Announcements

- **Homework 1 is out:**
  - It is **long**! Start early.
  - Analyze a co-authorship network
  - Due in **1 week (Oct 8 in class)**
- **Project proposals are due in 2 weeks!**
  - Groups of max 3 students
  - 1 proposal per group, max 3 pages
  - Read at least 3 papers from the Easley-Kleinberg book
    - Summarize them, describe strong/weak points and extensions
  - **Propose what you want to do** (can be one of the following):
    - Experimental evaluation
    - Theoretical project, model, simulation
    - In-depth critical survey

Small World in Real World Search in P2P Networks

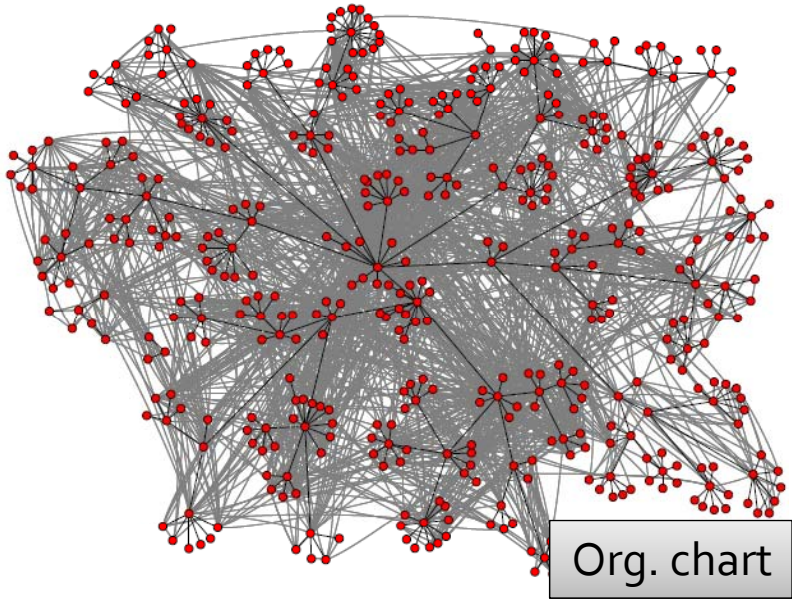
CS 322: (Social and Information) Network Analysis

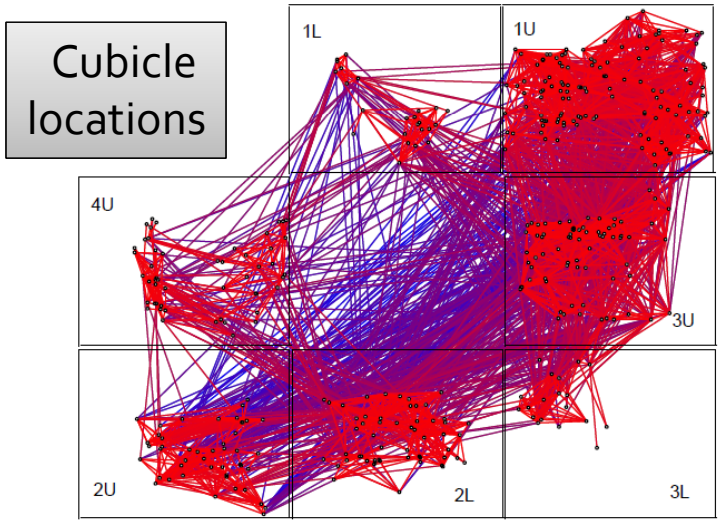
Jure Leskovec
Stanford University

[Adamic-Adar 2005]

Small-world in HP labs

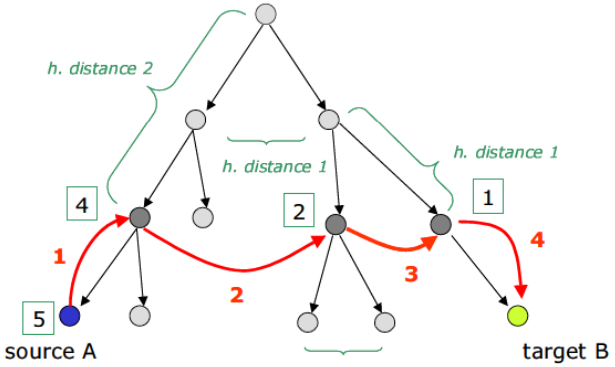
- Adamic-Adar 2005:
  - HP Labs email logs (436 people)
  - Link if u,v exchanged >5 emails each way
  - Map of the organization hierarchy
  - Finding:
    $P(u \rightarrow v) \sim 1 /$ (size of smallest group containing u and v)$^{3/4}$
- Differences:
  - Data has weighted edges
  - Data has people on non-leaf nodes
  - Data not b-ary or uniform depth
- Q: How can we adapt this to weighted graphs?

