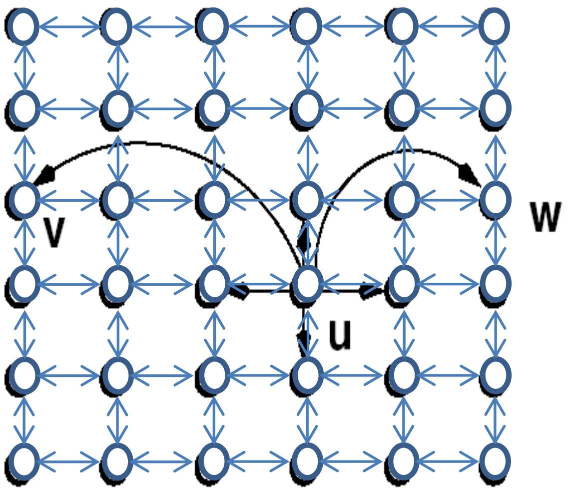
Kleinberg's Small-World Networks

$$P(u \to v) \sim \frac{1}{d(u,v)^r}$$

$$P(u \to v) = \frac{1}{d(u,v)^r} \cdot \frac{1}{Z}$$



- Normalization constant have to be calculated:

$$Z = \sum_{\forall i \neq u} \frac{1}{d(u,i)^r}$$

Example

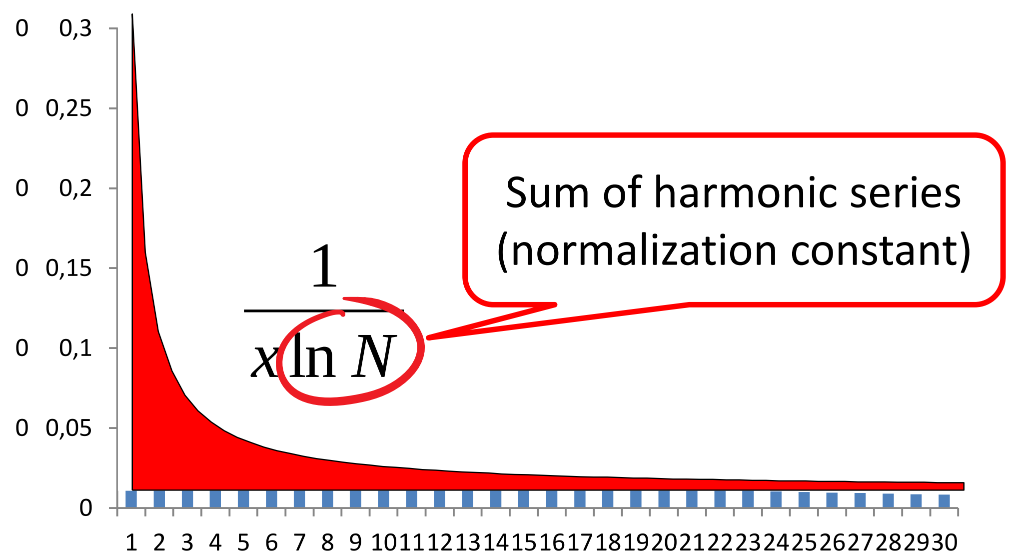
- Choose among 3 friends (1-dimension)
  - A (1 mile away)
  - B (2 miles away)
  - C (3 miles away)
- Normalization constant

$$\sum_{\forall i \neq u} \frac{1}{d(u,i)} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} = \frac{11}{6}$$

