Peer-to-Peer Networks

07 Degree Optimal Networks

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg

Diameter and Degree in Graphs

- CHORD:
  - degree O(log n)
  - diameter O(log n)
- Is it possible to reach a smaller diameter with degree g=O(log n)?
  - In the neighborhood of a node are at most g nodes
  - In the 2-neighborhood of node are at most g^2 nodes
  - ...
  - In the d-neighborhood of node are at most g^d nodes
- So,

$$(\log n)^d = n$$

- Therefore

$$d = \frac{\log n}{\log \log n}$$

- So, Chord is quite close to the optimum diameter.

Are there P2P-Netzwerke with constant out-degree and diameter log n?

- CAN
  - degree: 4
  - diameter: $n^{1/2}$
- Can we reach diameter O(log n) with constant degree?

Degree Optimal Networks

Viceroy

A Scalable and Dynamic Emulation of the Butterfly

Dahlia Malkhi, Moni Naor, David Ratajczak

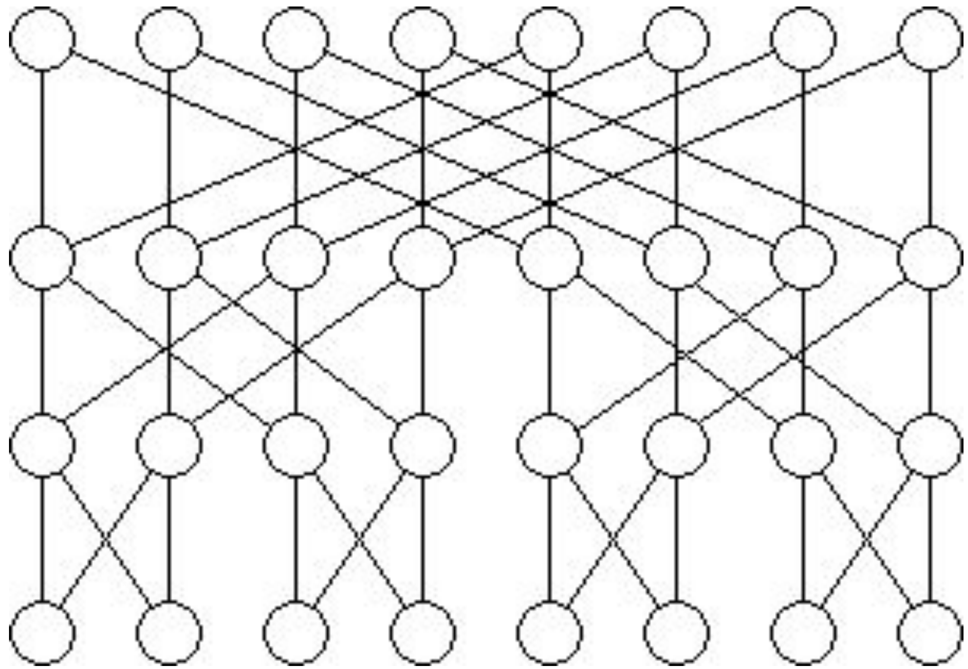
2001

Diameter and Degree

- Chord, Pastry, Tapestry
  - Diameter O(log n)
  - Degree O(log n)
- Wanted
  - Network with small degree
  - e.g. indegree and outdegree constant
  - Diameter O(log n)
- Solution:
  - Viceroy
  - Distance-Halving-Netzwerk
  - Koorde

Definition Butterfly-Graph

- Nodes: (i, S)
  - $i \in \{1,..,k\}$
  - S is k-digit binary string
- Interpretation
  - $m = 2^k$ nodes per level
  - k levels
  - Usually nodes of the k-th level are depicted twice
- Edges: From
- $(i, (b_1,..,b_i, ..., b_k))$
  - to $((i+1) \bmod k, (b_1,..,b_i, ..., b_k))$ and
  - to $((i+1) \bmod k, (b_1,..,\neg b_i, ..., b_k))$



Properties Butterfly-Graph

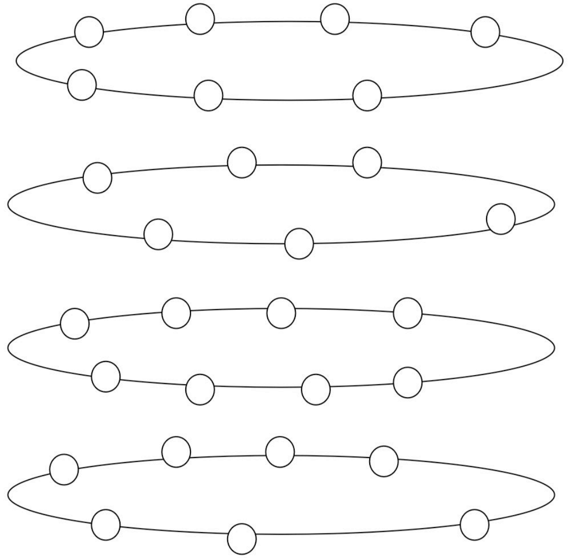
- Small degree
  - in/out - degree = 4
- Small diameter
  - diameter = 2 log m = O(log n) (optimal!)
- Good emulation properties
  - All networks can be efficiently embedded into a Butterfly-Graph
  - i.e. an edge of another network can be replaced by short paths in a Butterfly-Graph
- Simple routing
  - routing decision in constant time
- No bottlenecks
  - good routing algorithms can avoid traffic congestions in a node
- High error tolerance
  - Large number of node failures can be tolerated

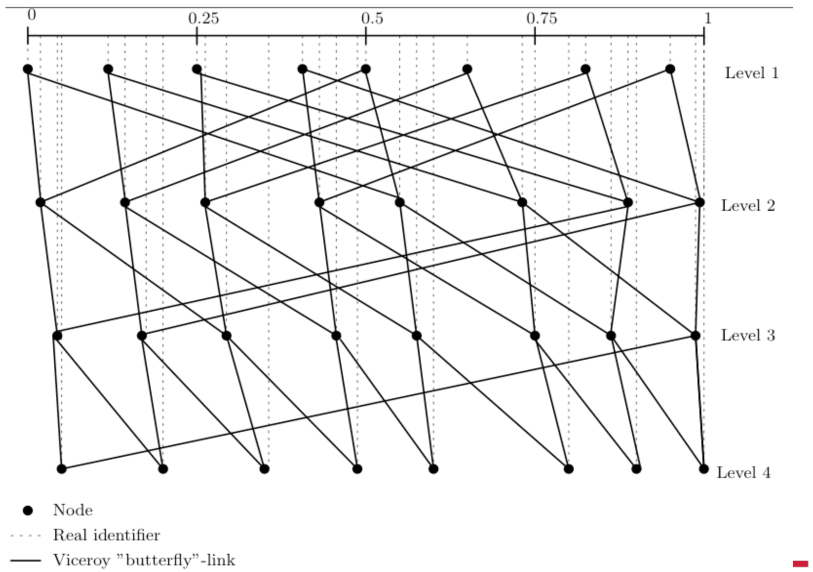
Overview Viceroy

- Goal
  - Scalability
  - Coping dynamic behavior
  - Distribution of traffic
- Congestion
  - maximum number of messages a peer needs to transport
- Costs for peers joining/leaving
  - minimize number of messages and time
- Length of a search time
  - a.k.a. dilation
- Viceroy has been the first peer-to-peer network with optimal degree-diameter relationship

Structure of Viceroy

- Nodes in Viceroy
  - at the beginning they choose a random level i of the butterfly graph and a random position x: (i,x), $i \in \{1,..,\log n\}$, $x \in [0,1)$
  - (it is necessary to know log n a-priori)
- Combination of three network structures
  - A ring for all nodes
    - connects all nodes
  - A ring for each of the log n levels
    - corresponds to the interval [0,1)
  - The Butterfly-Network between layers
    - i.e. in level i is a link from (i,x) to the
      - successor of $(i+1,x)$
      - successor of $(i+1,x+2^{-i} \bmod 1)$
      - predecessor of $(i-1, x)$
      - predeccessor of $(i-1, x-2^{-i} \bmod 1)$





Computation of log n

- Consider neighbor in a ring
- Let d be the distance on the ring [0,1) to this neighbor
- Then:
  - $E[d] = 1/n$
  - $P[d > c (\log n)/n] < n^{-O(1)}$
  - $P[d < 1/n^{1+c}] < n^{-c}$
- Therefore -log d is a constant factor approximation of log n with with probability.
- If one measures the average distance to the next O(log n) neighbors, then the average is a good approximation of log n with high probability.

