Constructing a Balanced, log(N)/loglog(N)-Diameter Superpeer Topology

Douglas S. Reeves ∽ Young June Pyun

N.C. State University

P2P 2004
August 26, 2004

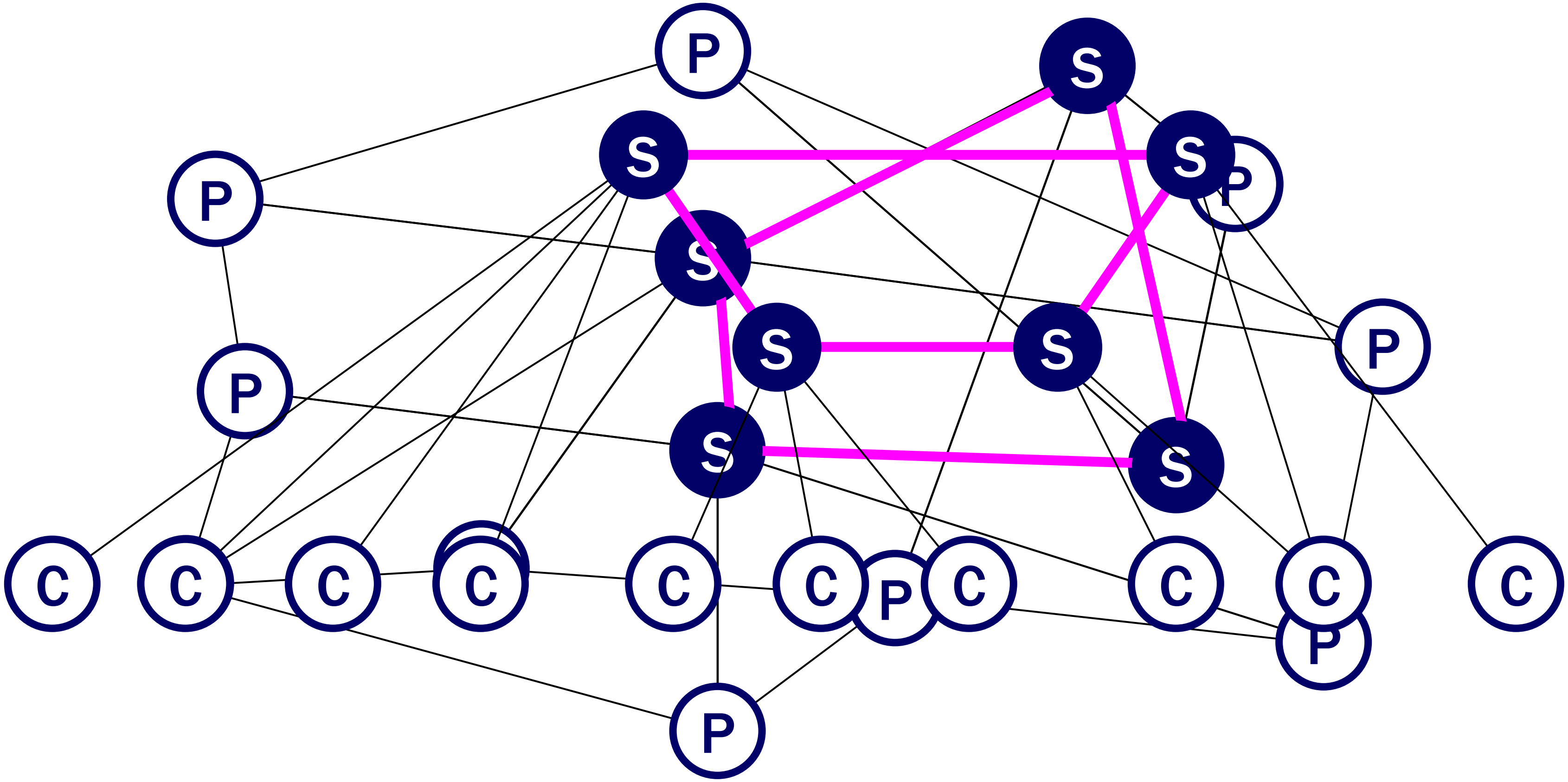
Outline

- Superpeer Topologies
- Approximating Random Graphs
- Results and Comparison

Superpeers

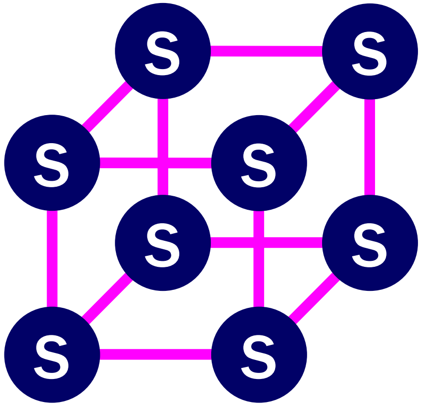
- Goal: improved scalability and performance for unstructured methods
- A two-level hierarchical P2P organization
  - upper level: super-peers
  - lower level: peers
- Queries are routed (via broadcasting) among super-peers only
- Proposed by KaZaA, adopted in Gnutella

