone better.

Only replace the well-connected.

Adaptive Flow-Control

- **Each peer sends tokens to its neighbours according to its (and their) capacity**
  - a peer must have a token from a neighbour in order to forward a query to that neighbour
  - if a peer is overloaded, it queues queries and reduces its token publication rate
  - tokens can be sent out separately or piggy-backed on other traffic
- **As tokens are assigned based on advertised capacity, it pays to advertise your true (high) capacity**
  - the opposite holds true in other systems – if you claim low capacity, you are not bothered by other users

One-Hop Indices Replication

- **All peers maintain indices over neighbours' resources**
  - thus all peers are able to answer queries for material held by their neighbours
  - this evens the load for peers with many resources
- **Query results contain pointers to the location of the resource – not the location of the index**
  - thus, duplicate query results are not created
- **But…**
  - what about popular content held by low ranking peers?

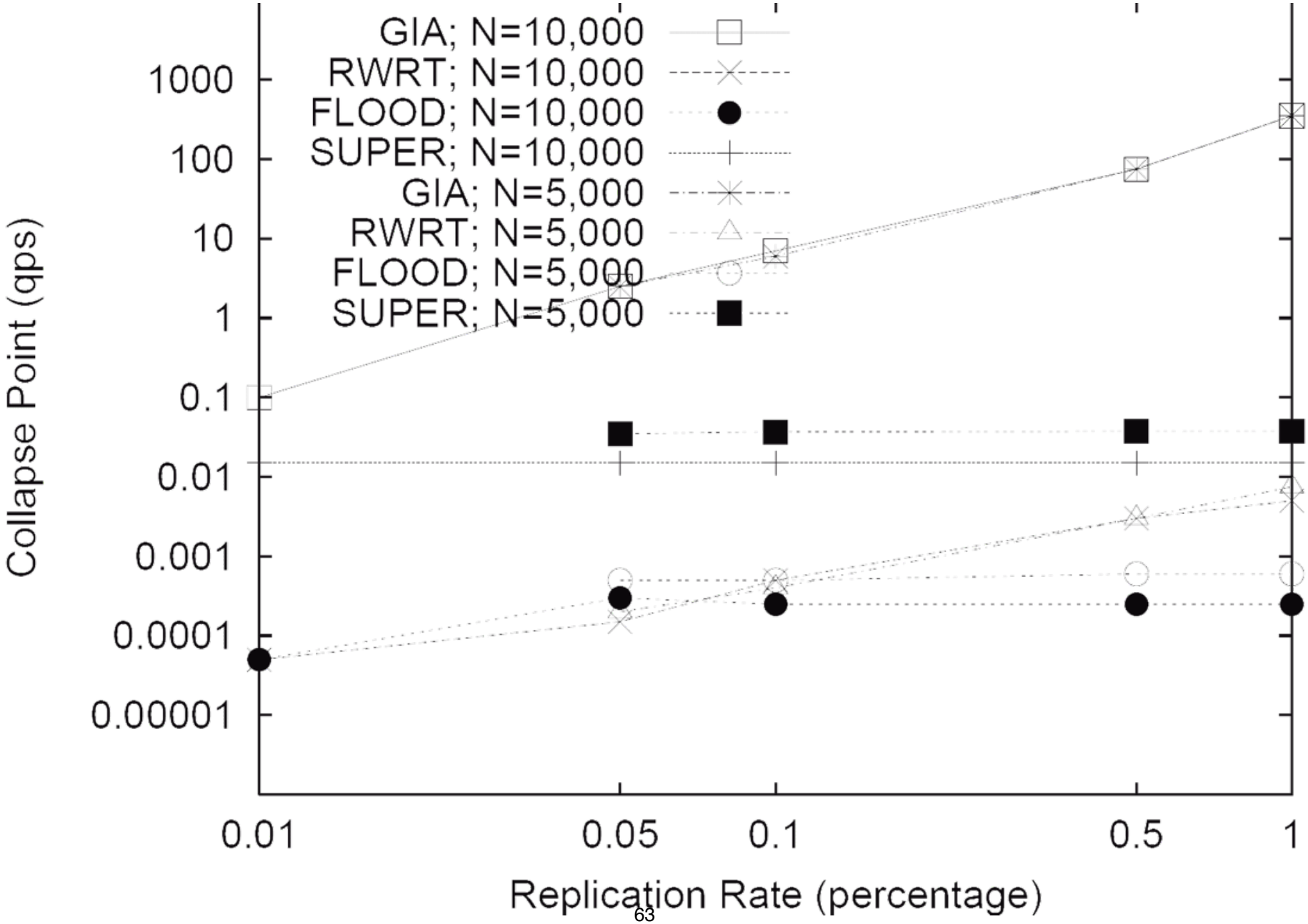
Biased Random Walker

- **Gia utilises a random walker algorithm where walkers are directed by the nodes towards the highest capacity neighbour it has tokens from**
  - queries are limited by TTL and MAX_RESPONSES
  - queries have GUIDs and are not forwarded to the same peer twice (unless the node is out of fresh neighbours)
  - queries return matches to source along query path
  - queries send keep-alive back to source (to handle network failures or rearrangements)

Gia Measurement Terms

- **Hop-count**
  - the number of hops needed to locate a resource
- **Collapse Point (CP)**
  - the point of traffic (queries) at a peer beyond which the success rate drops below 90% (because of traffic overload)
- **Hop-Count before Collapse (CP-HC)**
  - the average hop-count before the Collapse Point
  - simulation done on network with 10.000 nodes

Collapse Point



Conclusions

- A sophisticated system able to withstand high levels of traffic
- Designed with actual capacity in mind
- Many possibilities for fine-tuning and adjustment of the algorithms
- Also tested with actual computers!
- Not entirely unstructured, as neighbours are chosen carefully, but not nearly as rigid as the DHT systems (more about those in two weeks)

Overview

- Characteristics of file sharing networks
- The shape of the overlay network
- Alternatives to flooding
- Adaptive probabilistic search
- Super node architectures
- Gia
- **Summary**

Conclusions

- **Searching in an unstructured P2P network is hard**
  - the network will change
  - not much knowledge and no central index
- **Flooding is not an efficient approach (quick, but *dirty*)**
- **Random Walkers improve considerably on the network efficiency**
- **Learning from the network traffic improves search**
- **Super node topologies recognise that peers have different capabilities**
- **Significant gains from a multi-pronged approach, affecting topology, flow, replication, and biased walking**

Conclusions

- **Scalability**
  - random walkers scale much better than flooding – especially with regards to message duplication
- **Performance**
  - user perceived delays are increased with walkers
  - however, increasing *k* leads to shorter delays
- **Fairness**
  - super nodes can improve performance, but should be themselves be rewarded for their extra work
  - well-connectedness is not equal to being able to handle the load

Conclusions

- **Integrity and security**
  - power-law and super node topologies are more vulnerable to targeted attacks
- **Anonymity, deniability, censorship resistance**
  - a hostile super node would be ideally placed to monitor or disrupt the network
  - adaptive systems can be hurt – being probabilistic helps