Inserting Peers

1. Insert node at an arbitrary position in the overall connecting ring

2. Estimate log n

3. Choose random level i uniformly from {1,..,log n}

4. Look up position in ViceRoy network starting from neighbor in the ring

5. Insert peer in the level of the ViceRoyn network as follows

- Adjust links from and to neighbors of the ring i
- Adjust pointers from (i,x) to
  - successor of (i+1,x)
  - successor of (i+1,x+2^{-i})
  - predeccessor of (i-1, x)
  - predeccessor of (i-1, x-2^{-i})
- Adjust pointers from the levels i, i-1, i+1 towards this new node.

➢ Run time / number of messages

- Lookup: (O(log n)) +
- Finding successors or predecessors (O(log n))

Lookup

- Peer (i,x) receives lookup request towards (j,y)

IF i=j and |x-y| ≤ $(\log n)^2/n$ THEN

    Forward lookup request to ring neighbor in level i

ELSE

    IF y farther to the right than $x+2^i$ THEN

        Forward lookup request to sucessor of $(i+1,x+2^i)$

    ELSE

        Forward lookup request to sucessor of Z= (i+1,x)

        IF successor Z is farther to the right than x THEN

            Search the node (i+1,p) on the ring (i+1) starting from Z such that p<x

        FI

    FI

FI

Lemma

With high probability the lookup takes O(log n) time and messages.

Properties of ViceRoy

- Outdegree constant
- Expected indegree constant
  - worst case indegree O(log n)
- Diameter: O(log n) w.h.p.
- Communication can be balanced by the Butterfly-Graph

Talking about the Degree

- outdegree: 2+2+2+2 = 8
- If the distribution is perfect, i.e. $\Theta(1/n)$
  - then also the indegree ist constant
- But
  - Indegree is only constant in the expectation
  - yet, $\Omega(\log n)$ may occur
- Problem
  - Large distances to neighbors on a Viceroy ring attract a lot of incoming pointers
  - Small distances „spam“ peers on neihbored rings
- Solution:
  - Do not use hashing, but use the principle of multiple choice (see later)

Summary ViceRoy

- Butterfly graph
  - well suited for routing
  - often used and well known algorithms
- First peer-to-peer with constant (out-) degree
- But:
  - Multiple ring structure is complex
  - Inserting takes time O(log n)

Distance Halving

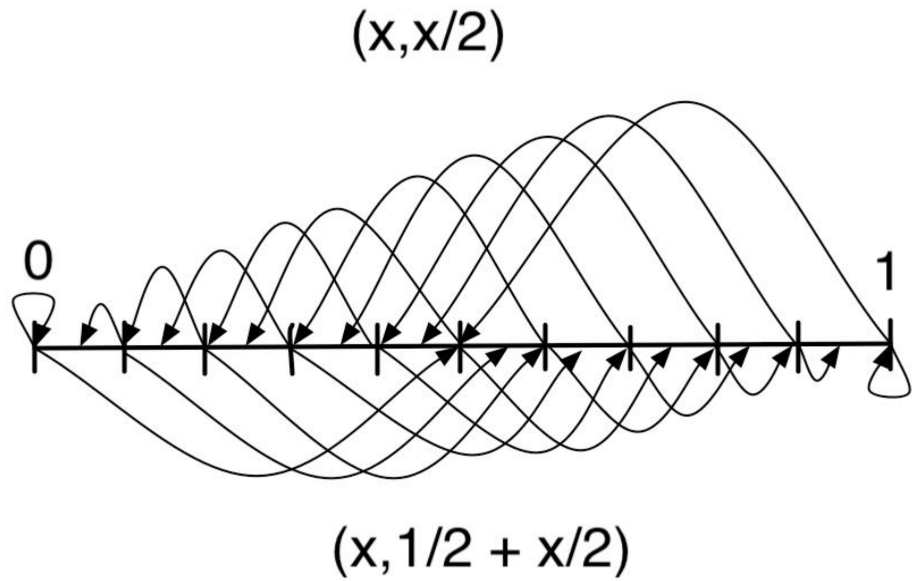
Moni Naor, Udi Wieder

2003

Continuous Graphs

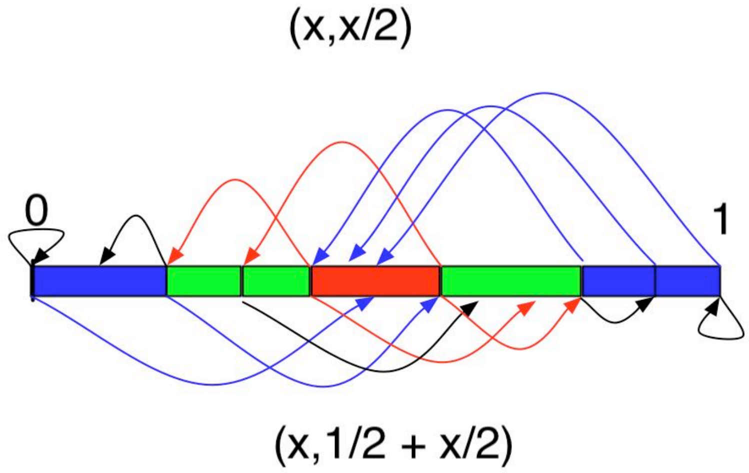
- are infinite graphs with continuous node sets and edge sets

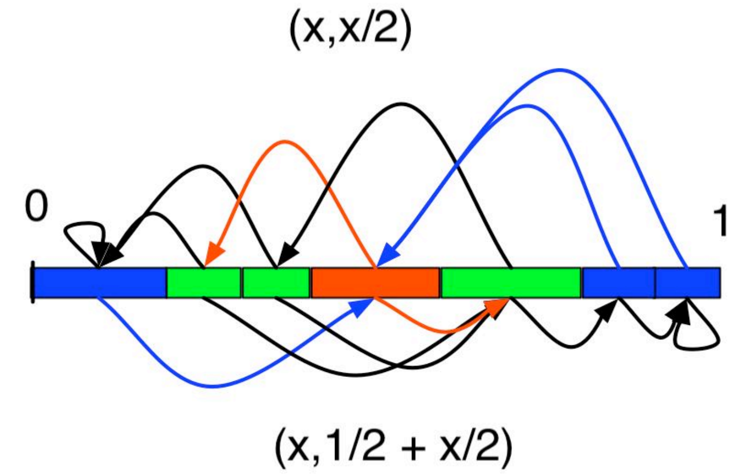
- The underlying graph
  - $x \in [0,1)$
  - Edges:
    - $(x,x/2)$, *left* edges
    - $(x,1+x/2)$, *right* edges
  - plus revers edges.
    - $(x/2,x)$
    - $(1+x/2,x)$



The Transition from Continuous to Discrete Graphs

- Consider discrete intervals resulting from a partition of the continuous space
- Insert edge between interval A and B
  - if there exists $x \in A$ and $y \in B$ such that edge (x,y) exists in the continuous graph
- Intervals result from successive partitioning (halving) of existing intervals
- Therefore the degree is constant if
  - the ratio between the size of the largest and smallest interval is constant
- This can be guarranteed by the principle of multiple choice
  - which we present later on





Principle of Multiple Choice

- **Before inserted check c log n positions**
- **For position p(j) check the distance a(j) between potential left and right neighbor**
- **Insert element at position p(j) in the middle between left and right neighbor, where a(j) was the maximum choice**
- **Lemma**
  - After inserting n elements with high probability only intervals of size 1/(2n), 1/n und 2/n occur.