Example: Adding Superpeers



Superpeer Topology?

- Desirable properties
  - connected
  - low diameter → fast search time
  - low node degree → efficient
  - regular → load balanced, avoid hot spots
- Example: the Binary Hypercube Topology
- Problem: highly structured!



Random Graphs

- Erdös and Rényi, 1959
- The random graph process:

G_0 = the graph consisting of N vertices and no edges

for τ = 1 to N*(N-1)/2 do

  select randomly an edge not in $G_{\tau-1}$

  construct G_τ by adding this edge to $G_{\tau-1}$

Properties of Random Graphs

- ER-model random graphs are almost regular
- When the number of edges $\tau = (N/2)\ \ln N$, the graph suddenly becomes connected
  - at this point, average node degree = $\ln N$
- Diameter $D(G_\tau)$ at this *hitting time* =

$$\left\lfloor \frac{\ln N - \ln 2}{\ln \ln N} \right\rfloor + 1 \le D(G_\tau) \le \left\lceil \frac{\ln N + 6}{\ln \ln N} \right\rceil + 3$$

- Has desired properties, even though unstructured!

Using Random Graphs for P2P?

- Problem: P2P systems are dynamic, not static
  - i.e., nodes may join and leave
- If edges are added equiprobably among all nodes, “older” nodes will have more edges than “younger” nodes

Approximating Random Graphs

- Adding a vertex to graph G_t (with N vertices):

$\delta = \lceil \ln(N) \rceil$

while there are vertices with degree < δ

    select randomly a vertex u with degree less than δ

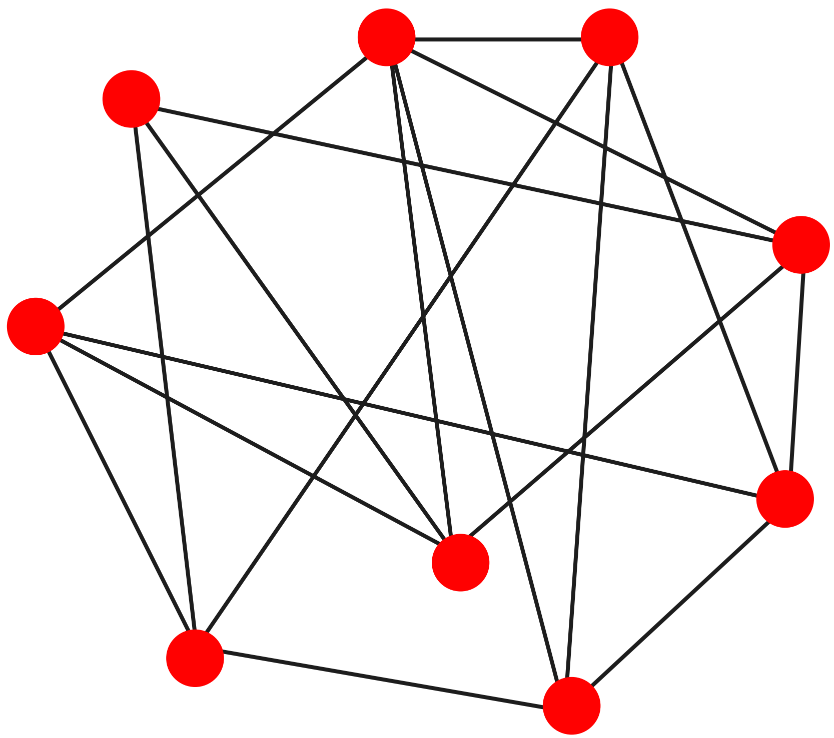
    connect u to a random vertex v of minimal degree

endwhile

return result as G_{t+1}

- This is referred to as the *δ-process*

Approximation Example



$N = 9$

$\delta = 3$

DONE

SUPS: An Approximately Random Superpeer Topology

- Based on the Random δ-Process
- Fully distributed
  - each node estimates the size of the network (*N*)
- Designed to tolerate faults, and support high join / leave rates