$$P(selectingA) = \frac{\frac{1}{1}}{\frac{11}{6}} = \frac{6}{11}$$

$$P(selectingB) = \frac{\frac{1}{2}}{\frac{11}{6}} = \frac{3}{11}$$

$$P(selectingC) = \frac{\frac{1}{3}}{\frac{11}{6}} = \frac{2}{11}$$

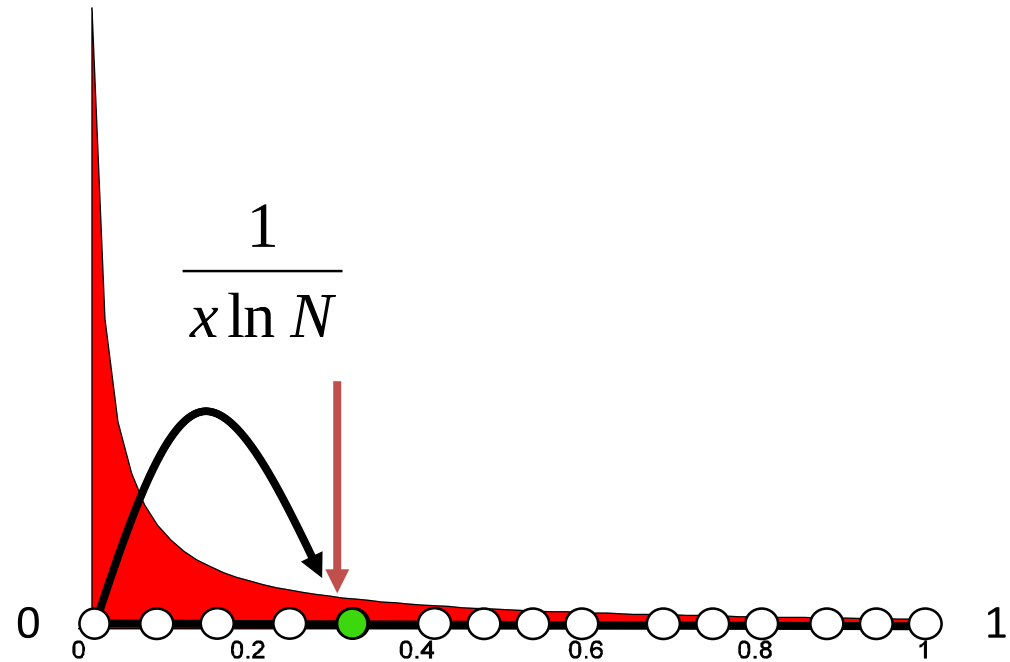


1-dimensional continuous case

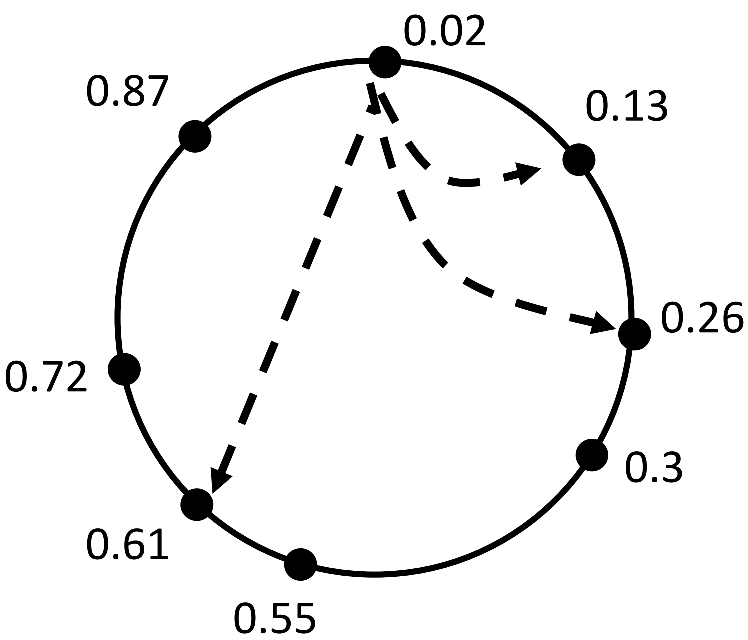
- Peers uniformly distributed on a unit interval (or a ring structure)
- Long range links chosen by Kleinberg's small-world principle
  - Search cost

O($\log^2 N/k$) with k long-range links

O(logN) with O(logN) long-range links



Small-World based P2P Overlay



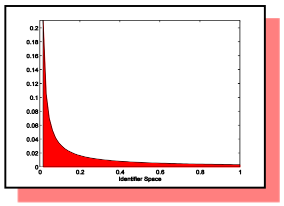
Systems:

Symphony (Manku et al, USITS 2003)

Accordion (Li et al, NSDI 2005)

- **Peers** mapped on the ring
  - Uniform hash function (e.g., SHA-1)
  - Establishing ring links
- Small-World **connectivity establishment**