Proof of Lemma

1st Part: With high probability there is no interval of size larger than 2/n

follows from this Lemma

Lemma*

Let c/n be the largest interval. After inserting 2n/c peers all intervals are smaller than c/(2n) with high probability

From applying this lemma for c=n/2,n/4, ...,4 the first lemma follows.

Proof

- **2nd part: No intervals smaller than 1/(2n) occur**
  - The overall length of intervals of size 1/(2n) before inserting is at most 1/2
  - Such an area is hit with probability at most 1/2
  - The probability to hit this area more than c log n times is at least

$$2^{-c \log n} = n^{-c}$$

  - Then for c>1 such an interval will not further be divided with probability into an interval of size 1/(4m).

Chernoff-Bound

- Theorem Chernoff Bound
  - Let $x_1,...,x_n$ independent Bernoulli experiments with
    - $P[x_i = 1] = p$
    - $P[x_i = 0] = 1-p$
  - Let

$$S_n = \sum_{i=1}^{n} x_i$$

  - Then for all c>0

$$\mathbf{P}[S_n \geq (1+c) \cdot \mathbf{E}[S_n]] \quad \leq \quad e^{-\frac{1}{3}\min\{c,c^2\}pn}$$

  - For 0≤c≤1

$$\mathbf{P}[S_n \leq (1-c) \cdot \mathbf{E}[S_n]] \quad \leq \quad e^{-\frac{1}{2}c^2 pn}$$

Proof of Lemma*

- Consider the longest interval of size c/n. Then after inserting 2n/c peers all intervals are smallver than c/(2n) with high probability.
- Consider an interval of length c/n
- With probability c/n such an interval will be hit
- Assume, each peer considers t log n intervals
- The expected number of hits is therefore

$$E[X] = \frac{c}{n} \cdot \frac{2n}{c} \cdot t \log n = 2t \log n$$

- From the Chernoff bound it follows

$$P[X \leq (1-\delta)E[X]] \leq n^{-\delta^2 t}$$

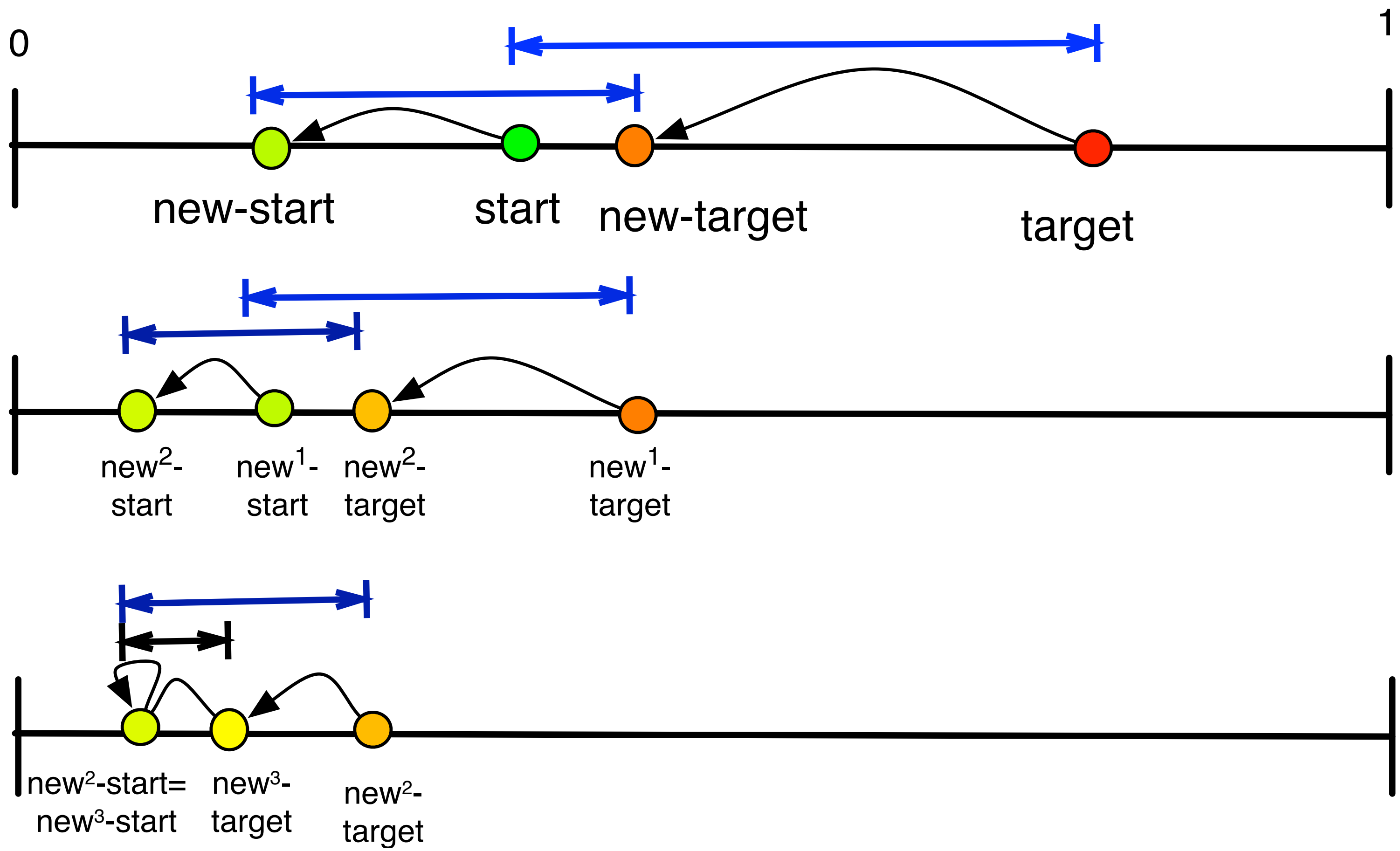
- If $\delta^2 t \geq 2$ then this interval will be hit at least $2(1-\delta)t \log n$ times
- Choose $2(1-\delta) \geq 1$

$$\delta \geq \frac{1}{2} \qquad t \leq \frac{1}{2}\delta^2$$

- Then, every interval is partitioned w.h.p.