Information Needed by SUPS

- Each node *i* maintains the following information

N_i	a local estimate of N
δ_i	a local estimate of δ
$d(i)$	i's current degree
MinNode(i)	a random lowest-degree non-neighbor
MinList(i)	sorted list of δ_i *MinNodes(i)*
PriNodes(i)	3 primary neighbors

- Each node *i* periodically broadcasts i, N_i, $d(i)$
  - "piggy-backed" onto query messages

The Distributed δ-Process

- Each node *i* independently executes the following when its estimate N_i changes

$\delta_i = \lceil \ln(N_i) + 1 \rceil$

while $d(i) < \delta_i$

    select randomly a node *v* from *MinList(i)*

    connect *to v*

endwhile

Joining the Superpeer Topology

- A peer wishing to promote to a superpeer
  1. inherits the information maintained by its two "parent" superpeers
  2. invokes the distributed δ-process
- *PriNodes(i)* = first 3 nodes to which it connects
  - each of these 3 nodes broadcasts a Join(i) message
- A node j receiving 2 or more of these messages
  1. increments its value N_j
  2. invokes the distributed δ-process

SUPS Resilience to Failures

- Any value of N_i for which

$$\lceil \ln (N_i) + 1 \rceil = \lceil \ln (N) + 1 \rceil$$

  will produce the correct estimate for δ_i
  - i.e., there is a fairly large "noise margin" for estimation error

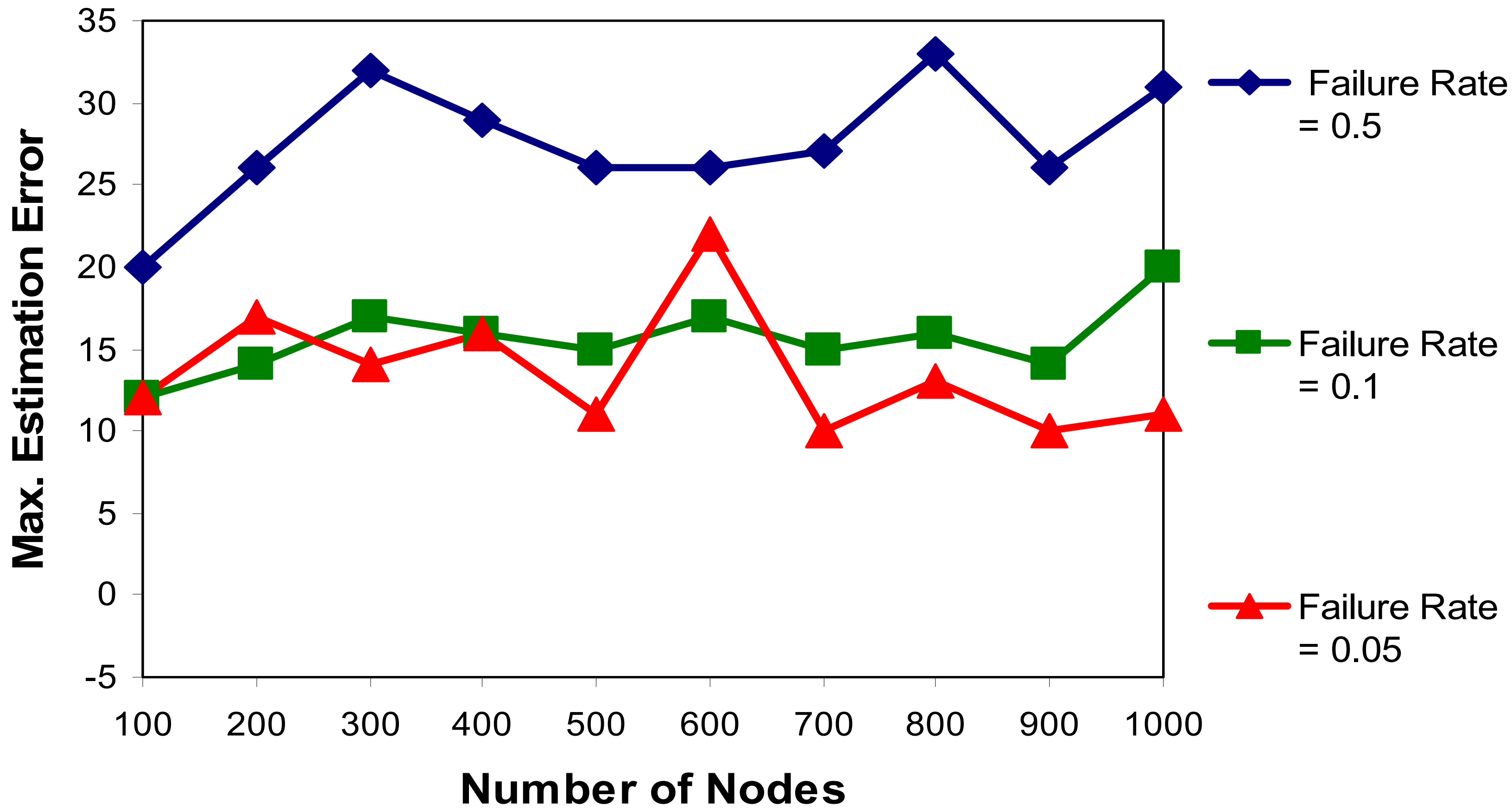
- When receiving a Join message, synchronize N_i with values from other nodes (on *MinList(i)*)