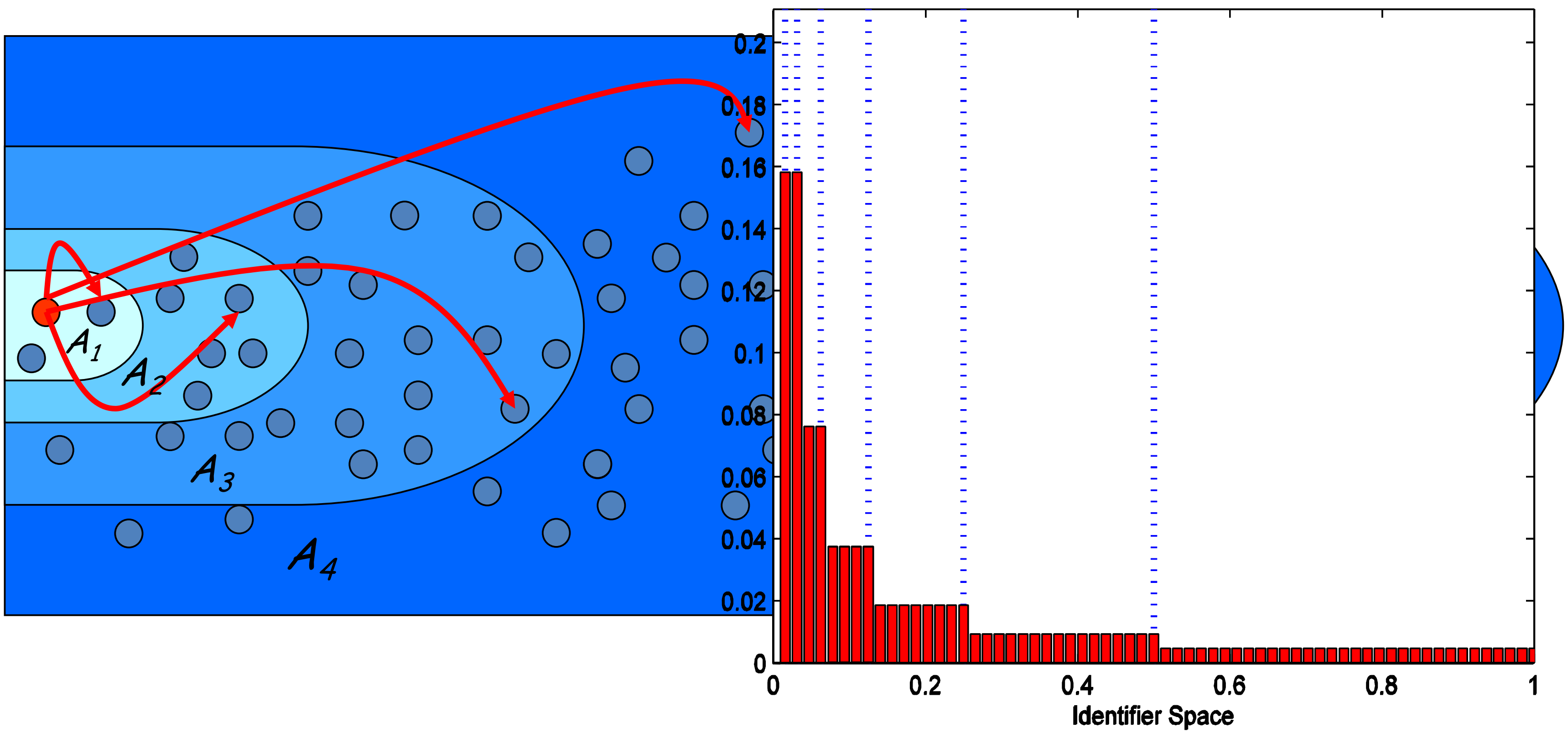
$$\frac{1}{x \ln N}$$



- No restrictions on peer-degree
- Will **not have optimal routing performance** if the node-ids are **non-uniformly distributed**!