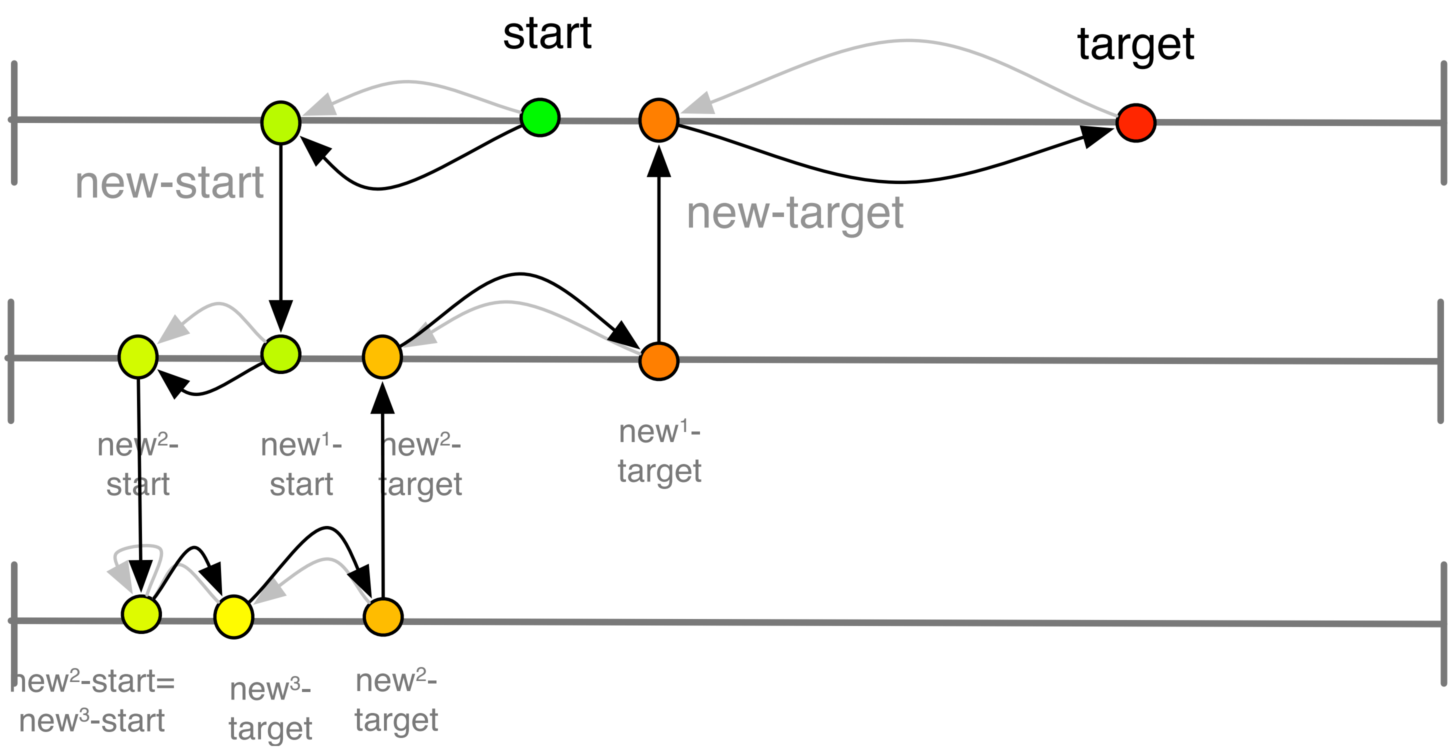
Lookup in Distance-Halving

- Map start/target to new-start/ target by using left edges
- Follow all left edges for 2+ log n steps
- Then, the new-new...-new-start and the new-new-...new-target are neighbored.



Lookup in Distance-Halving

- Follow all left edges for 2+ log n steps
  - Use neighbor edge to go from new*-start to new*-target
- Then follow the reverse left edges from new^{m+1}-target to new^{m}-target

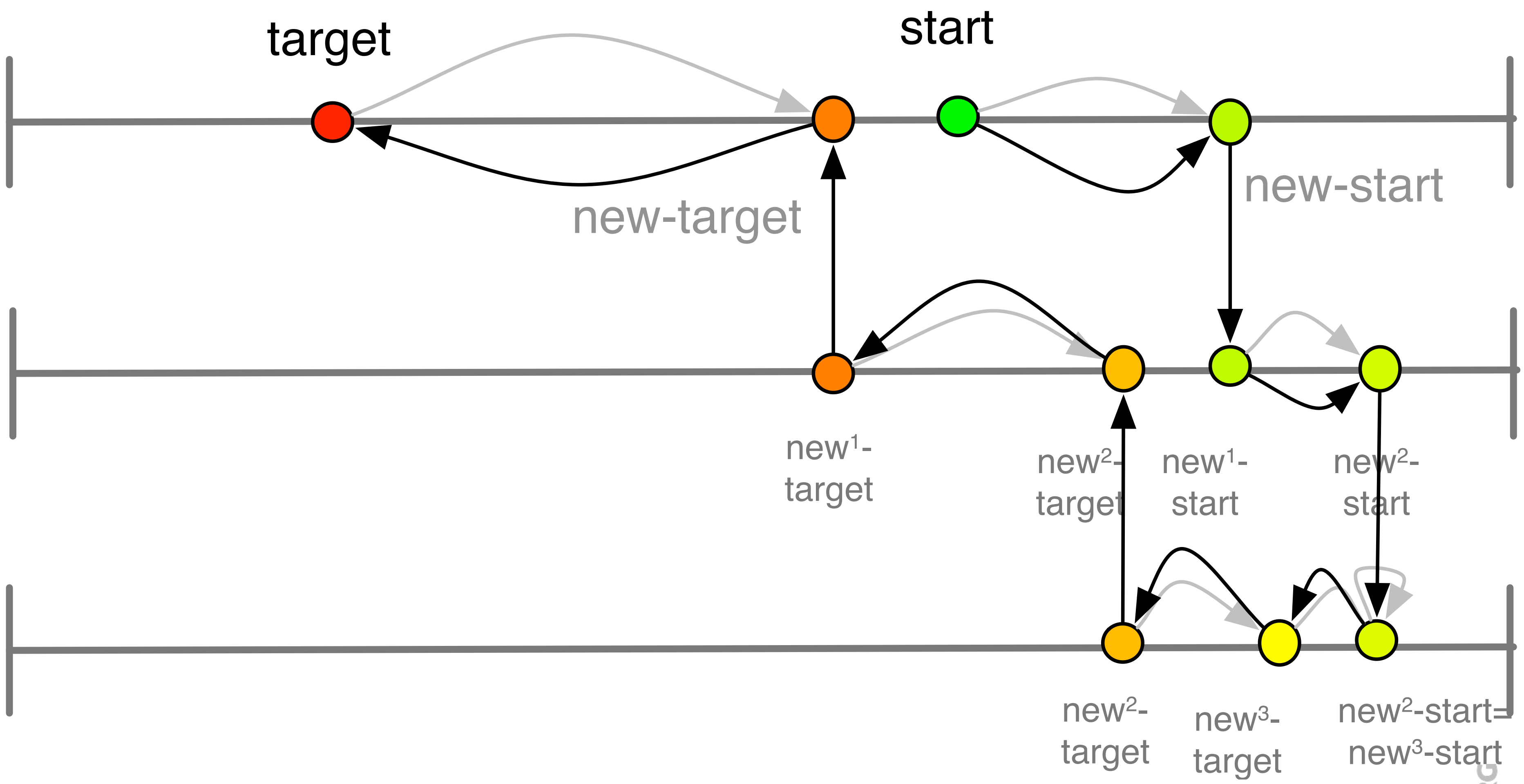


Structure of Distance-Halving

- Peers are mapped to the intervals
  - uses DHT for data
- Additional neighbored intervals are connected by pointers
- The largest interval has size 2/n w.h.p.
  - i.e. probability $1-n^{-c}$ for some constant c
- The smallest interval size 1/(2n) w.h.p.
- Then the indegree and outdegree is constant
- Diameter is O(log n)
  - which follows from the routing