- Periodically tabulate (from scratch) the number of nodes N in the system, and broadcast N

Evaluation of SUPS

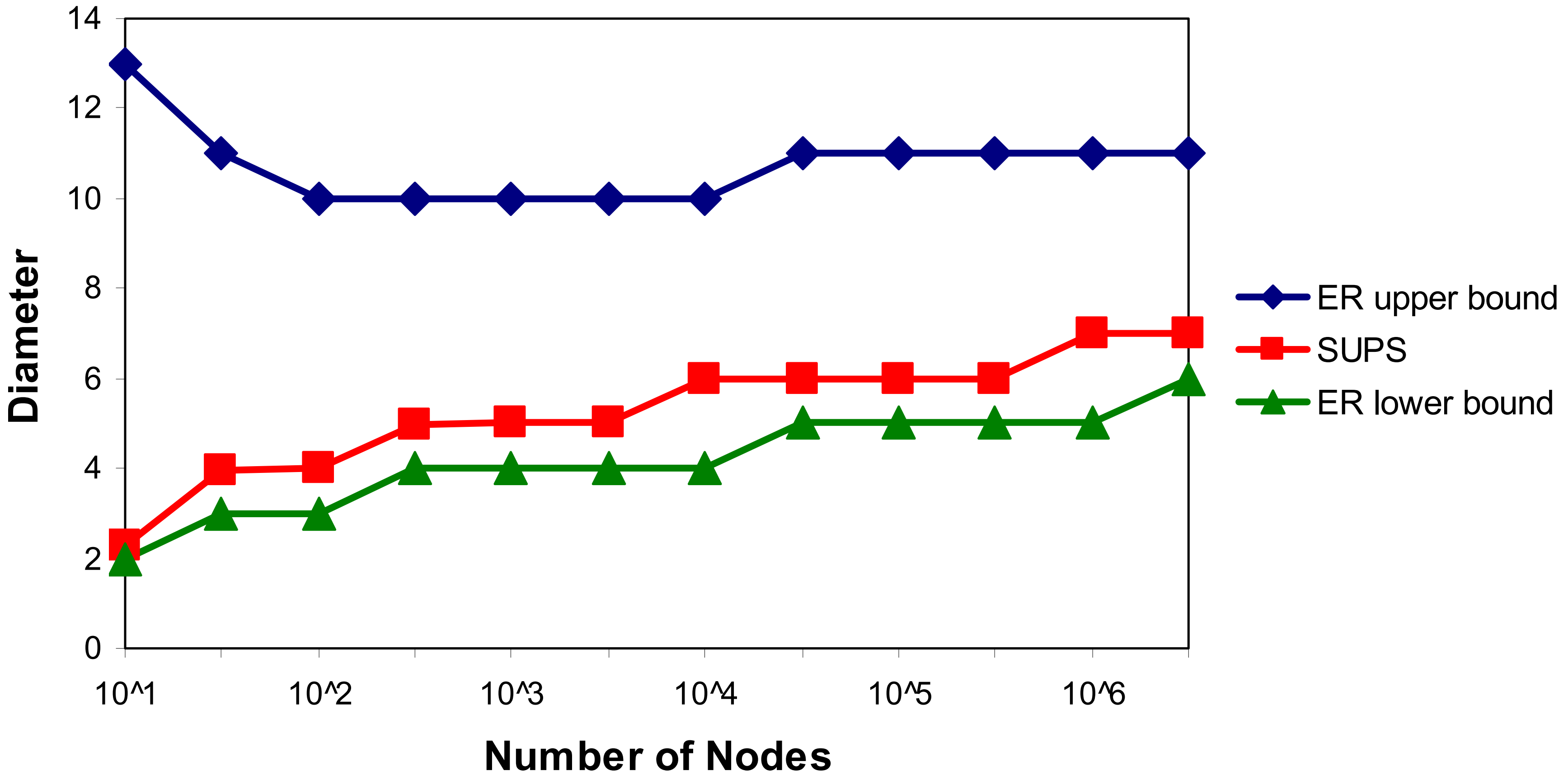
- Simulated just the graph construction
  - since queries are broadcast, there is no need to simulate them
- Measurements
  - impact of failures on the accuracy of N_i and δ_i
  - diameter of network
  - node degree
  - (fraction of nodes affected by a node join or leave)
- Resulting graphs always connected