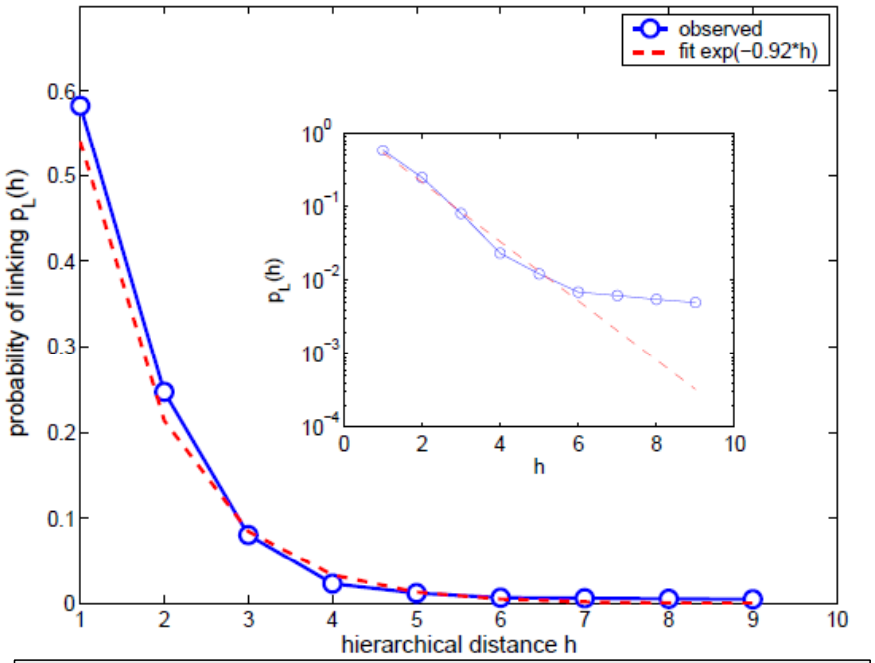




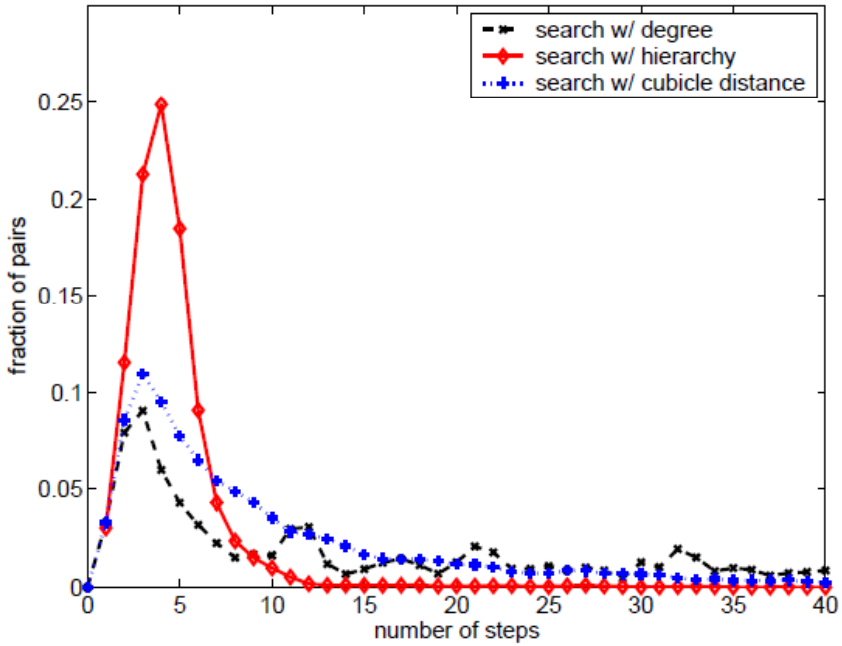
[Adamic-Adar 2005]

Small-world in HP labs





Probability of link vs. distance in the hierarchy



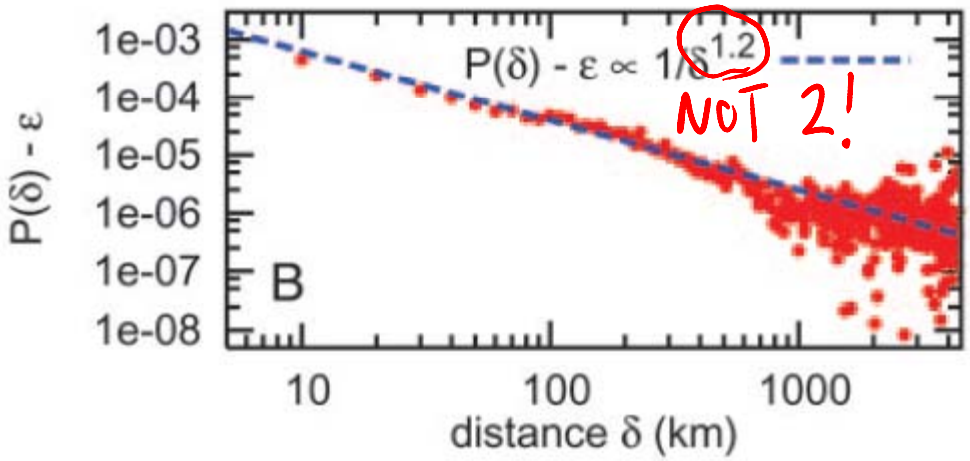
Search strategies using degree, hierarchy, distance between the cubicles

[Liben-Nowell et al. '05]

Small-world in LiveJournal

Liben-Nowell et al '05:

- Live Journal data (blogers+zip codes)



Link length in a network of bloggers (0.5 million bloggers, 4 million links)