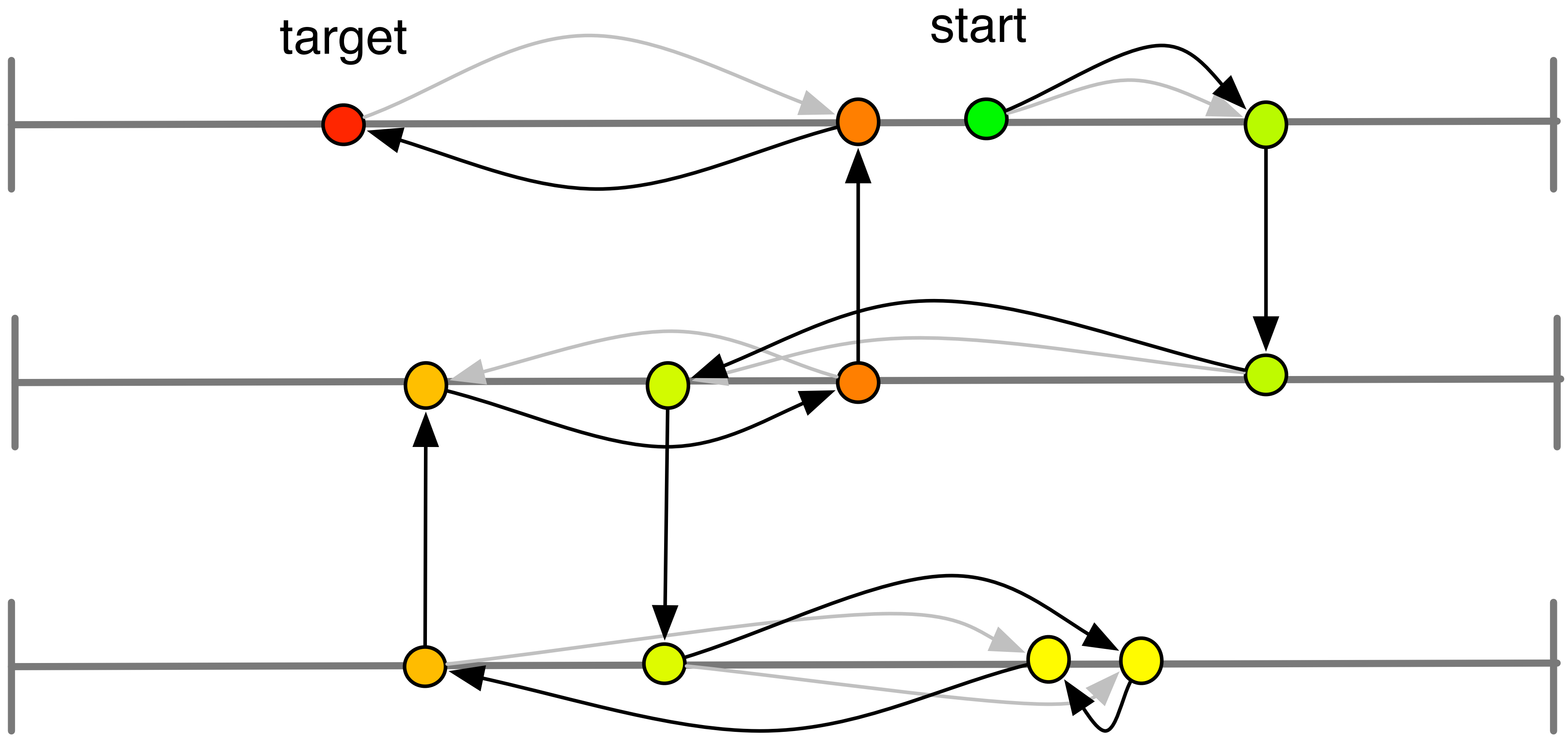
Lookup in Distance-Halving

- This works also using only right edges



Lookup in Distance-Halving

- This works also using a mixture of right and left edges



Congestion Avoidance during Lookup

- Left and right-edges can be used in any ordering
  - if one stores the combination for the reverse edges
- Analog to Valiant‘s routing result for the hyper-cube one can show
- The congestion ist at most O(log n),
  - i.e. every peer transports at most a factor of O(log n) more packets than any optimal network would need
- The same result holds for the Viceroy network

Inserting peers in Distance-Halving

1. Perform multiple choice principle
   - i.e. c log n queries for random intervals
   - Choose largest interval
   - halve this interval
2. Update ring edges
3. Update left and right edges
   - by using left and right edges of the neighbors

Lemma

Inserting peers in Distance Halving needs at most $O(\log^2 n)$ time and messages.

Summary Distance-Halving

- Simple and efficient peer-to-peer network
  - degree O(1)
  - diameter O(log n)
  - load balancing
  - traffic balancing
  - lookup complexity O(log n)
  - insert $O(\log^2 n)$

- We already have seen continuous graphs in other approaches
  - Chord
  - CAN
  - Koorde
  - ViceRoy

Degree Optimal Networks

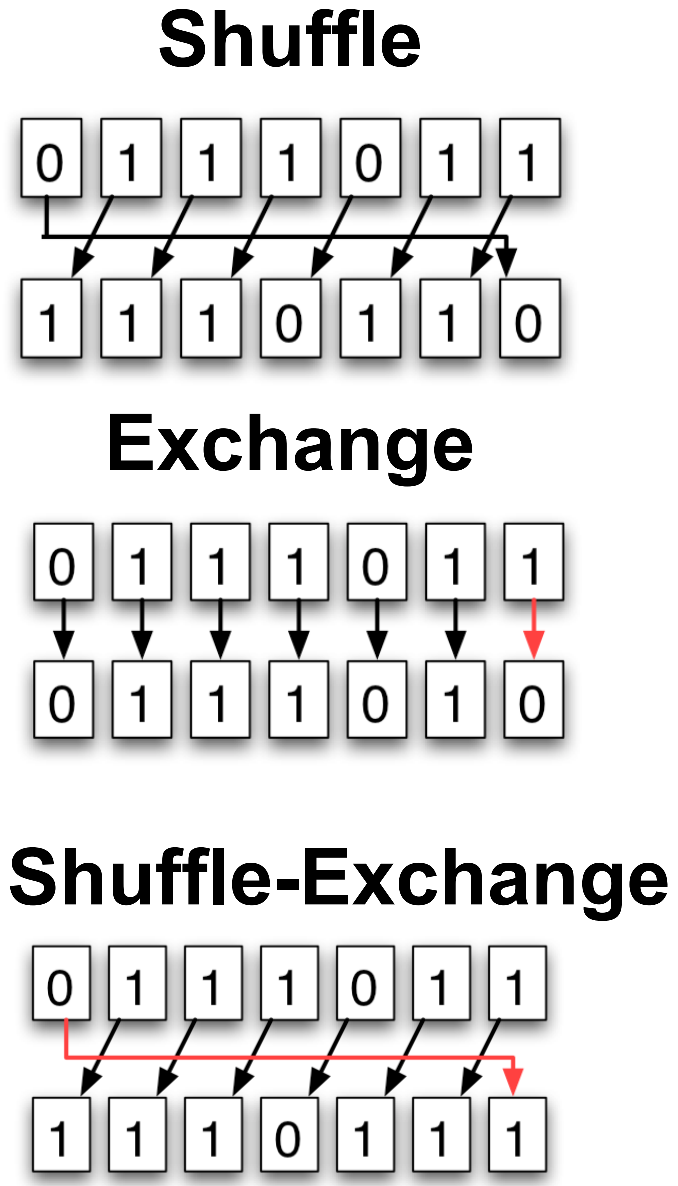
Koorde

M. Frans Kaashoek and David R. Karger 2003

Shuffle, Exchange, Shuffle-Exchange

- Consider binary string s of length m
  - shuffle operation:
    - $\text{shuffle}(s_1, s_2, s_3, \ldots, s_m) = (s_2, s_3, \ldots, s_m, s_1)$
  - exchange:
    - $\text{exchange}(s_1, s_2, s_3, \ldots, s_m) = (s_1, s_2, s_3, \ldots, \neg s_m)$
  - shuffle exchange:
    - $SE(S) = \text{exchange}(\text{shuffle}(S)) = (s_2, s_3, \ldots, s_m, \neg s_1)$
- Observation:

  Every string a can be transformed into a string b by at most m shuffle and shuffle exchange operations



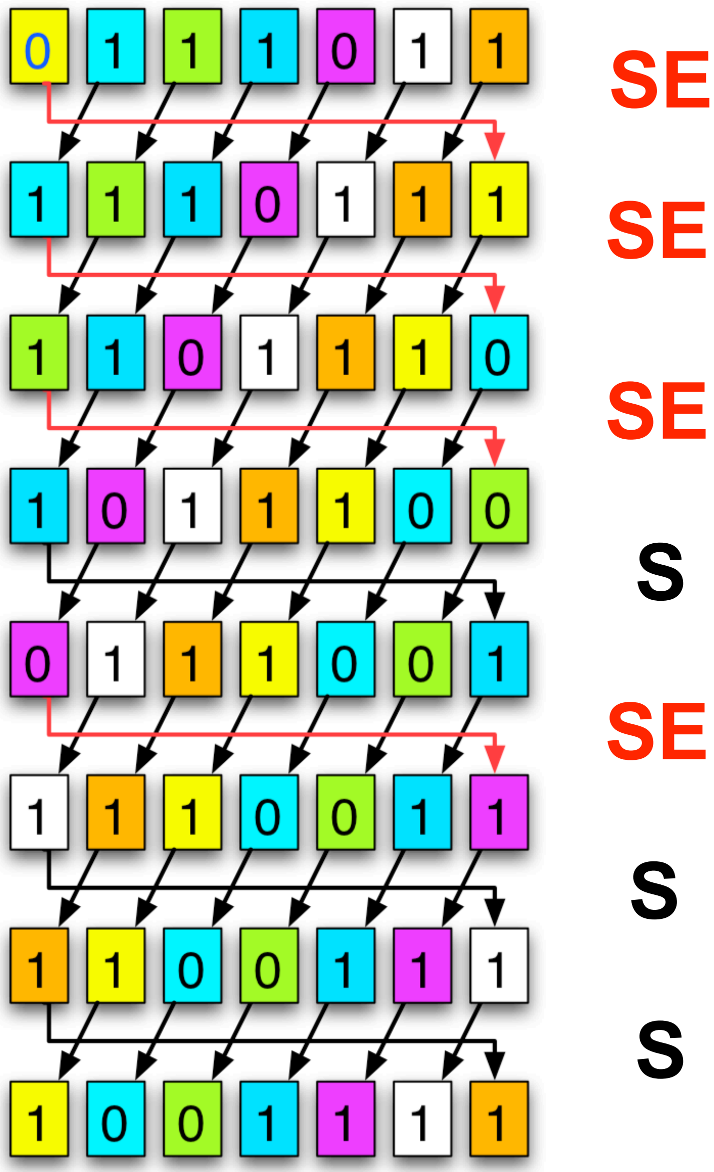
Magic Trick

- Observation

  Every string a can be transformed into a string b by at most m shuffle and shuffle exchange operations Beispiel:

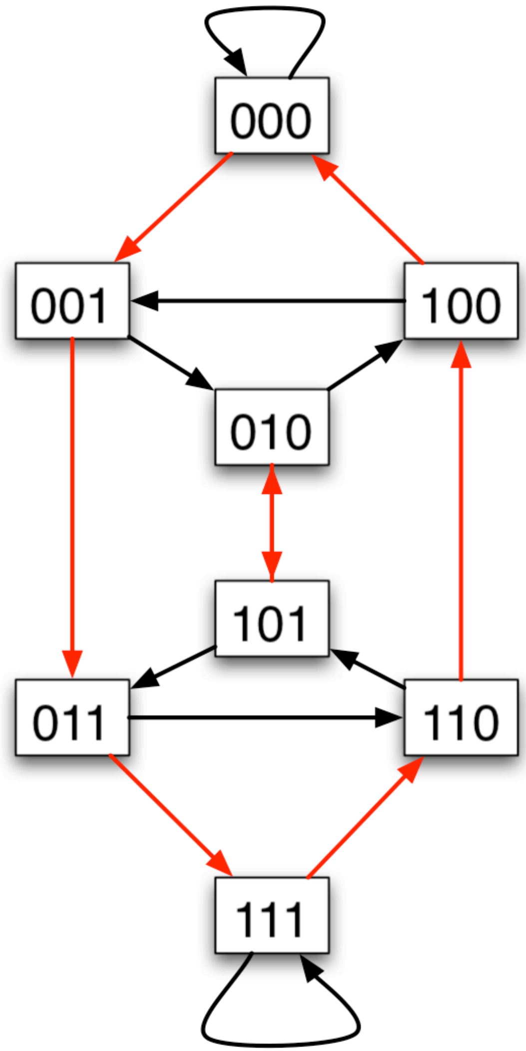
From	0	1	1	1	0	1	1
to	1	0	0	1	1	1	1
via	SE	SE	SE	S	SE	S	S

operations



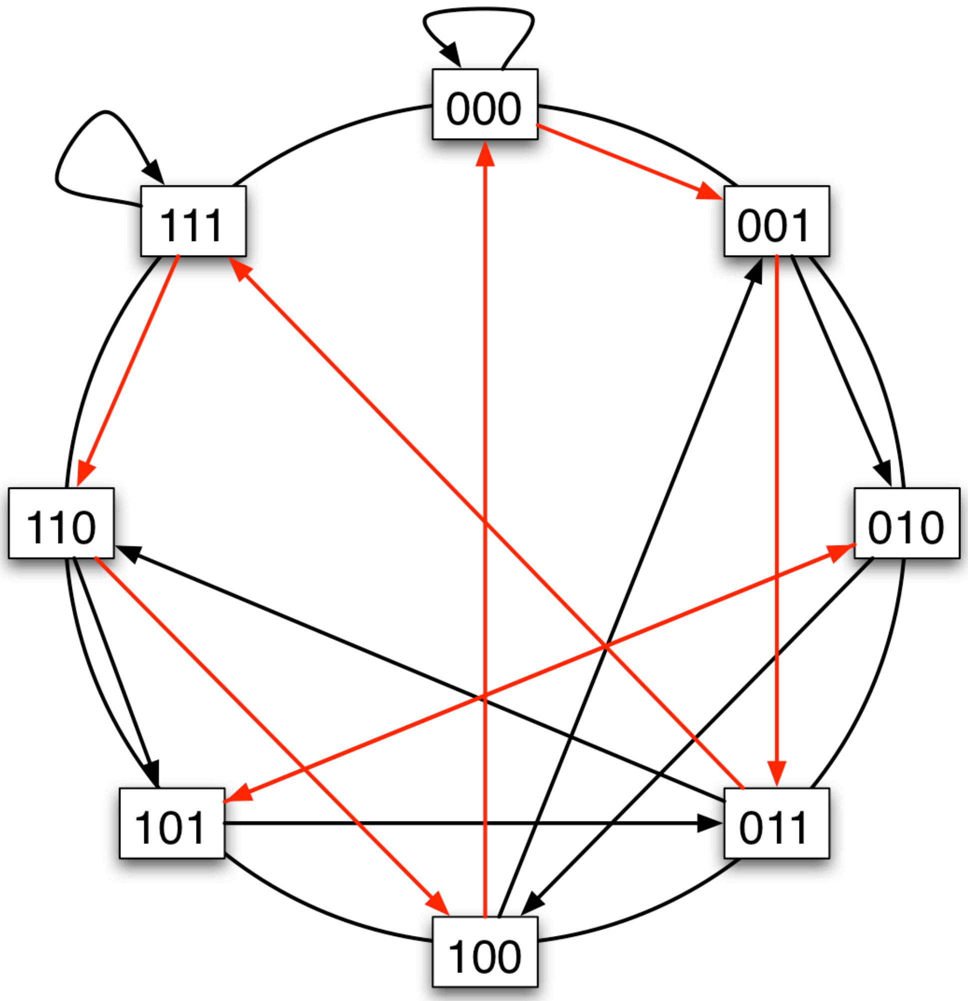
The De Bruijn Graph

- A De Bruijn graph consists of n=2m nodes,
  - each representing an m digit binary strings
- Every node has two outgoing edges
  - (u,shuffle(u))
  - (u, SE(u))
- Lemma
  - The De Bruijn graph has degree 2 and diameter log n
- Koorde = Ring + DeBruijn-Graph