Results: Error in Estimating *N*



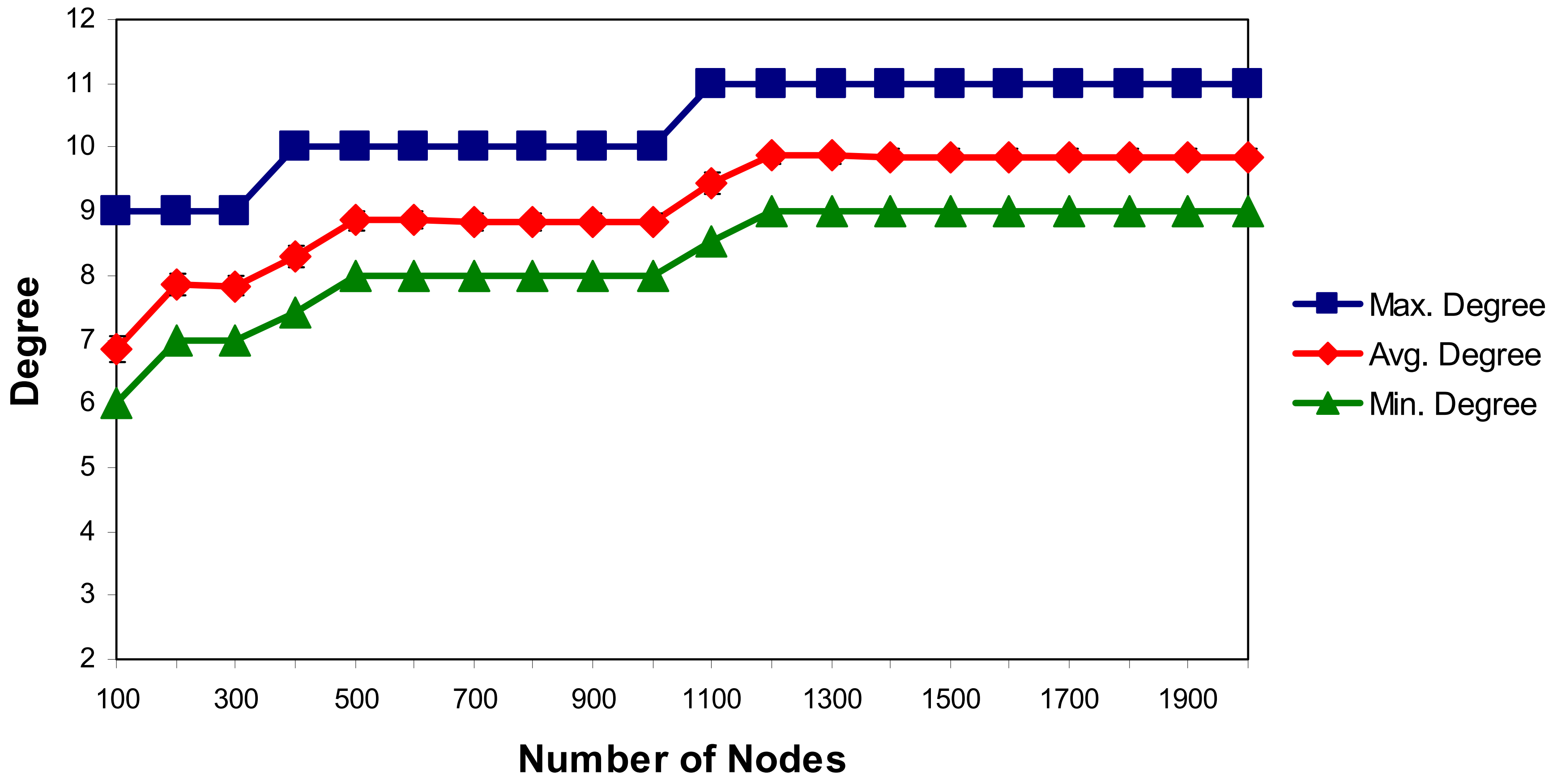
➔ Error in estimating δ always ≤ 1