$P(u, v) = \delta^{-x}$

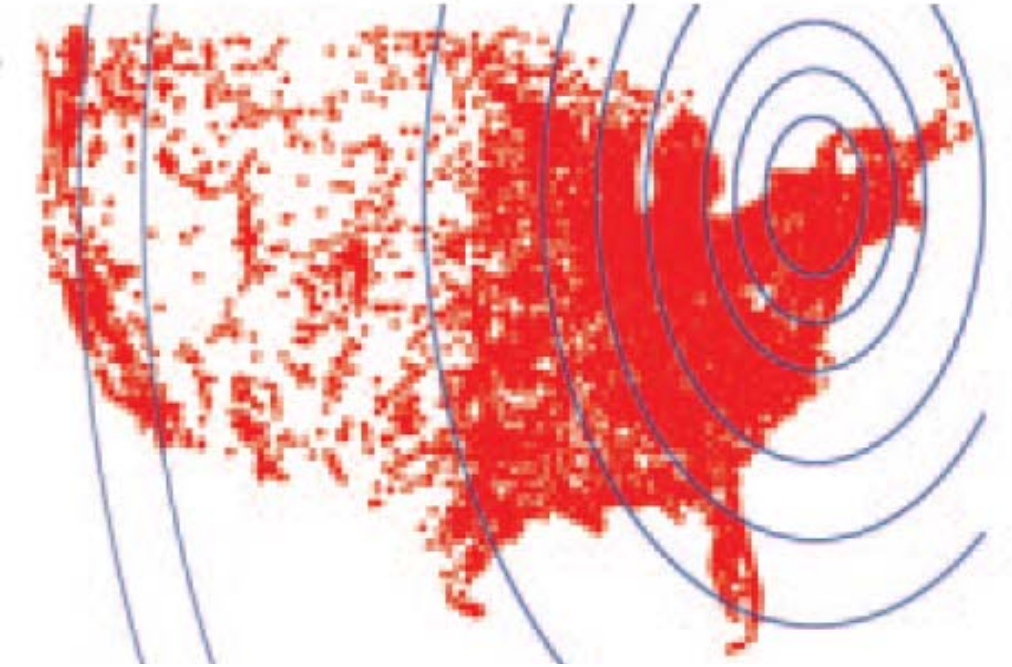
$x = 2$

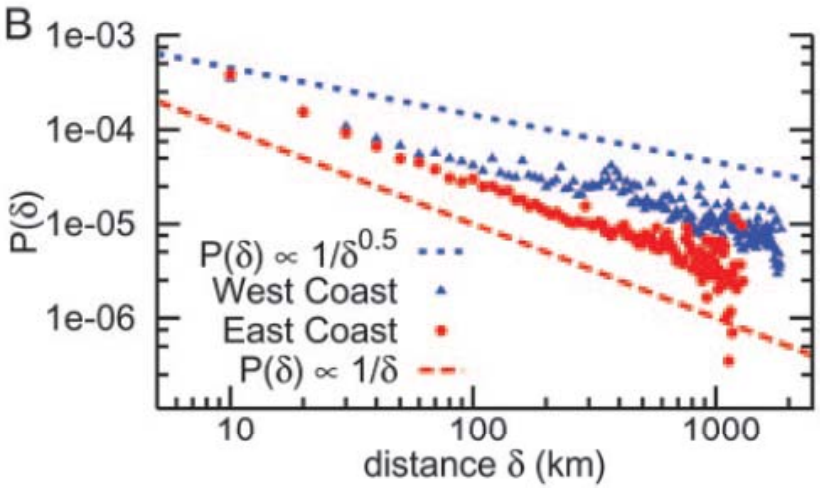
- Solution:

  Rank based friendship

[Liben-Nowell et al. '05]

Improved model





$$rank_u(v) := \left|\{w : d(u,w) < d(u,v)\}\right|$$

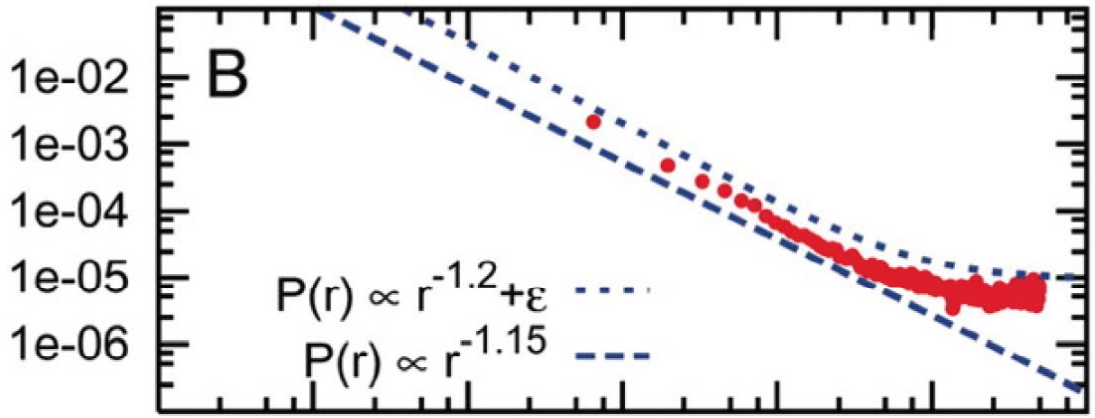
PEOPLE ARE NOT SPACED UNIFORMLY

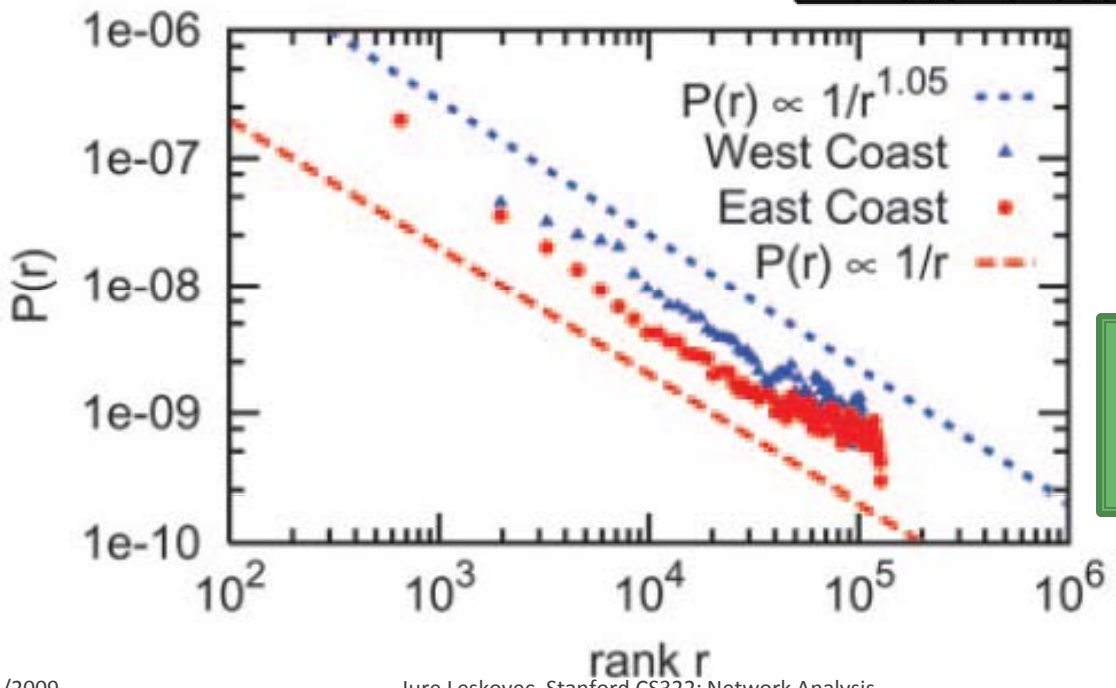
$$\Pr[u \to v] \propto \frac{1}{rank_u(v)}.$$

[Liben-Nowell et al. '05]

Rank based friendships

- Close to theoretical optimum of -1

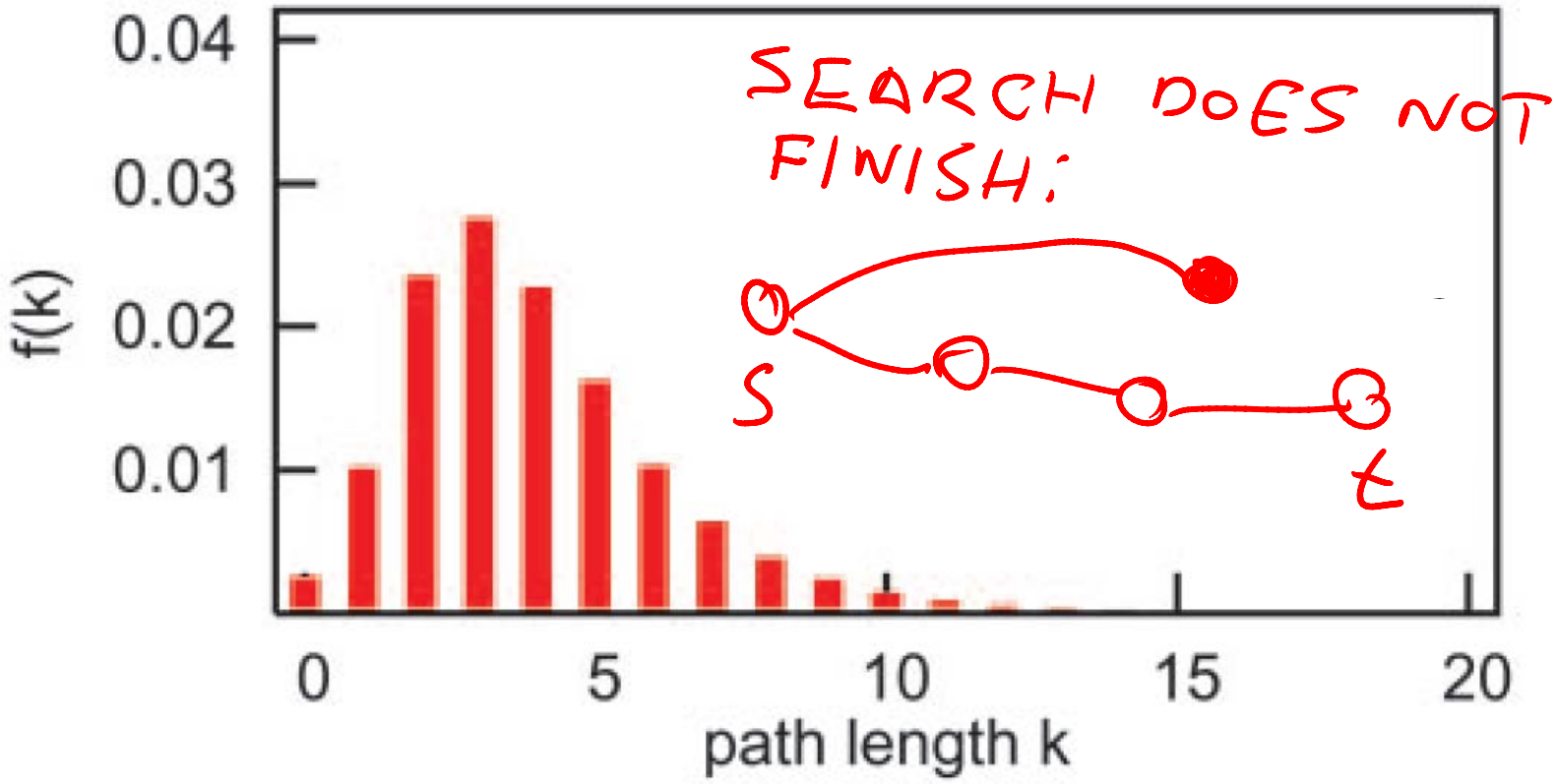




The difference between the East and West coast disappears

[Liben-Nowell et al. '05]

Geographic Navigation



- Decentralized search in a LiveJournal network
  - 12% searches finish, average 4.12 hops

Question:

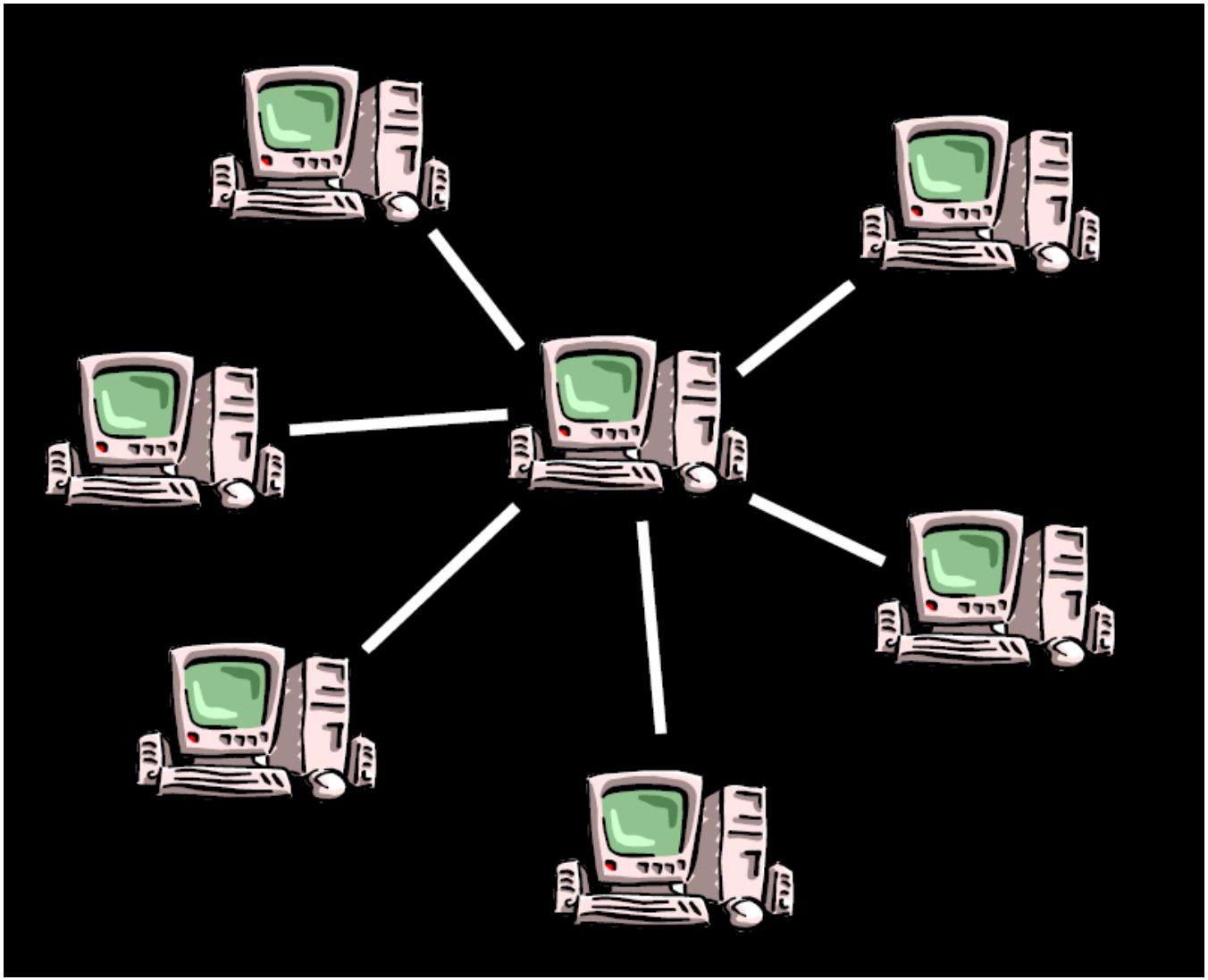
- Why is rank exponent close to -1?
  - Why in any network? Why online?
  - How robust/reproducible?
- Conjecture [Sandbeng-Clark 2007]:
  - Nodes in a ring with random edges
  - Process of morphing links
    - Update step: Randomly choose s, t, run decentralized search alg.
    - Path compression: each node on path updates long range link to go directly to t with some small prob.
  - Conjecture from simulation:
    $P(u \to v) \sim dist^{-1}$

Question

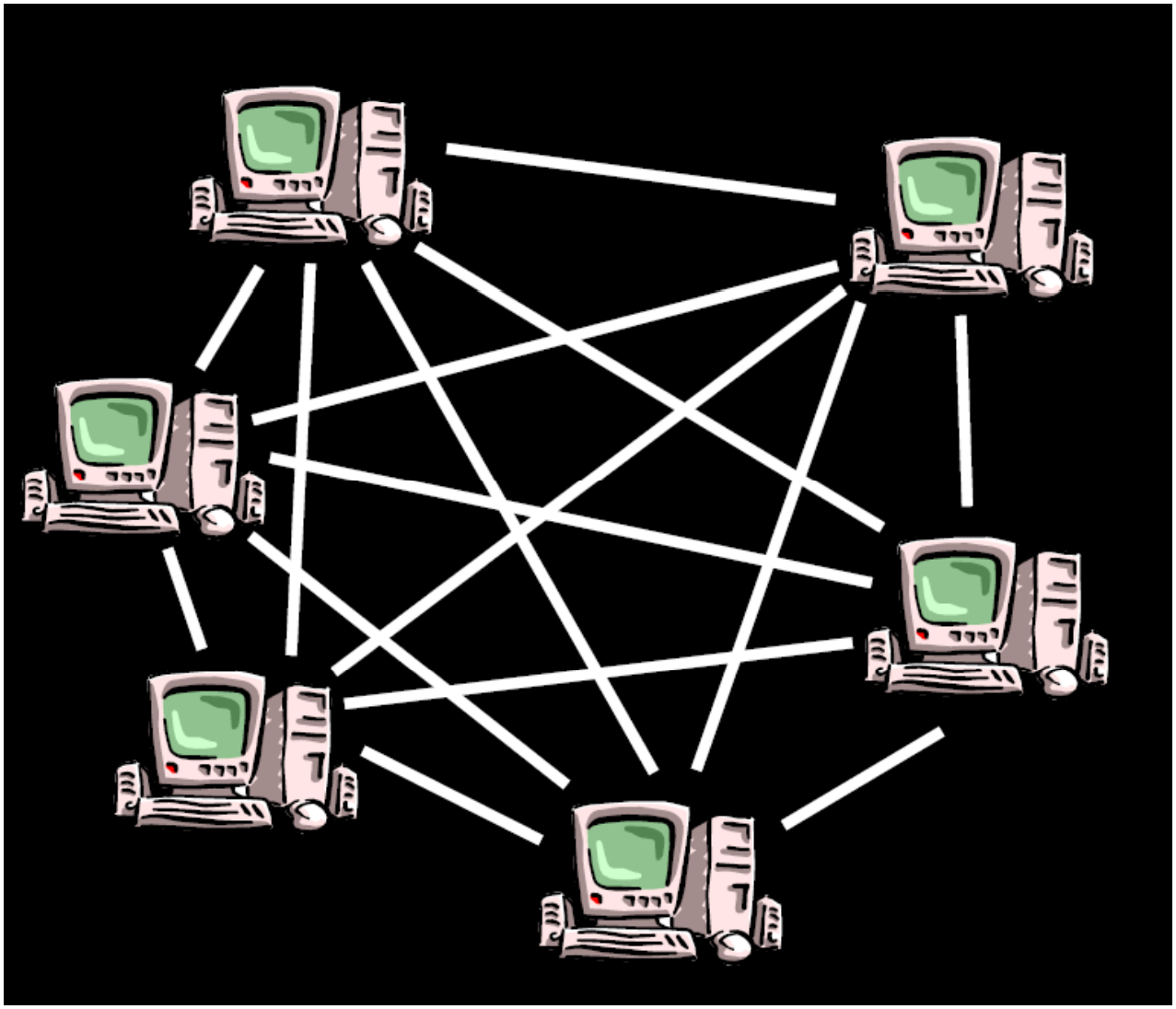
Algorithmic consequence:

How to find files in Peer-to-Peer networks?

Client – Server



P2P: only clients



Napster

- Napster existed from June 1999 and July 2001
- Hybrid between P2P and a centralized network
- Once layers got the central server to shut down the network fell apart

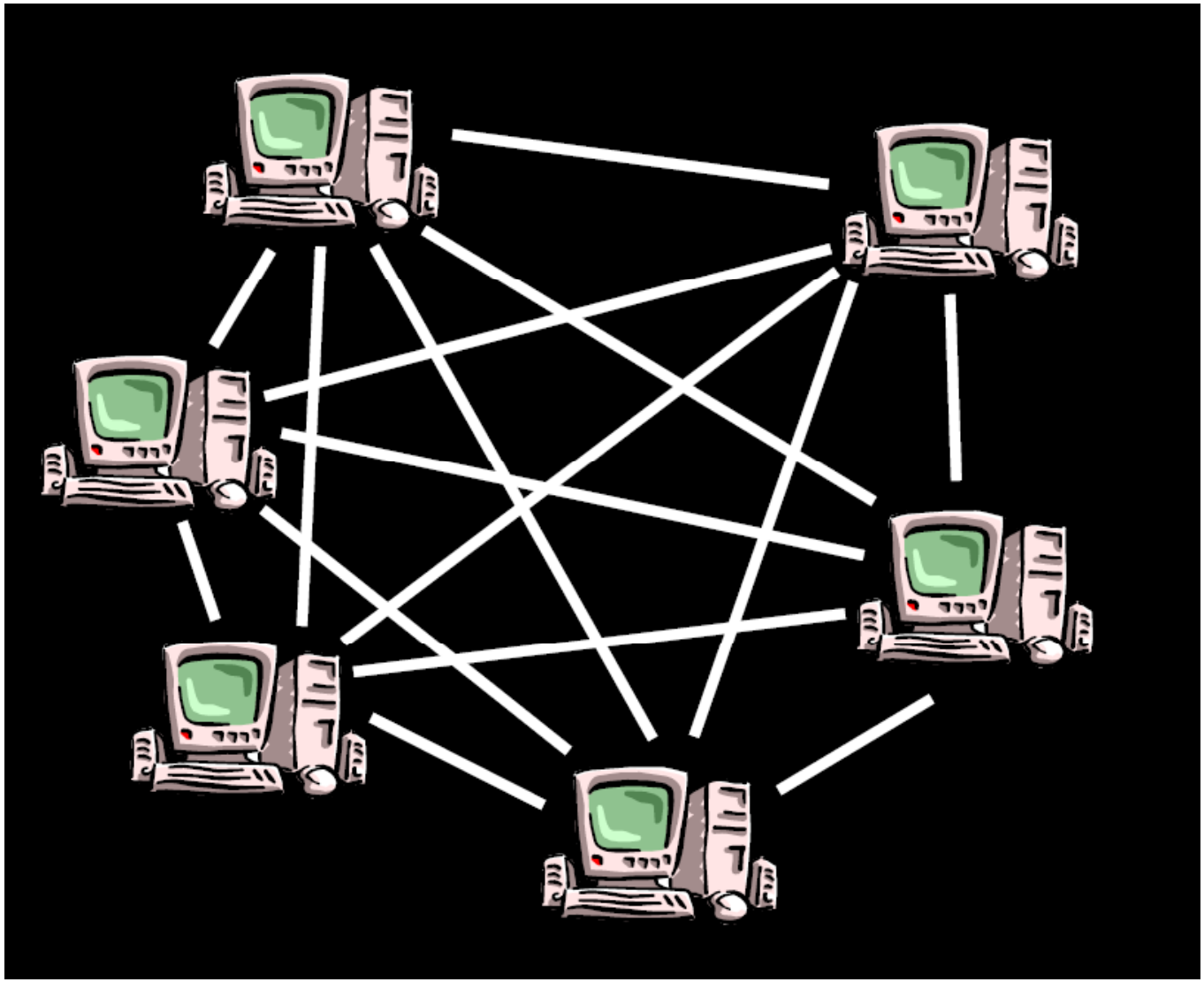
True P2P networks

- Networks that can't be turned "off"
  - BitTorrent
  - ML-donkez
  - Kazaa
  - Gnutella

Question

Can we find a file in a network without a central server?

Bad idea: Ask everyone

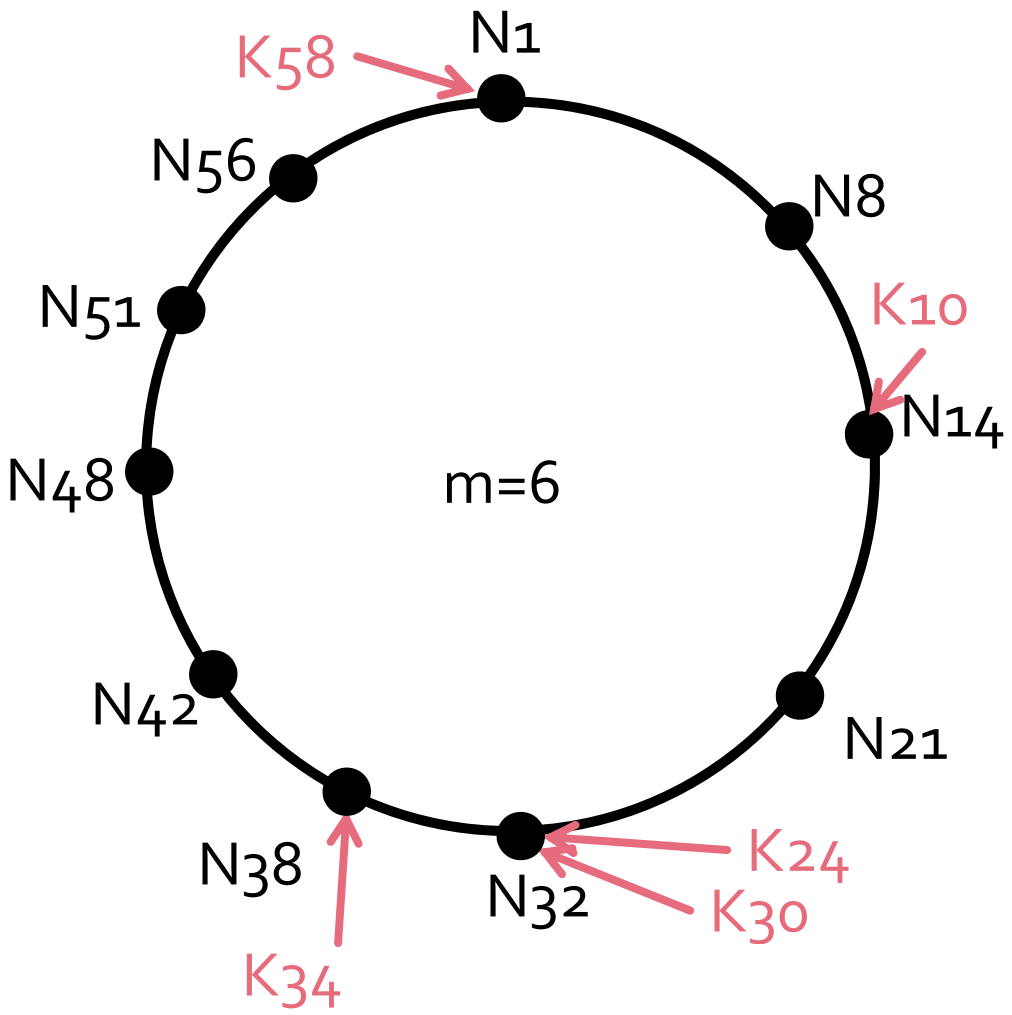


Protocol Chord

- Protocol Chord consistently maps key (filename) to a node:
  - Keys are files we are searching for
  - Computer that keeps the key can then point to the true location of the file
- Keys and nodes have m-bit IDs assigned to them:
  - Node ID is a hash-code of the IP address
  - Key ID is a hash-code of the file

Chord on a cycle

- Cycle, with node ids *0* to 2^{m-1}
- Key k is assigned to a node $a(k)$ with ID $\geq k$

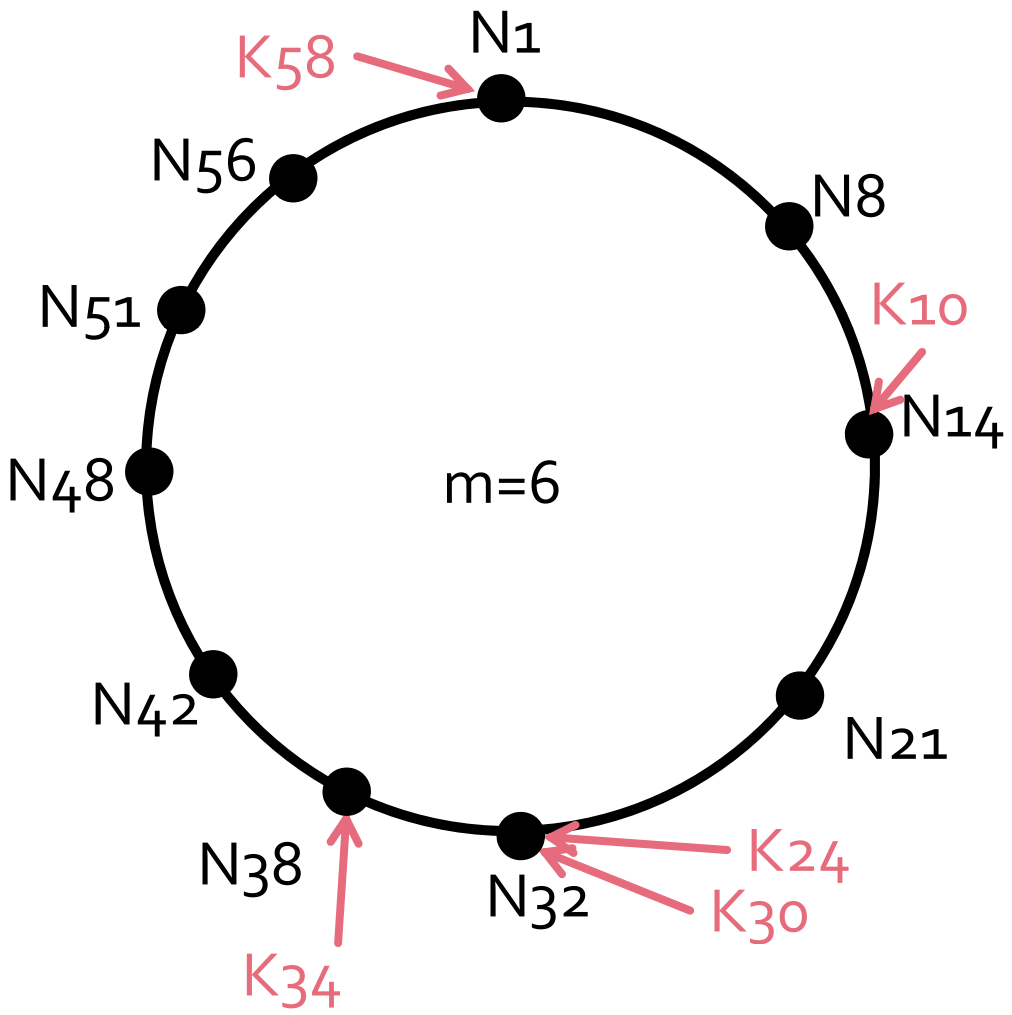


Basics

- Assume we have N nodes and K keys (files). How many keys has each node?
- When a node joins/leaves the system it only needs to talk to its immediate neighbors
  - When $N+1$ nodes join or leave, then only O(K/N) keys need to be rearranged
- Each node know the IP address of its immediate neighbor