Approximation of Kleinberg's model

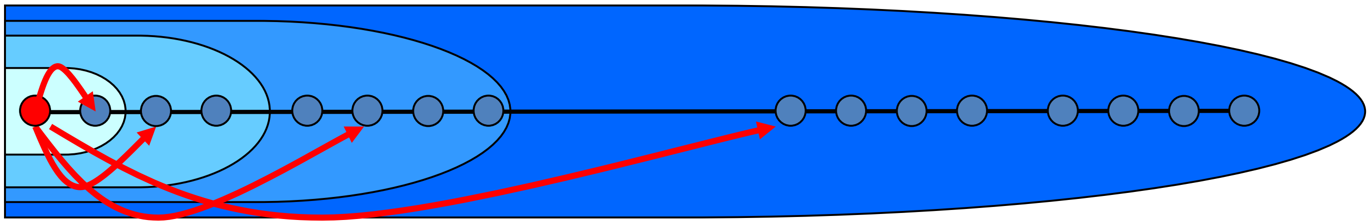
- Given node *u* if we can partition the remaining peers into sets $A_1, A_2, A_3, \ldots, A_{logN}$, where A_i, consists of all nodes whose distance from *u* is between 2^{-i} and 2^{-i+1}.
  - *Then given r=dim each long range contact of u is nearly equally likely to belong to any of the sets* A_i
  - *When q=logN – on average each node will have link in each set of* A_i



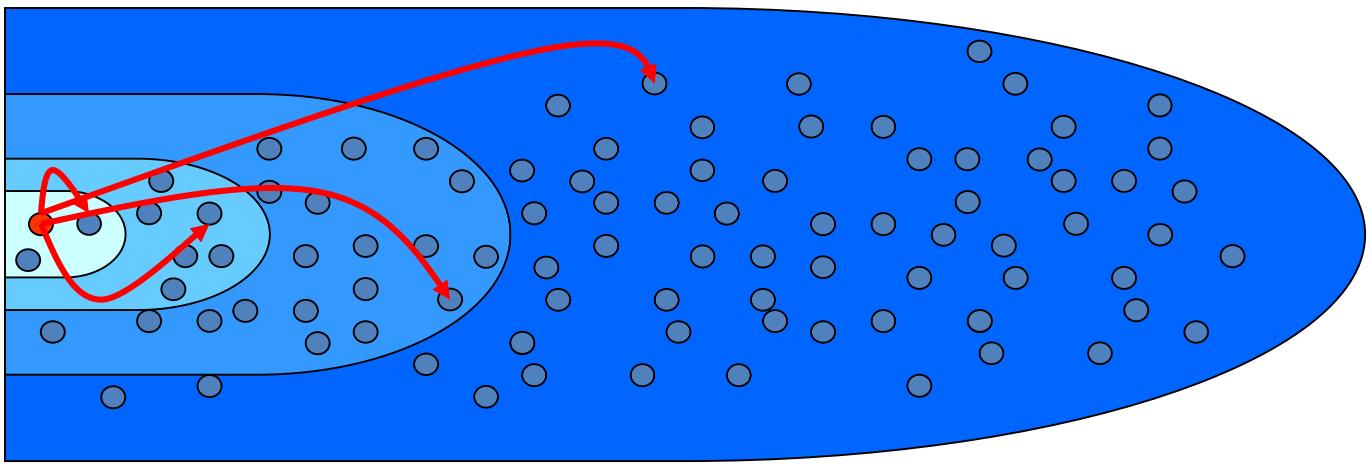
Traditional DHTs and Kleinberg model

- Most of the structured P2P systems are similar to Kleinberg’s model and are called logarithmic-like approaches. E.g.
  - Chord (randomized version) q=logN, r=1

- Randomized Chord’s model



- Kleinberg’s model



Basic Navigation Principles

- So **why** can we navigate in the network and **find short paths** without **any global view** of the system?
  - We have **globaly agreed ID space**, with a distance function –>
    - Allows us to make local decisions on minimizing distance to the target
  - Existinace of the **underlying lattice** (i.e., ring) ->
    - Assures us to always be able to make progress navigating towards the target, i.e.,
    - Target will always be reached!
  - Existance of **Kleinbergian long-range links** ->
    - Allows us to progres towards the target rapidly (in polylog steps)

What to take away from the small-world tour?

- What is the main difference between a **random graph** and a **SW graph**?
- What is the main difference between **Watts/Strogatz** and **Kleinberg models**?
- Why are we able to find **short paths** without global knowledge?
- What is the relationship between **structured overlay networks and small world** graphs?
- What are possible **variations of the small world** graph model?

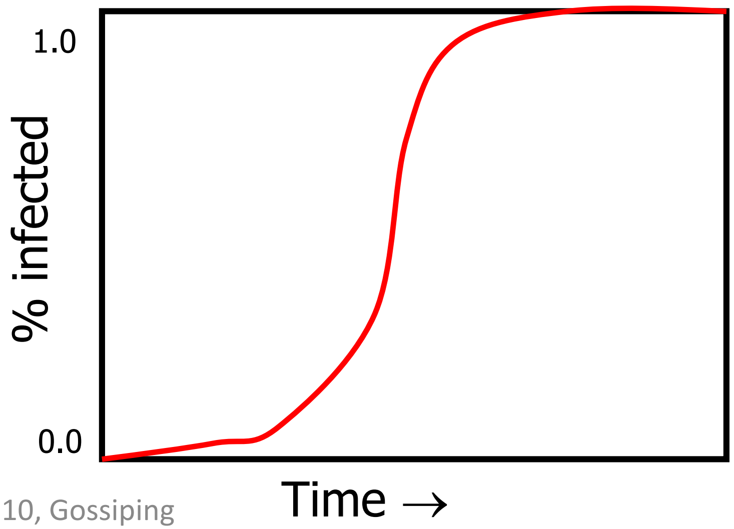
Gossiping Algorithms

Gossip Intro.

- Suppose that I **know** something
- I'm sitting next to Alice, and I tell her
  - Now 2 of us "know"
- Later, she tells Bob and I tell Carol
  - Now 4 of us "know"
- This is an example of a ***push*** epidemic
- ***Pull*** happens if Alice asks me instead
- ***Push-pull*** occurs if we exchange data

Gossip scales very nicely

- Participants' loads are independent of system size
- Network load is linear in system size
- Usually information spreads in log(system size) time
  - **Any guess why?**



So what is a gossip protocol ?

- Cyclic/Periodic, pair-wise interaction between peers
- The amount of information exchanged is of **(small) bounded size** per cycle
- The **state** of each peer is **bounded (small)**
- During interaction the **state** of one or both peers **changes** in a way that reflects the **state of the other peer**

So what is a gossip protocol (cont)?

- (random) **peer selection** from
  - the full peer set, or
  - small set of neighbors
- Reliable communication is not assumed
- The **protocol cost is negligible**
  - The frequency of interaction is much lower than message round-trip times

Example

- Everyone comes up with **a number**
- Every person periodically contacts one of their friends, **exchange** their (current) numbers, **calculate the avg** of the two and **update their number** to the avg.
  - What will happen?
  - What will happen if all choose ”0” but exept one node which chooses ”1”?
    - Network size estimation!
  - Will it always happen the same?
  - How fast will it happen?
  - What is needed?

Random peer selection

- Most of the gossip protocols work efficiently because the messages are gossiped to ***random*** peers
  - **Any idea why?**
    - **Assures connected graph**
    - **Assures fast mixing rate**
- How to select a random peer?
  - Several approaches: **Cyclon**, Newscast, SCAMP

Cyclon

- **Cyclon: Inexpensive membership management for unstructured p2p overlays (Voulgaris et al)**
  - that gives access to random peers
  - has low diameter
  - has low clustering coefficient
  - Resilient to massive node failures
    - Good expander!

properties of random graphs

Creates an Overlay that is a Random Graph

- Each node maintains only a list of addresses of some other nodes
  - Assumes unreliable communication e.g. UDP

Example Problem

- We have a population of ***N*** people
- Each person has ***c*** Bussiness cards of his/her friends
  - ***Arbitrary initial network***
- How to make sure that each person ends up with ***c*** bussiness cards of random people?
  - ***Random final network***
- **Ideas?**
  - **Problems on the way?**
    - **How to assure that the network is always connected?**
      - **Make sure the interacticng parties remain connected and do not partition the network**
    - **How to assure ”dead” links do not pollute the network/overlapping cards?**
      - **Continuously replace ”refresh” the card pool.**

Cyclon: Basic idea

- Assume we have arbitrarily connected peers
  - *We will later show how peers join/leave*
- Each peer has a neighborhood set called a cache of size **c**
- Peers periodically (after ΔT) perform a **shuffle** operation
  - Basic idea: Shuffle is an exchange of a subset of neighbors between a pair of peers

Cyclon – Shuffling at P

- Select a random subset of neighbors l_p ($l_p \subseteq N_i$) of size l
- Select a random neighbor Q from l_p
- Replace Q's address with P's address in l_p
- Send l_p to Q
- Receive l_q fro
- Update cache

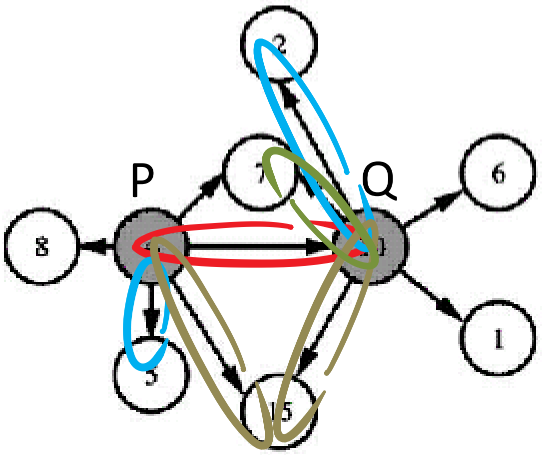
At Q on receipt of l_p

- Select random subset l_q from cache
- Send l_q to P
- Update cache to include l_p

Rules to update cache

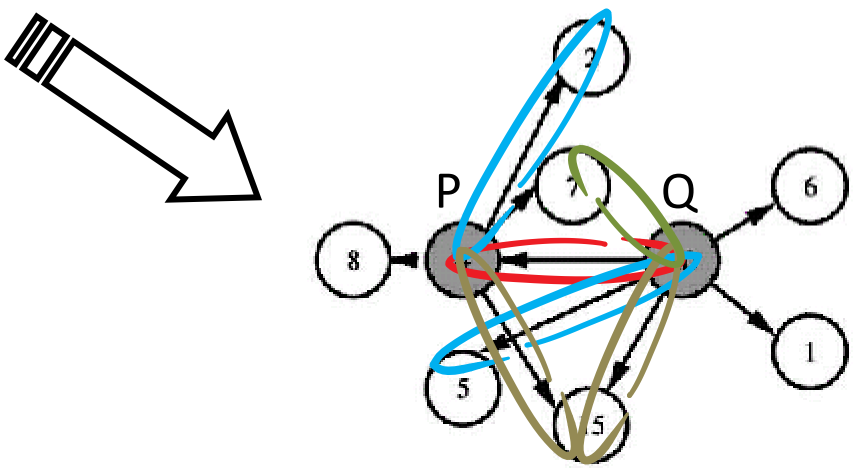
- At **P** on receiving I_q
  - – Discard entries pointing to P in I_q
    - A peer is never its own neighbor
  - – Fill any empty cache slots (if any)
  - – Replace the entries of I_p

Demonstration of shuffling



$I_p=\{4, 5, 15\}$

$I_q=\{2, 7, 15\}$



Cyclon – Shuffling

- No peer becomes disconnected
  – pointers move, so peers change from being neighbour of one peer to being the neighbour of another peer
- If P initiates a shuffle to Q, then after the shuffle
  – P becomes a neighbour of Q
  – Q is no longer neighbour of P
  – **Edge reverses direction**
- **After how many steps can we expect random graph to emerge?**

Cyclon – Enhanced Shuffling

- What is not nice?
  - Pointers to dead nodes keep getting passed around until a shuffle is performed with the dead node
    - No limit on the time until a node is chosen for shuffling
  - Can we impose a limit on the number of pointers to a node i.e. in-degree?
    - Making the graph “better” than random

Cyclon – Enhanced Shuffling (cont)

- Basic Idea:
  - Each entry in cache has an **age** associated with it
  - Age is roughly the time since the entry was initially created
  - Perform shuffling with oldest entry in cache