Results: Diameter



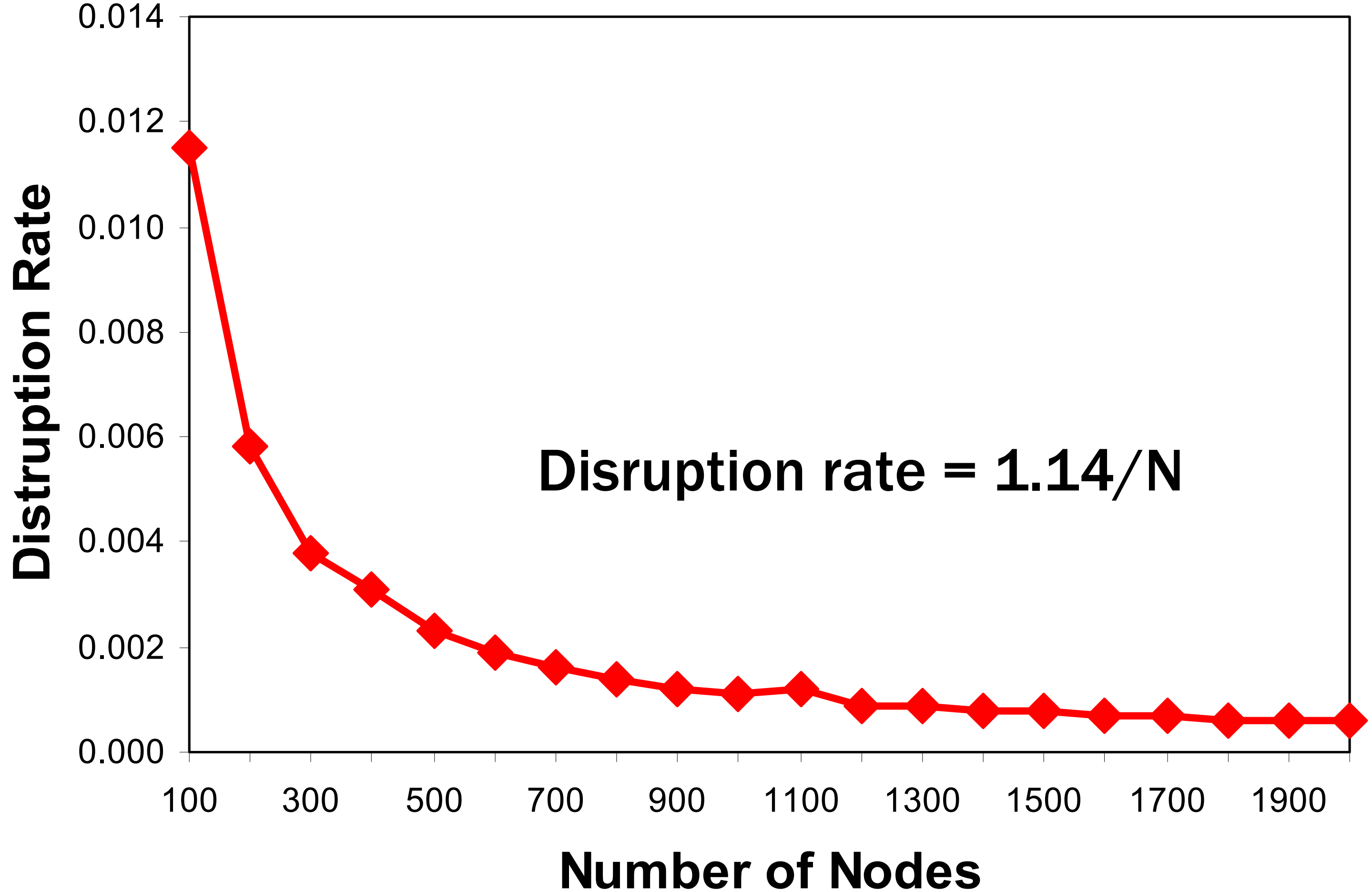
- Diameter of SUPS close to ER model lower bound