Searching the network

- If every node knows its immediate neighbor then use sequential search

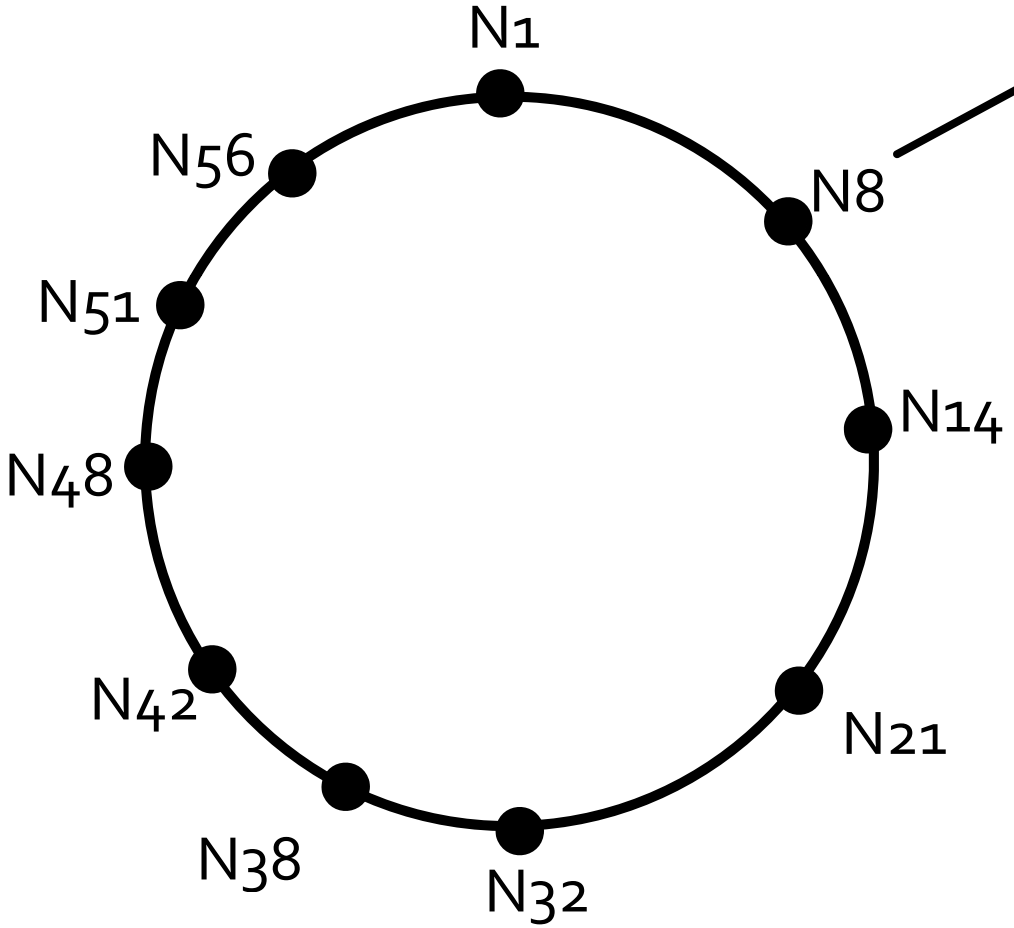


Faster search

- A node maintains a table of $m=log(N)$ entries
- i-th entry of a node n contains the address of $(n+2^i)$-th neighbor

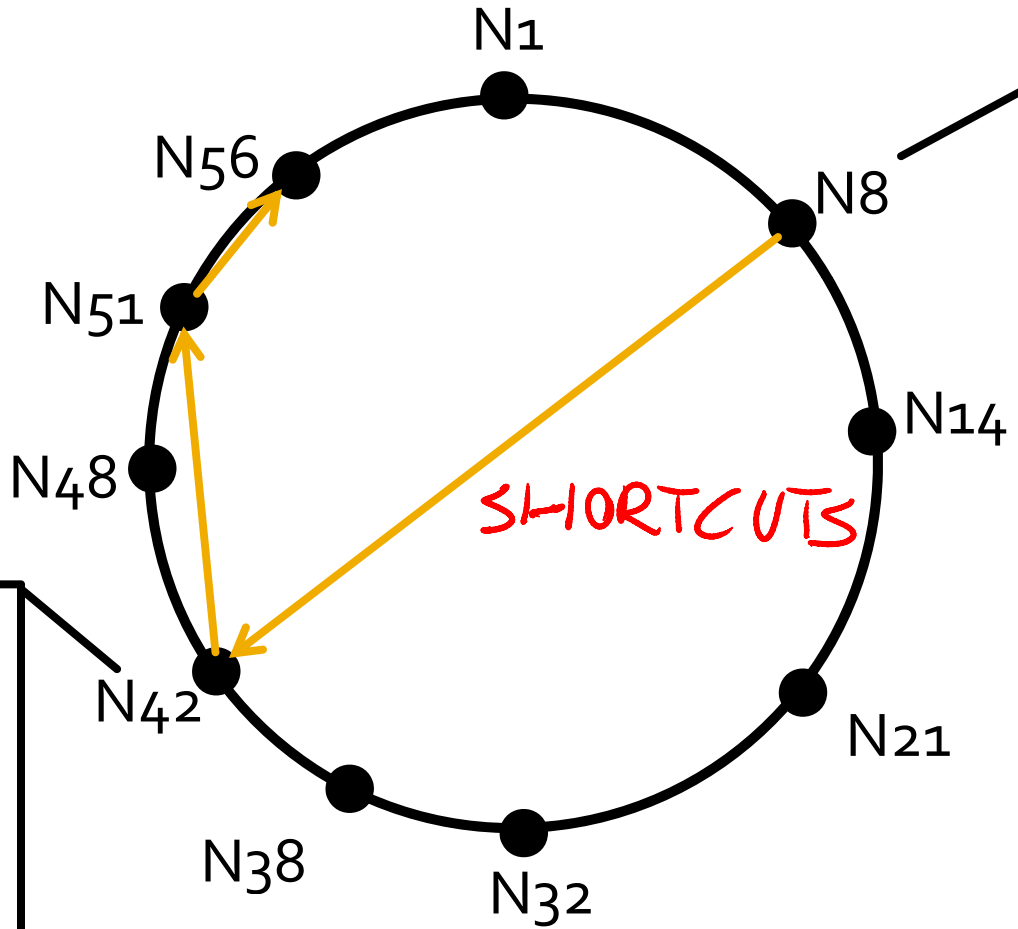
- Search algorithm:
  - Take the longest link that does not overshoot
    - This way with each step we half the distance to the target

i-th entry of N has the address of (N+2^i)-th node



N8+1 = N14
N8+2 = N14
N8+4 = N14
N8+8 = N21
N8+16 = N32
N8+32 = N42

Find key with ID 54



N8+1 = N14
N8+2 = N14
N8+4 = N14
N8+8 = N21
N8+16 = N32
N8+32 = N42

N42+1 = N48
N42+2 = N48
N42+4 = N48
N42+8 = N51
N42+16 = N1
N42+32 = N8

How long does it take to find a key?

- Search for a key in a network of N nodes visits O(log N) nodes

- Assume that node n queries for key k
- Let the key k reside at node t

- How many steps do we need to reach t?

Proof

- We start the search at node n
- Let i be a number such that t is contained in interval $[n+2^{i-1}, n+2^{i}]$
- Then the table at node n contains a pointer to node $n+2^{i-1}$ – the smallest node f from the interval
- Claim: f is closer to t than n
- **So, in one step we halved the distance to t**
- We can do this at most $log\ N$ times
- Thus, we find t in $O(log\ N)$ steps

Hyper-cube

- Hypercube:
  - $N = 2^d$
  - Edge u-v if bit-string differs in 1 position
- Properties:
  - Low max-degree: log N
  - Low diameter: log N
  - Good expansion: 1 (cut out 2^{d-1} hypercube)
- Simplest graph with above three properties