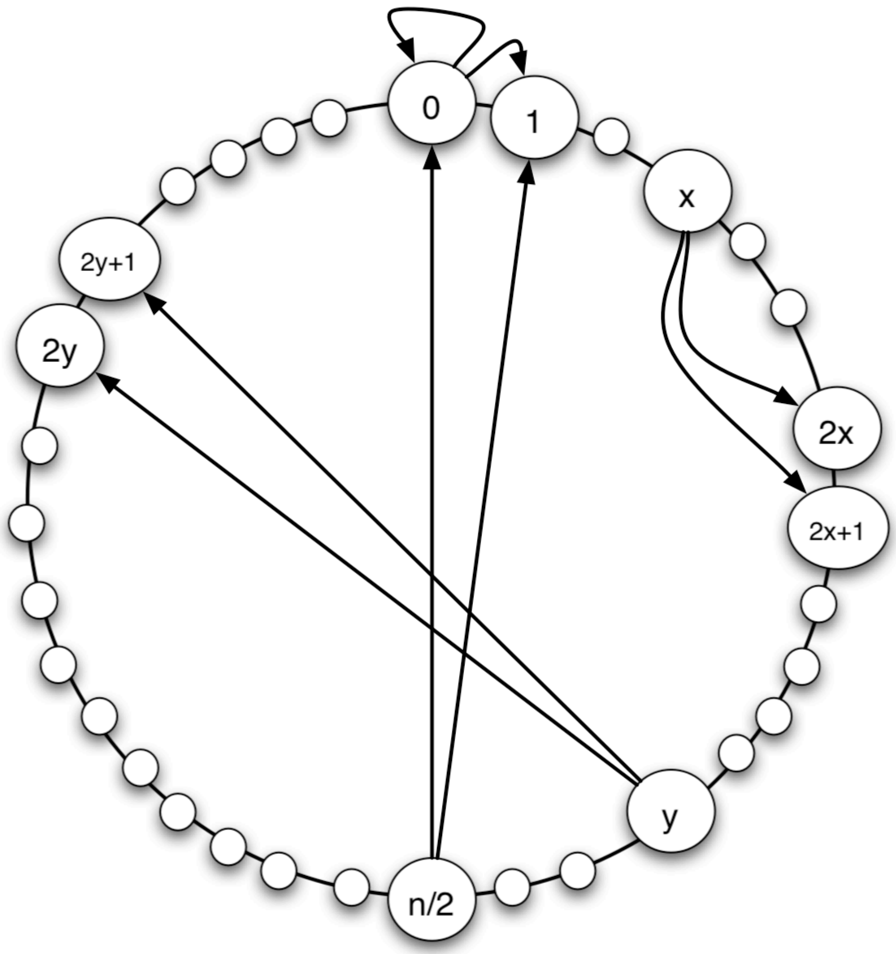
Koorde = Ring + DeBruijn-Graph

- Consider ring with 2^m nodes and De Bruijn edges



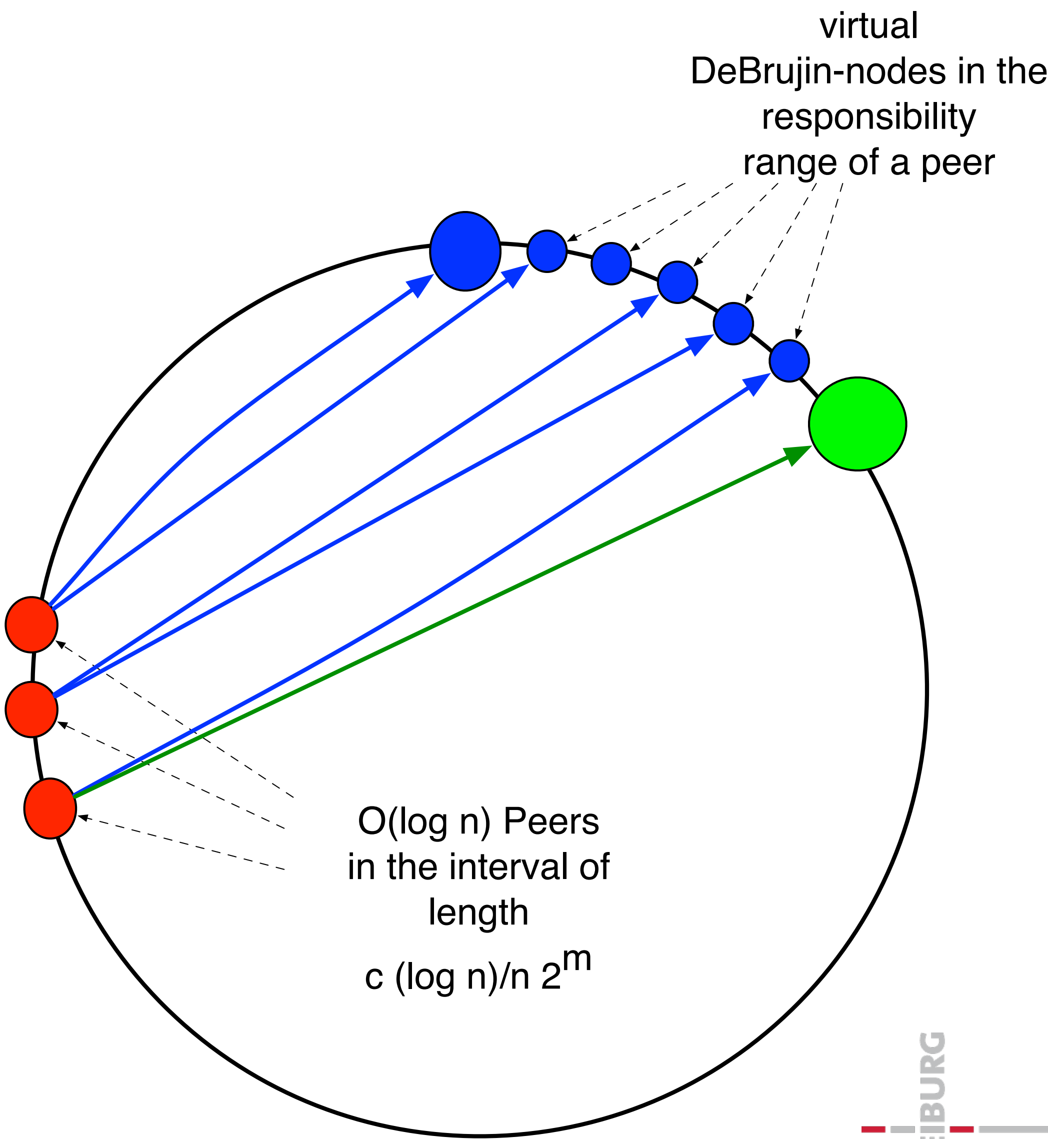
Koorde = Ring + DeBruijn-Graph

- Note
  - shuffle($s_1, s_2, \ldots, s_m$) = ($s_2, \ldots, s_m, s_1$)
    - shuffle (x) = (x div 2^{m-1})+(2x) mod 2^m
  - SE(S) = ($s_2, s_3, \ldots, s_m, \neg s_1$)
    - SE(x) = 1-(x div 2^{m-1})+(2x) mod 2^m
  - Hence: Then neighbors of x are
    - 2x mod 2^m and
    - 2x+1 mod 2^m



Virtual DeBruijn Nodes

- To avoid collisions we choose
  - m > (2+c) log (n)
- Then the probability of two peers colliding is at most n^{-c}
- But then we have much mor nodes in the graph than peers in the network
- Solution
  - Every peer manages all DeBruijn nodes between his position and his successor on the ring
  - only for incoming edges
  - outgoing edges are considered only from the peer's poisition on the ring



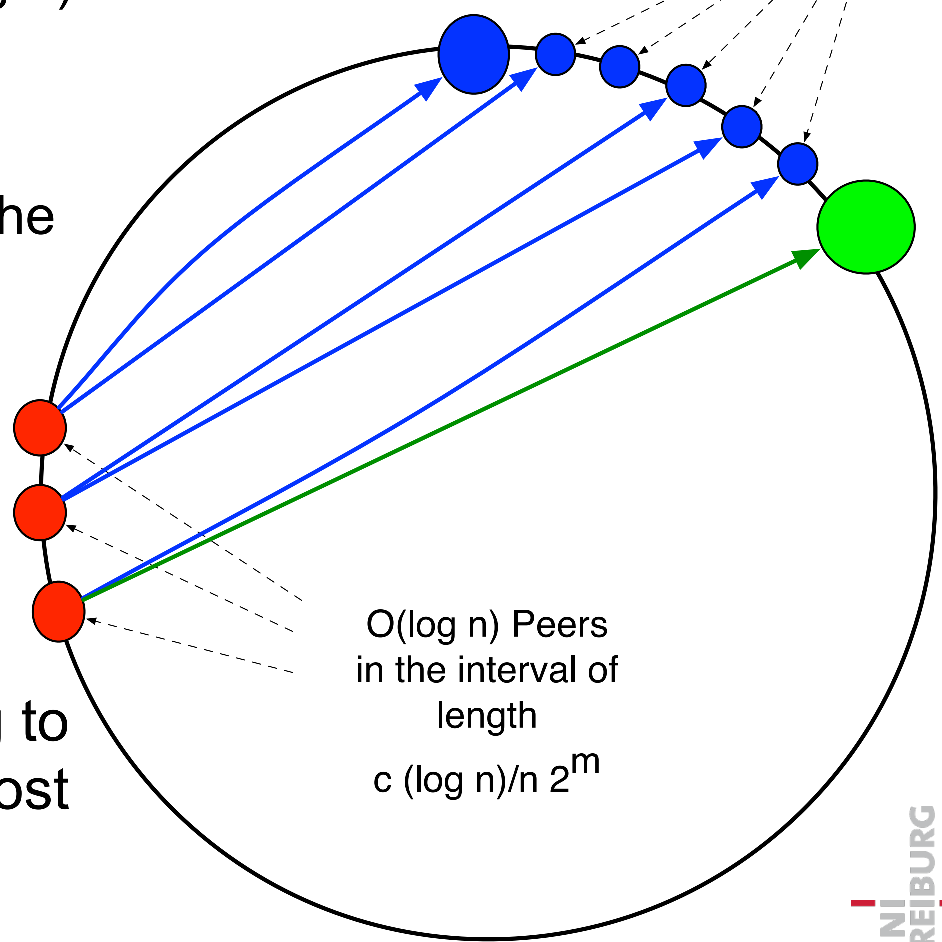
Properties of Koorde

- Theorem
  - Every node has four pointers
  - Every node has at most O(log n) incoming pointers w.h.p.
  - The diameter is O(log n) w.h.p.
  - Lookup can be performed in time O(log n) w.h.p.