Results: Node Degree / Regularity



➔ SUPS produces an almost θ(ln*N*)-regular graph

(Results: Disruption Rate)

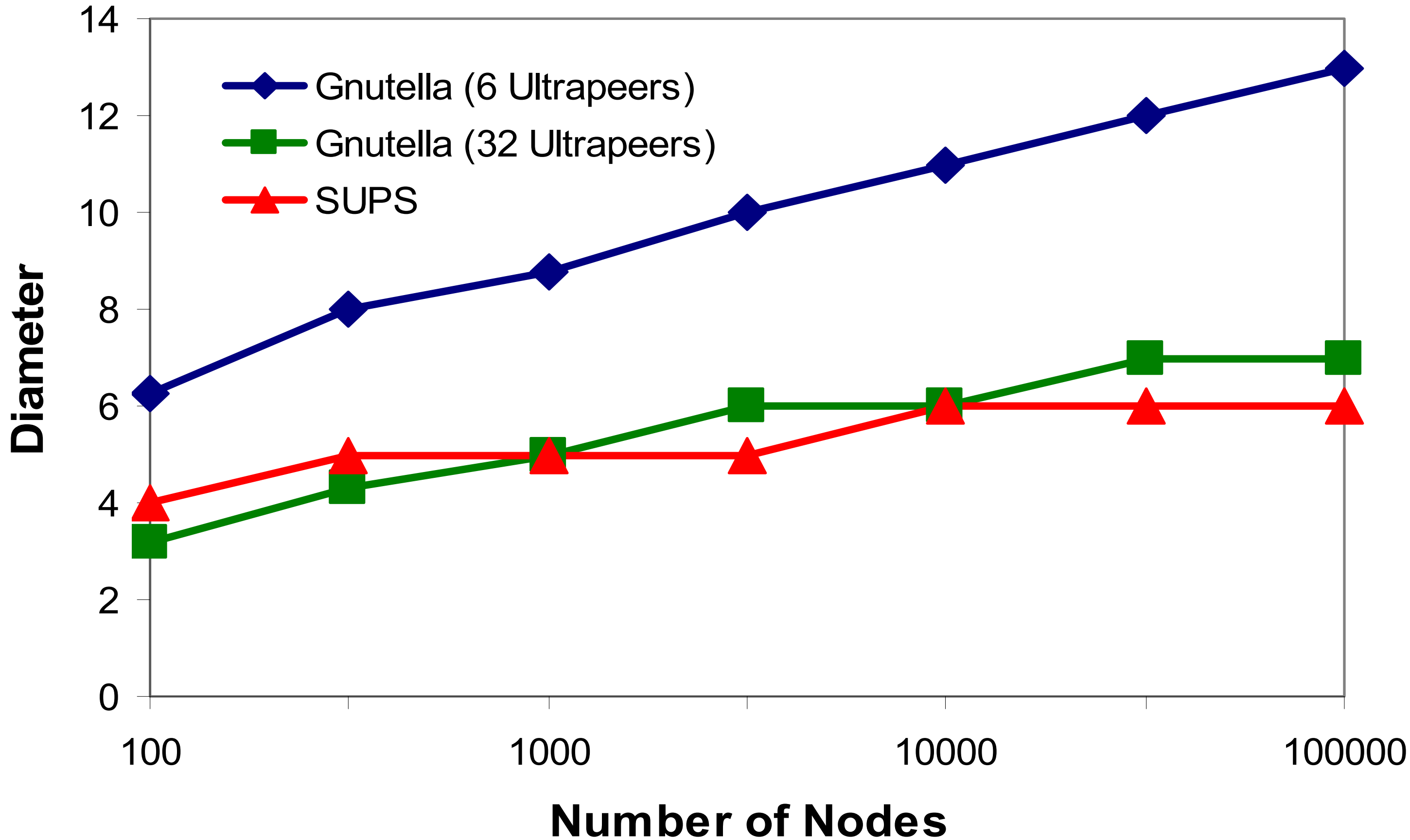


- Avg. of 1.14 nodes are affected regardless of N

Comparison with Gnutella v0.6

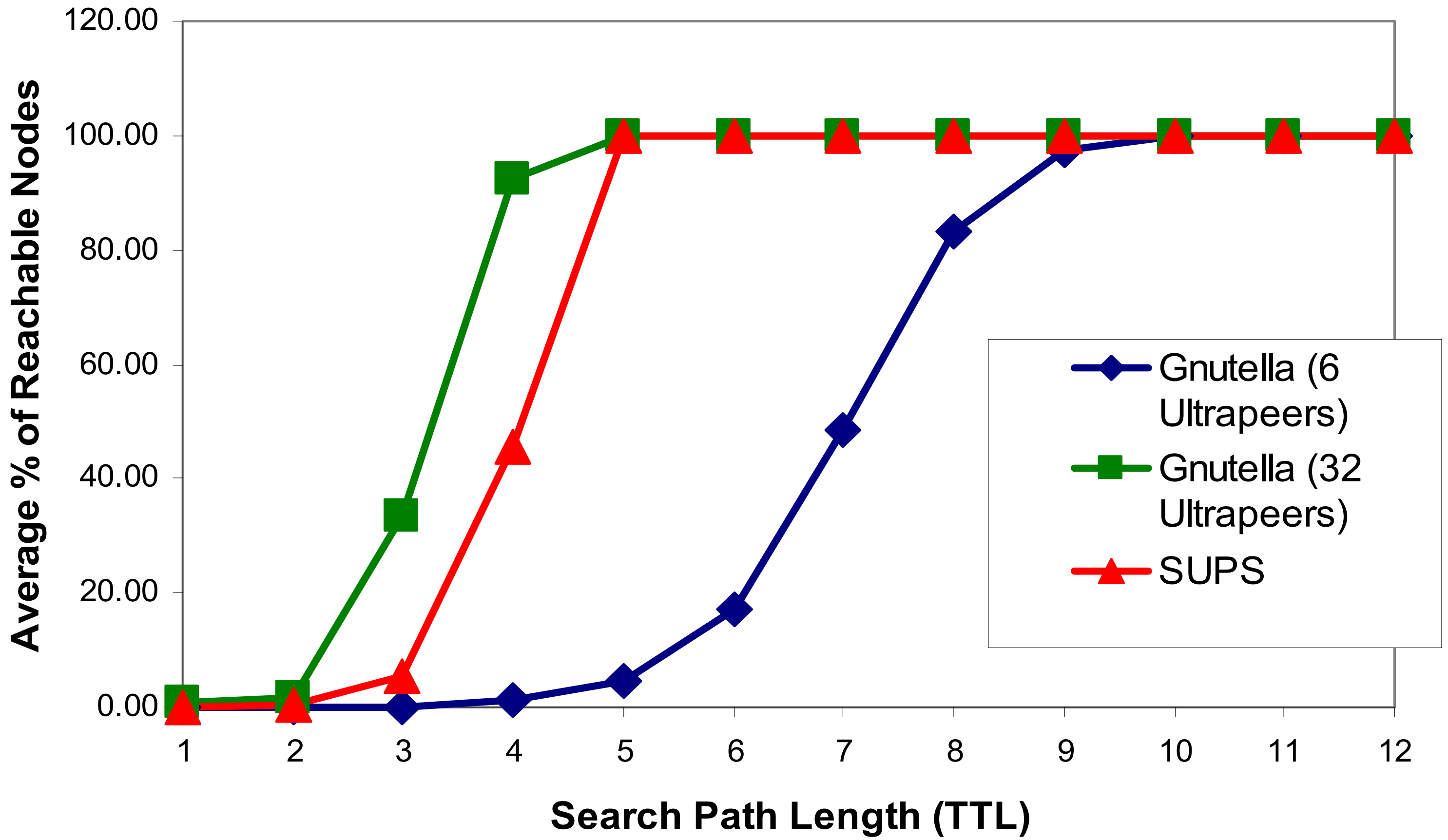
- Gnutella has proposed *ultrapeers*
  - degree of each ultra peer = 6 (earlier) or 32 (more recently)
  - we generated connections perfectly randomly (unrealistically favorable)
- Measurements
  - diameter
  - reachability = fraction of nodes reachable within a given TTL

Comparison: Diameter



- G32 : same diameter (w. 190% more edges)
- G6: 83% bigger diameter (w. 45% fewer edges)

Comparison: Reachability



- G32 : equivalent reachability in 1 less "hop"
- G6 : much worse reachability

Conclusions

- Random graphs are a very good unstructured topology for P2P
- SUPS produces a good approximation of a random graph
  - low overhead, simple, and practical
  - robust to failures and rapid changes of nodes
  - compatible with currently-deployed P2P systems
- Suitability of the δ-process for other distributed applications?