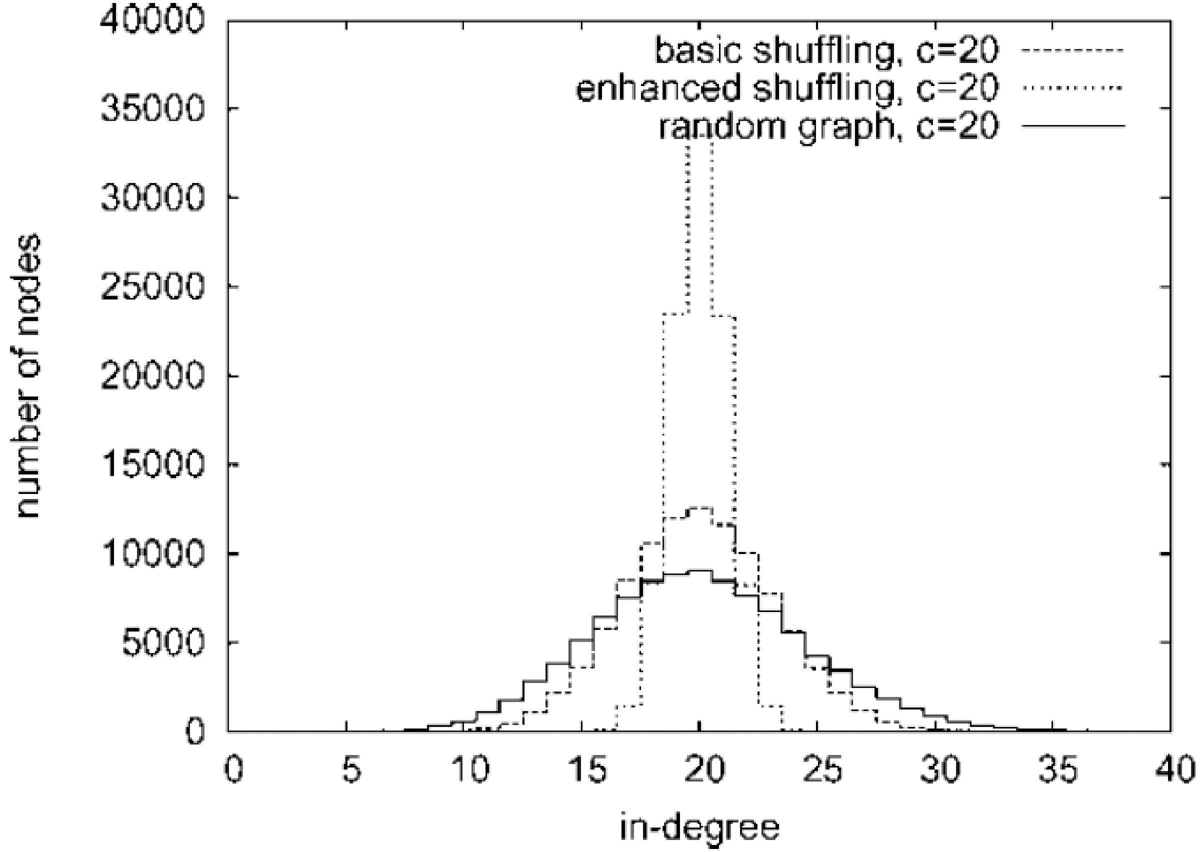
Cyclon – Enhanced Shuffling at P

- **Increment age of all cache entries**
- **l_p :**
  Select Q with highest age
- **Select *l-1* random neighbours**
- Replace Q's address with P's address in l_p
- Sends l_p to Q
- Receive l_q from Q
- Update cache to include l_q

with age 0

Degree distribution

- Degree distribution is important for



-

Cyclon – Node joins

- Property of random graphs
  - A random walk on a random (expander) graph of length at least equal to the average path length (O(logN)) is guaranteed to end at a *random* node irrespectively of the starting node
- A joining node P
  - contacts an existing node Q
  - performs **c** random walks of the expected average path length from Q
- For each random walk, the target node R performs a shuffle of length 1 with P
  - R gets P in its cache (age 0)
  - P gets the replaced entry of R

Cyclon – Node failures/leaves

- The contacted node during a shuffle is removed if it does not respond
- Such a node has higher age and therefore if it fails it will be selected and removed

Cyclon – Properties

- Think of these properties of Cyclon
  - Connectivity
  - Convergence
  - Clustering Coefficient
  - Degree distribution
  - Effect of cache size
  - Robustness and Self-healing
  - Removing dead pointers