- But:
  - Connectivity of the network is very low.

Properties of Koorde

- Theorem
  - 1. Every node has four pointers
  - 2. Every node has at most O(log n) incoming pointers w.h.p.
- Proof:
  - 1. follows from the definition of the DeBruijn graph and the observation that only non-virtual nodes have outgoing edges
  - 2. The distance between two peers is at most c (log n)/n 2^m with high probability
  - The number of nodes pointing to this distance is therefore at most c (log n) with high probability

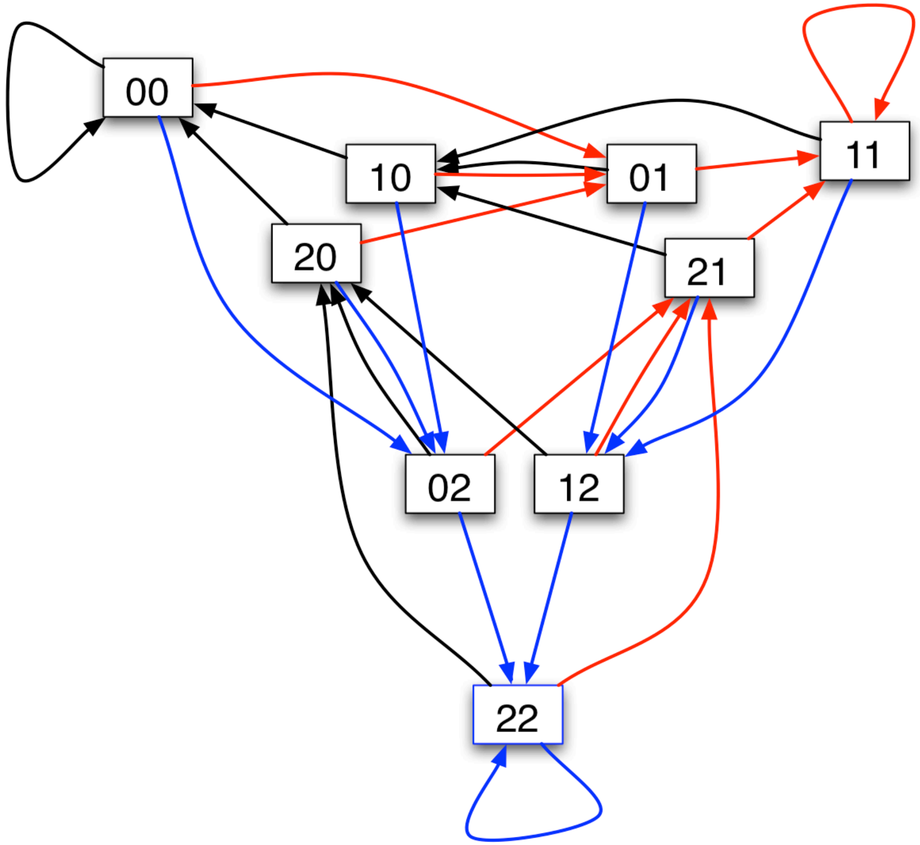


Properties of Koorde

- Theorem
  - The diameter is O(log n) w.h.p.
  - Lookup can be performed in time O(log n) w.h.p.
- Proof sketch:
  - The minimal distance of two peers is at least $n^{-c} 2^m$ w.h.p.
  - Therefore use only the c log n most significant bits in the routing
    - since the prefix guarantees that one end in the responsibility area of a peer
  - Follow the routing algorithm on the De-Bruijn-graph until one ends in the responsibility area of a peer

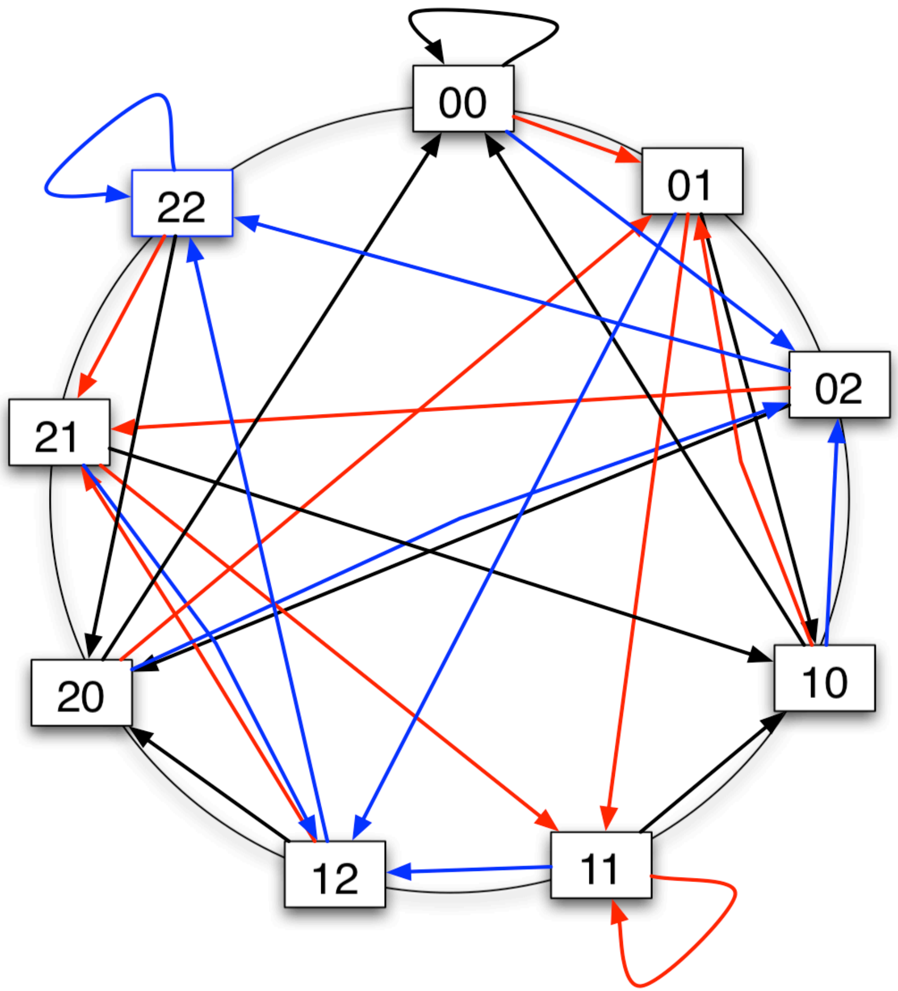
Degree k-DeBruijn-Graph

- Consider alphabet using k letters, e.g. k = 3
- Now, every k-DeBruijn-node has successors
  - (kx mod km)
  - (kx +1 mod km)
  - (kx+2 mod km)
  - ... (kx+k-1 mod km)
- Diameter is reduced to
  - (log m)/(log k)
- Graph connectivity is increased to k



k-Koorde

- Straight-forward generalization of Koorde
  - by using k-DeBruijn graphs
- Improves lookup time and messages to O((log n)/(log k)) steps



Peer-to-Peer Networks
07 Degree Optimal Networks

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg