Peer-to-Peer Networks

DHT & CAN
2nd Week

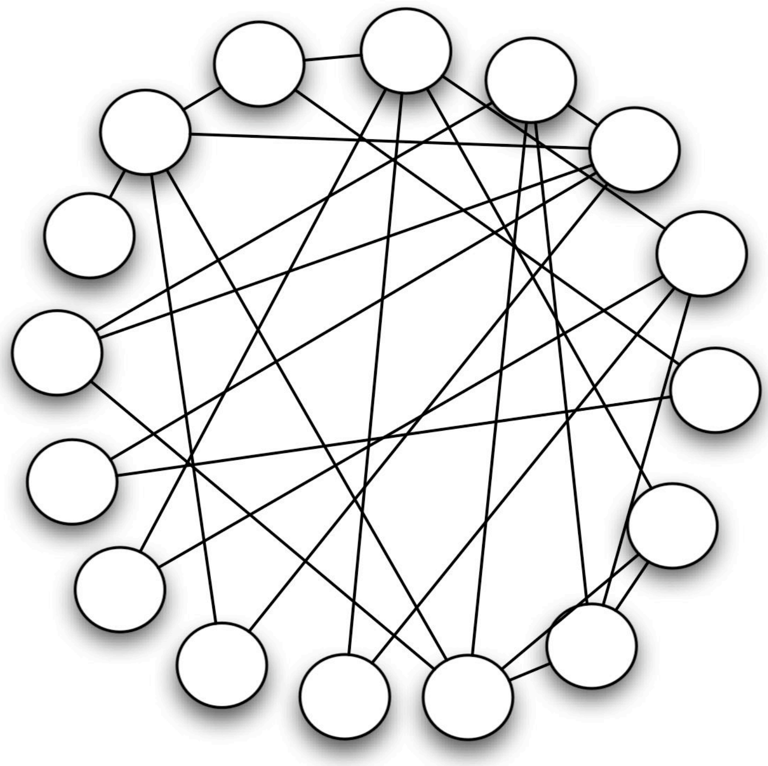
Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer
Summer 2008

Peer-to-Peer Networks

Distributed Hash Tables (DHT)

Why Gnutella Does Not Really Scale

- **Gnutella**
  - graph structure is random
  - degree of nodes is small
  - small diameter
  - strong connectivity
- **Lookup is expensive**
  - for finding an item the whole network must be searched
- **Gnutella‘s lookup does not scale**
  - reason: no structure within the index storage

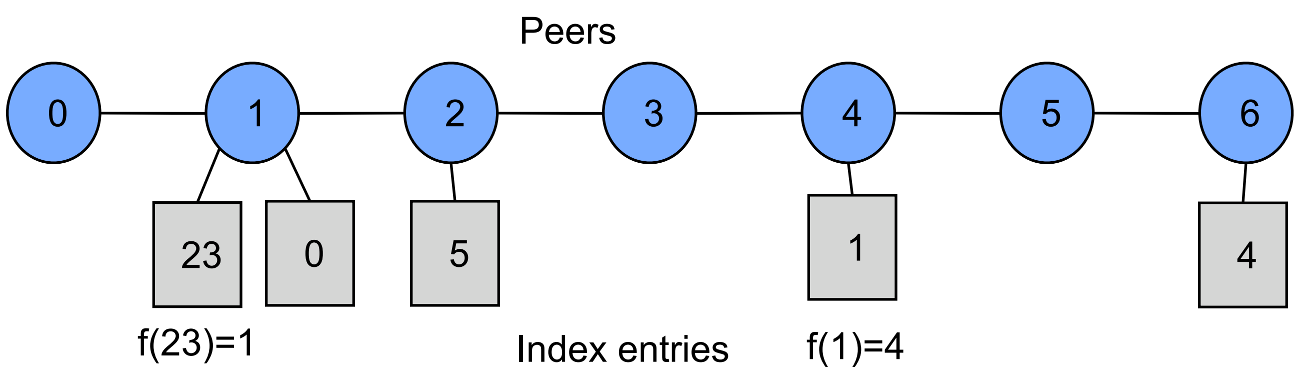


Two Key Issues for Lookup

- **Where is it?**
- **How to get there?**
- **Napster:**
  - Where? on the server
  - How to get there? directly
- **Gnutella**
  - Where? don‘t know
  - How to get there? don‘t know
- **Better:**
- **Where is x?**
  - at f(x)
- **How to get there?**
  - all peers know the route

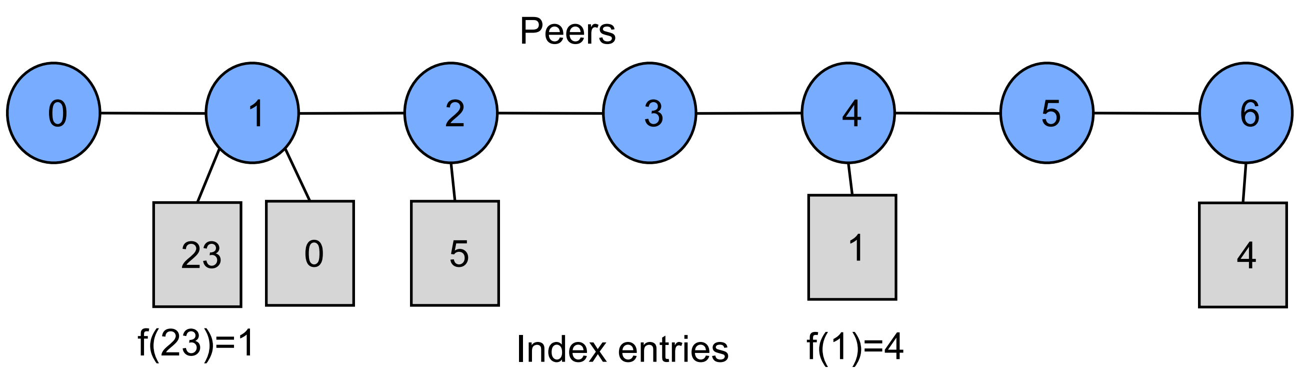
(Bad) Idea: Use Hashing

- **Give each of n peers a number 0,1,..,n-1**
  - use hash function
    - e.g. f(x) = (3x+1 mod 23) mod 7
  - peers are connected on a chain
- **Lookup**
  - compute f(x)
  - forward message to f(x) along the chain



Problems with Pure Hashing

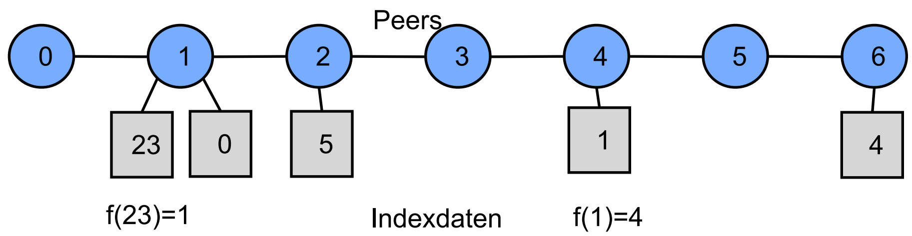
- **Insert and deletion of peers critical**
  - if a peer leaves without warning then network breaks up
  - inserting a peer implies readjusting the whole entries
    - hash function must be changed to new version
  - how to assign the numbers to peers?
- **Lookup is not efficient**
  - takes linear time on the average
  - the peers in the middle see 50% of all lookups



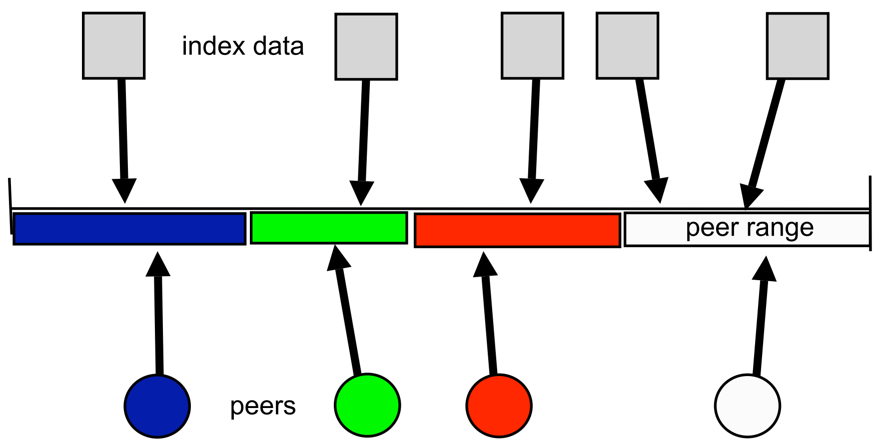
Distributed Hash-Table (DHT)

- **Hash table**
  - does not work efficiently for inserting and deleting
- **Distributed Hash-Table**
  - peers are „hashed“ to a position in an continuos set (e.g. line)
  - index data is also „hashed“ to this set
- **Mapping of index data to peers**
  - peers are given their own areas depending on the position of the direct neighbors
  - all index data in this area is mapped to the corresponding peer
- **Literature**
  - *“Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web”,* David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, Rina Panigrahy, STOC 1997

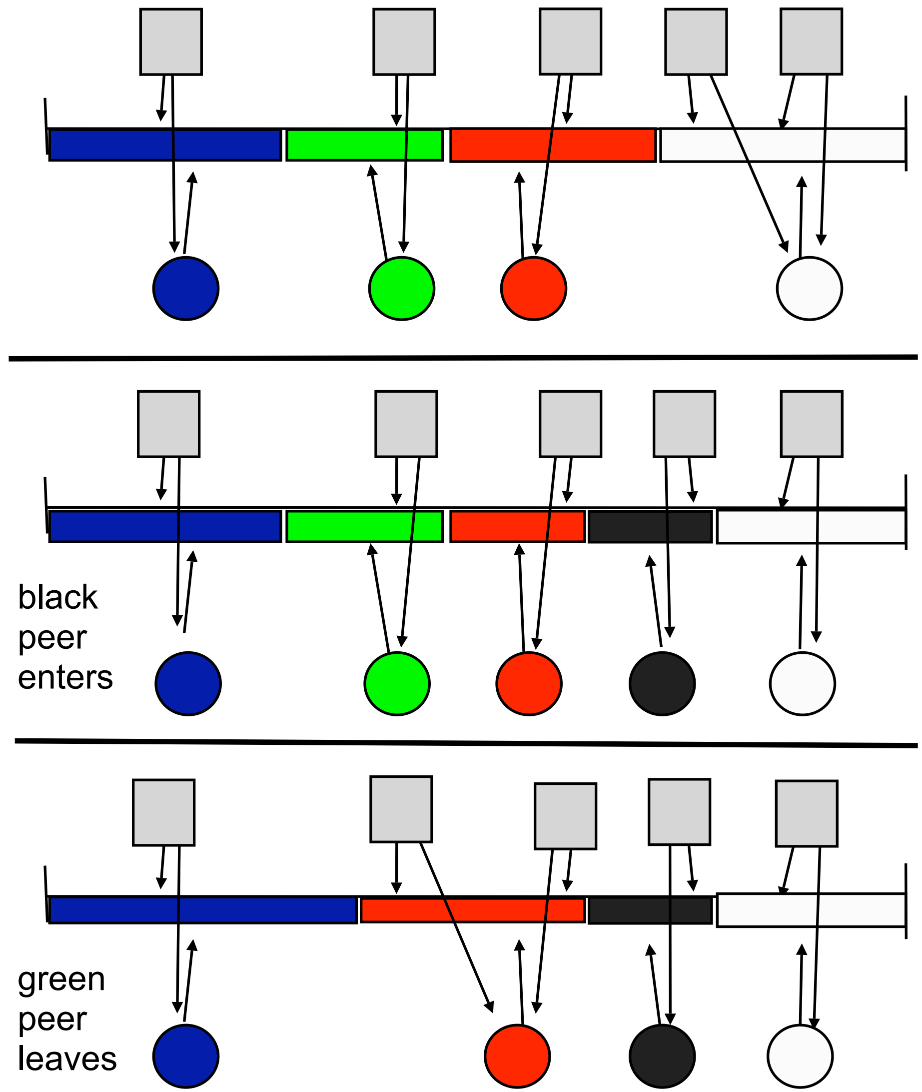
Pure (Poor) Hashing



DHT



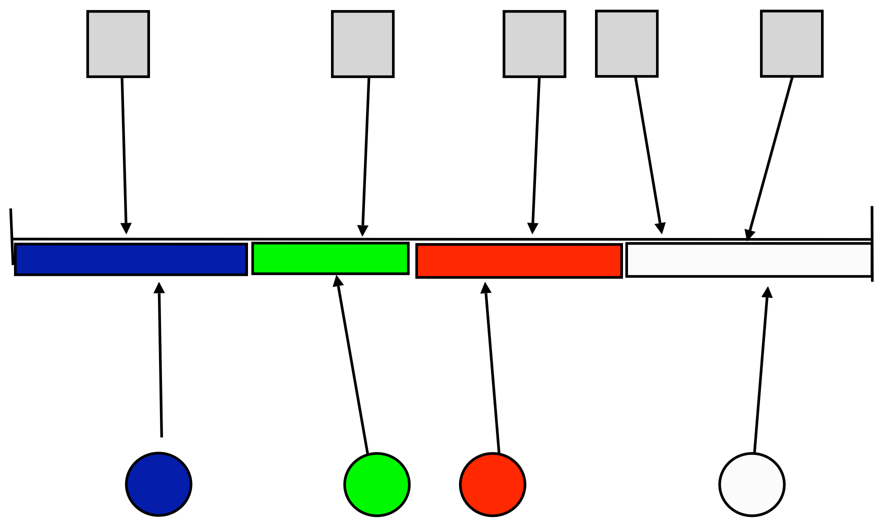
Entering and Leaving a DHT



- **Distributed Hash Table**
  - peers are hashed to to position
  - index files are hashed according to the search key
  - peers store index data in their areas
- **When a peer enters**
  - neighbored peers share their areas with the new peer
- **When a peer leaves**
  - the neighbors inherit the responsibilities for the index data

Features of DHT

- **Advantages**
  - Each index entries is assigned to a specific peer
  - Entering and leaving peers cause only local changes
- **DHT is the dominant data struction in efficient P2P networks**
- **To do:**
  - network structure



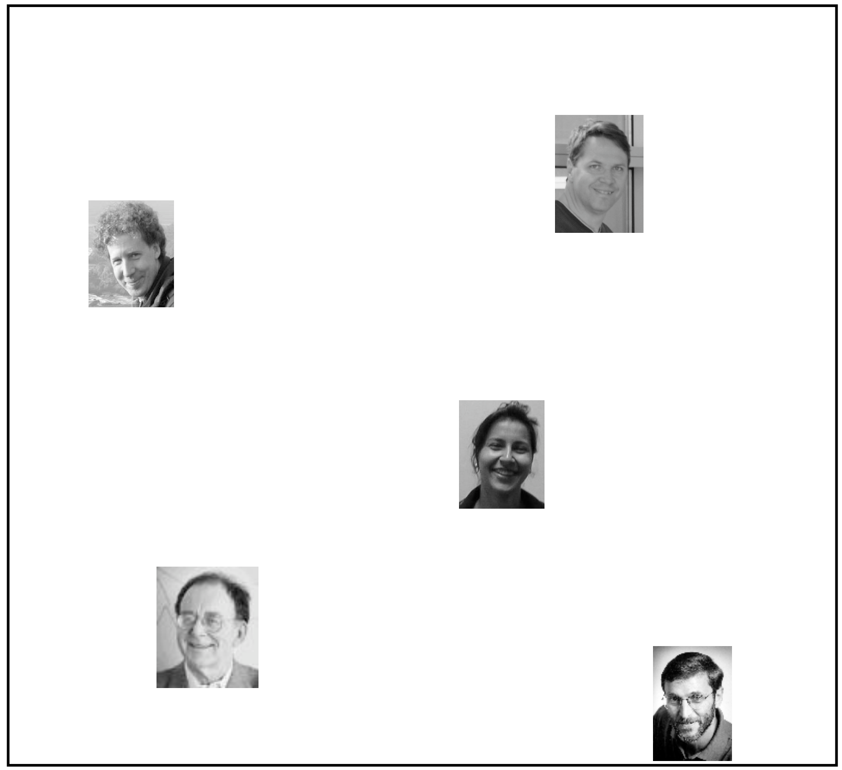
Peer-to-Peer Networks

Content Addressable Network (CAN)

10

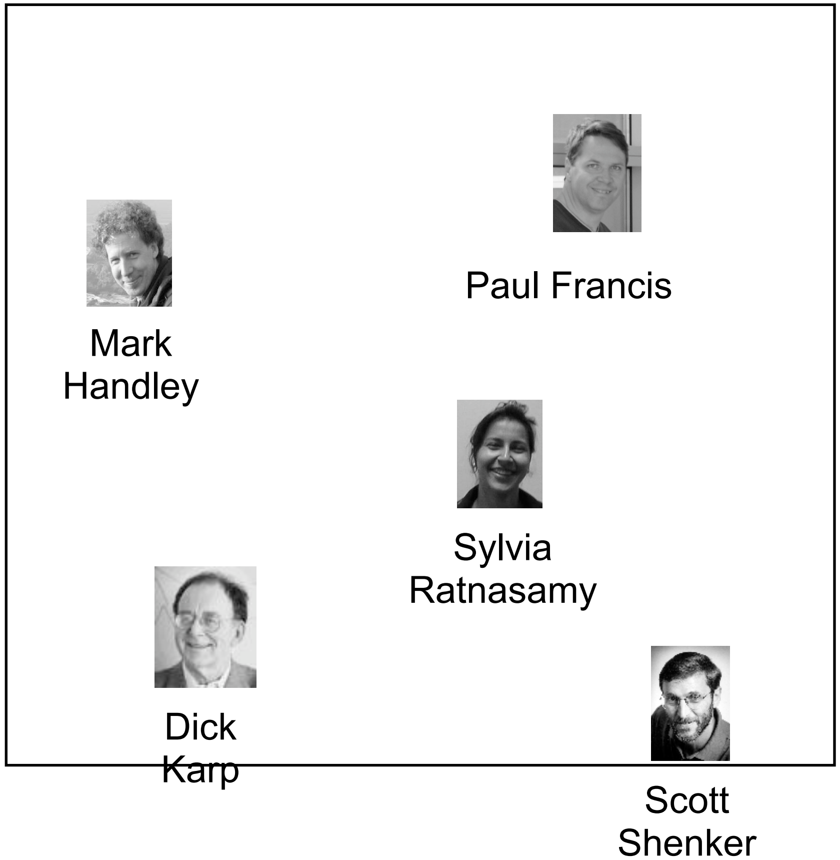
CAN Playground

- **Index entries are mapped to the square $[0,1]^2$**
  - using two hash functions to the real numbers
  - according to the search key
- **Assumption:**
  - hash functions behave a like a random mapping



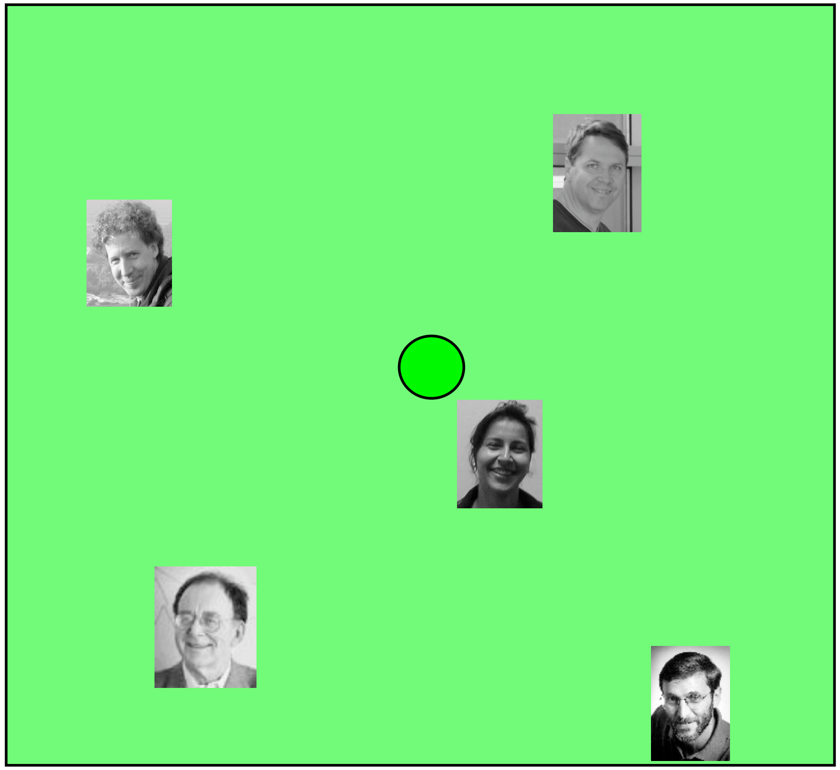
CAN Index Entries

- **Index entries are mapped to the square $[0,1]^2$**
  - using two hash functions to the real numbers
  - according to the search key
- **Assumption:**
  - hash functions behave a like a random mapping
- **Literature**
  - Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. In: Computer Communication Review. Volume 31., Dept. of Elec. Eng. and Comp. Sci., University of California, Berkeley (2001) 161–172



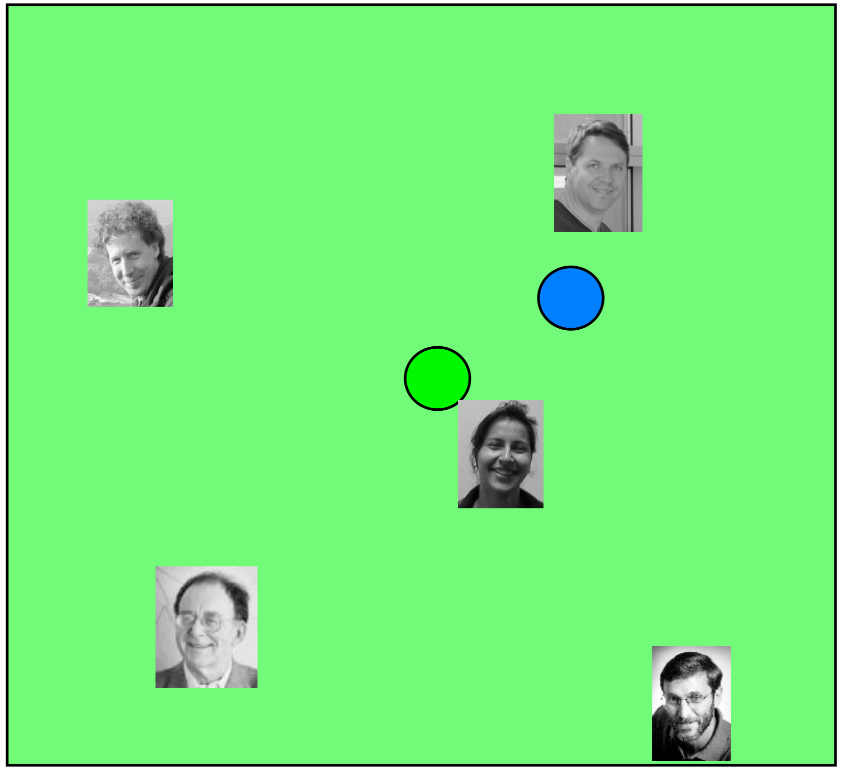
First Peer in CAN

- **In the beginning there is one peer owning the whole square**
- **All data is assigned to the (green) peer**



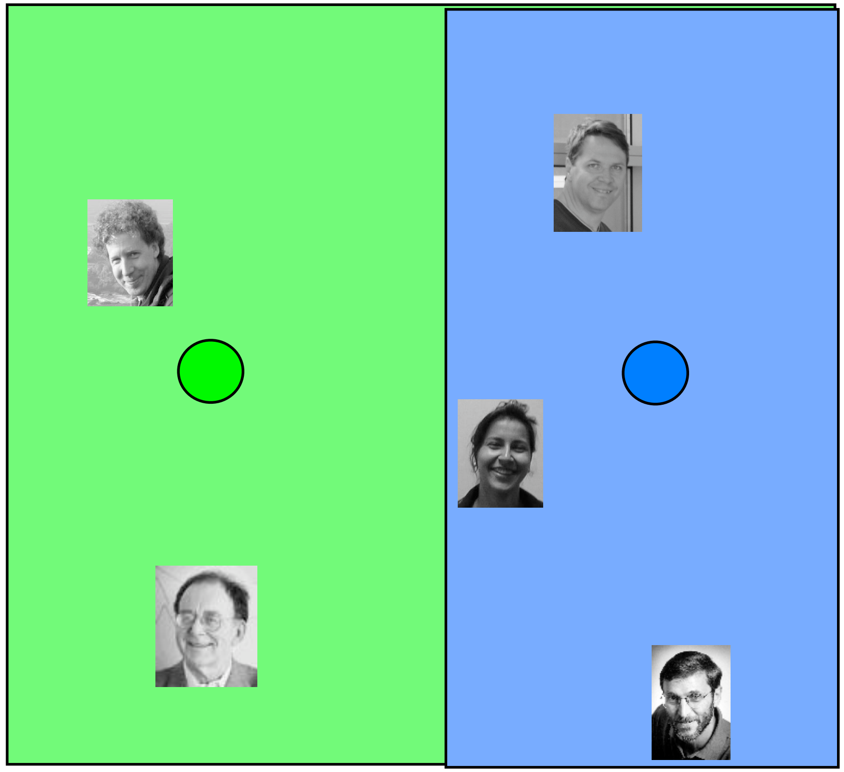
CAN: The 2nd Peer Arrives

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner



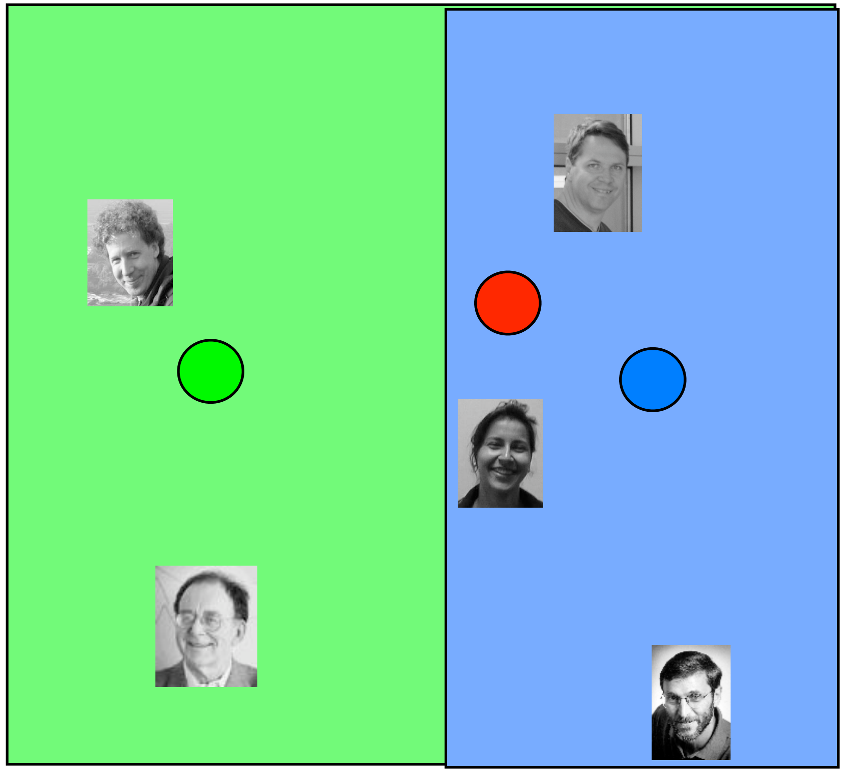
CAN: 2nd Peer Has Settled Down

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



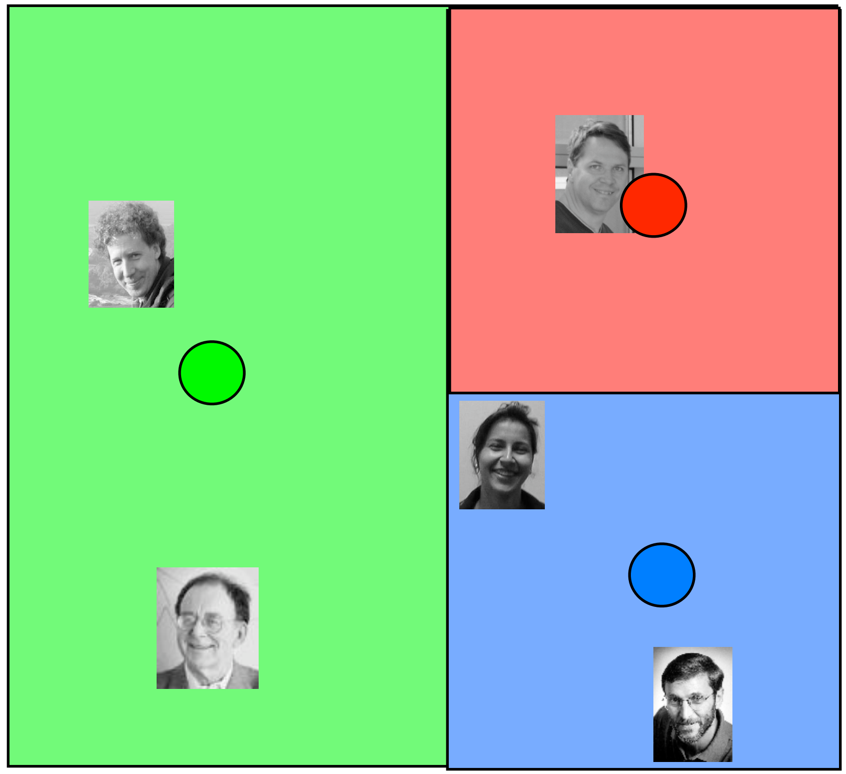
3rd Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



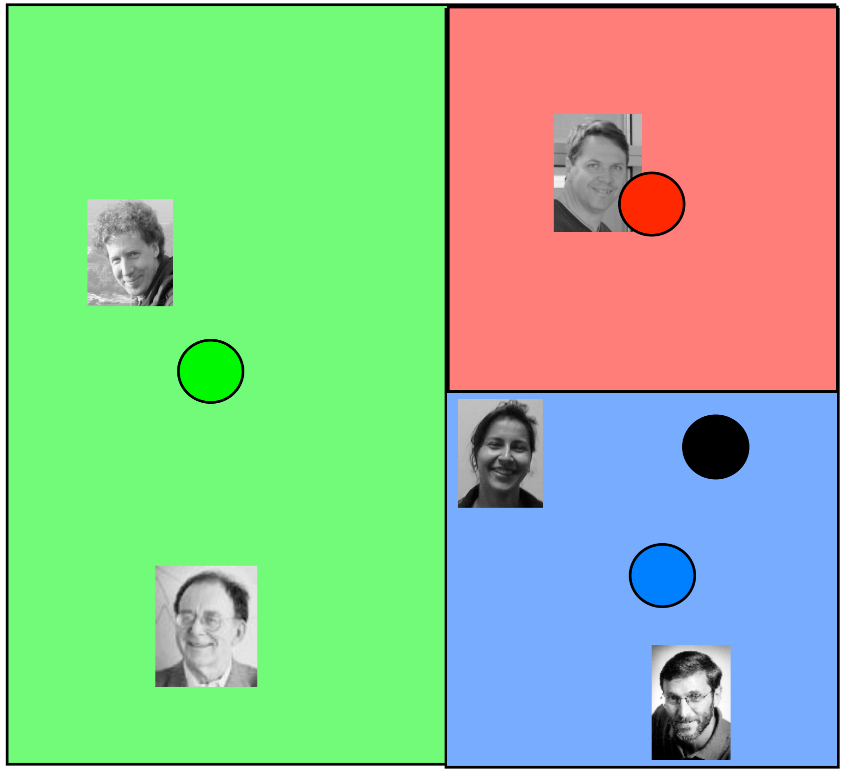
CAN: 3rd Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



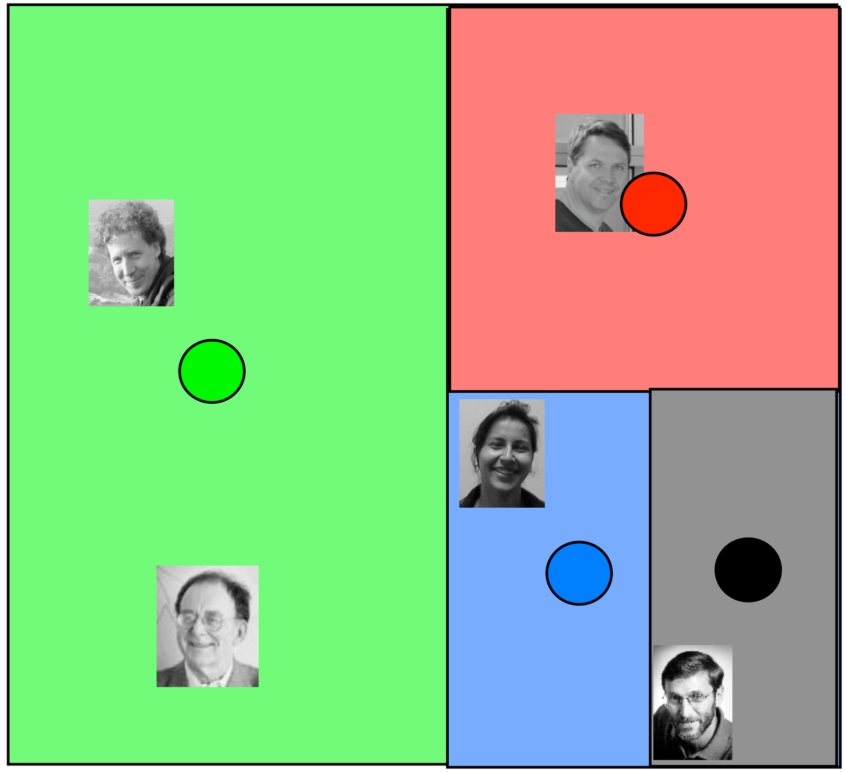
CAN: 4th Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



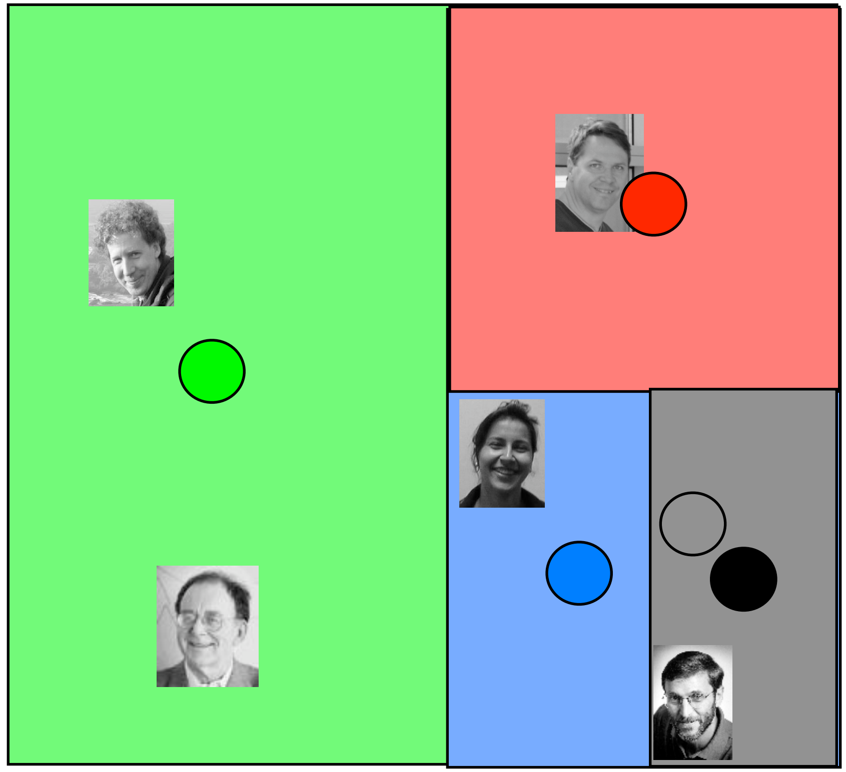
CAN: 4th Peer Added

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



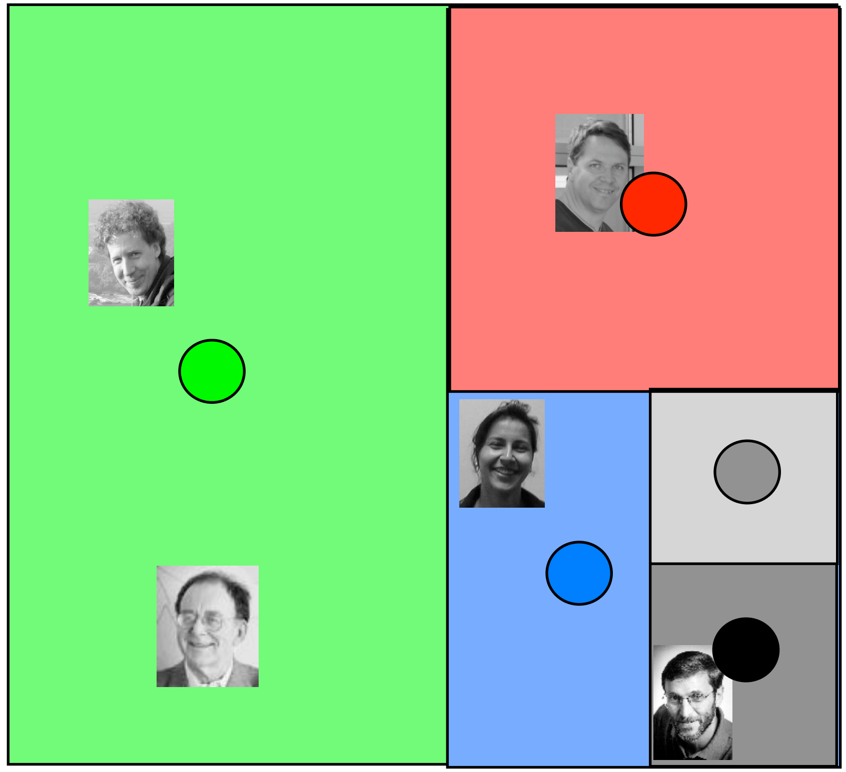
CAN: 5th Peer

- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



CAN: All Peers Added

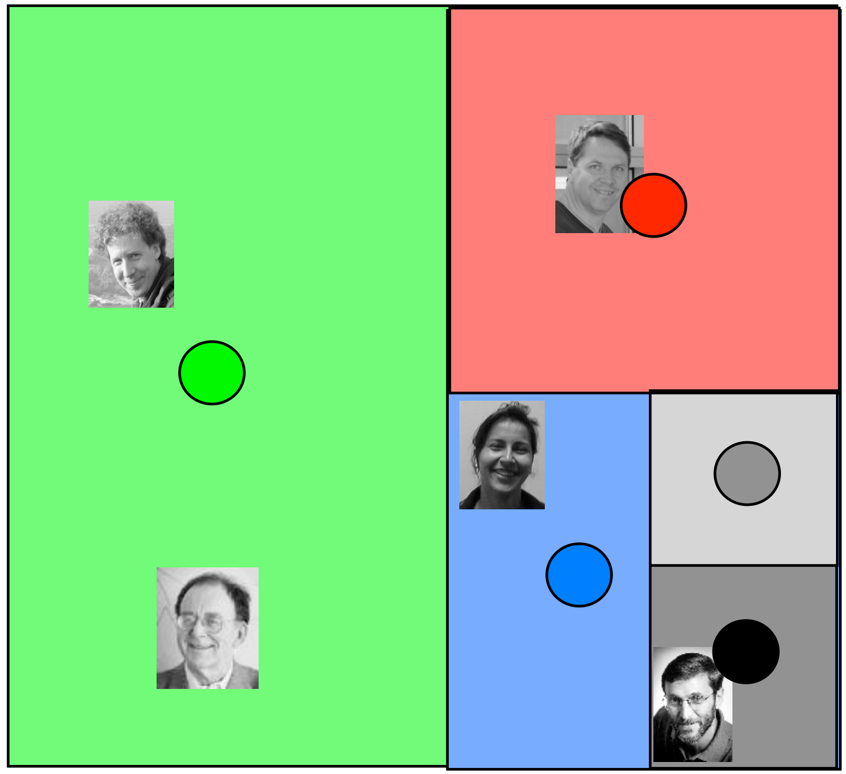
- **The new peer chooses a random point in the square**
  - or uses a hash function applied to the peers Internet address
- **The peer looks up the owner of the point**
  - and contacts the owner
- **The original owner divides his rectangle in the middle and shares the data with the new peer**



On the Size of a Peer's Area

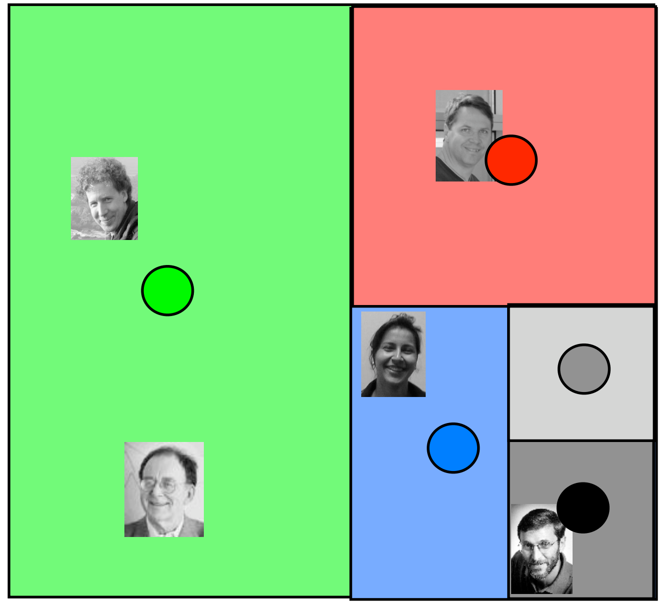
- **R(p): rectangle of peer p**
- **A(p): area of the R(p)**
- **n: number of peers**
- **area of playground square: 1**
- **Lemma**
  - For all peers we have $E[A(p)] = \frac{1}{n}$
- **Lemma**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have

$$P_{R,n} \leq e^{-n\mathrm{Vol}(R)}$$



Expected Area of a Peer

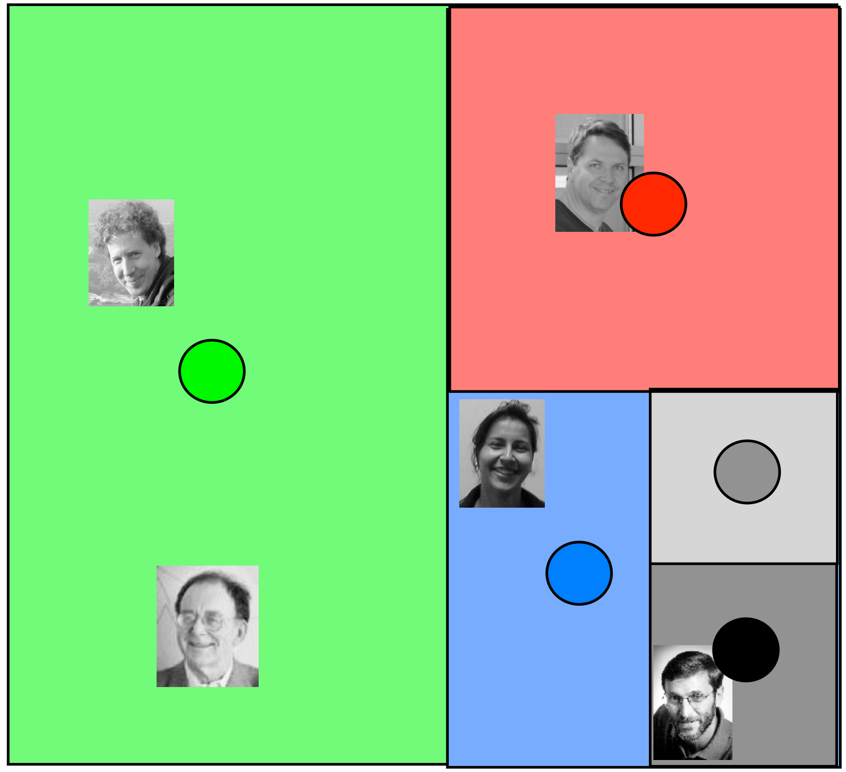
- **Lemma**
  - For all peers we have $E[A(p)] = \frac{1}{n}$
- **Proof**
  - Let {1,..,n} be the peers
  - inserted in a random order
  - Then
$$\sum_{i=1}^{n} A(p) = 1$$
  - Because of symmetry
$$\forall i \in \{1,\ldots,n\} \ : \ A(i) = A(1)$$
  - Therefore
$$1 = \sum_{i=1}^{n} A(i) = E\left[\sum_{i=1}^{n} A(i)\right] = \sum_{i=1}^{n} E[A(i)] = nE[A(1)]$$



On the Slize of a Peer's Area

- **R(p): rectangle of peer p**
- **A(p): area of the R(p)**
- **n: number of peers**
- **area of playground square: 1**
- **Lemma**
  - For all peers we have $E[A(p)] = \frac{1}{n}$
- **Lemma**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have

$$P_{R,n} \leq e^{-n\mathrm{Vol}(R)}$$



Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

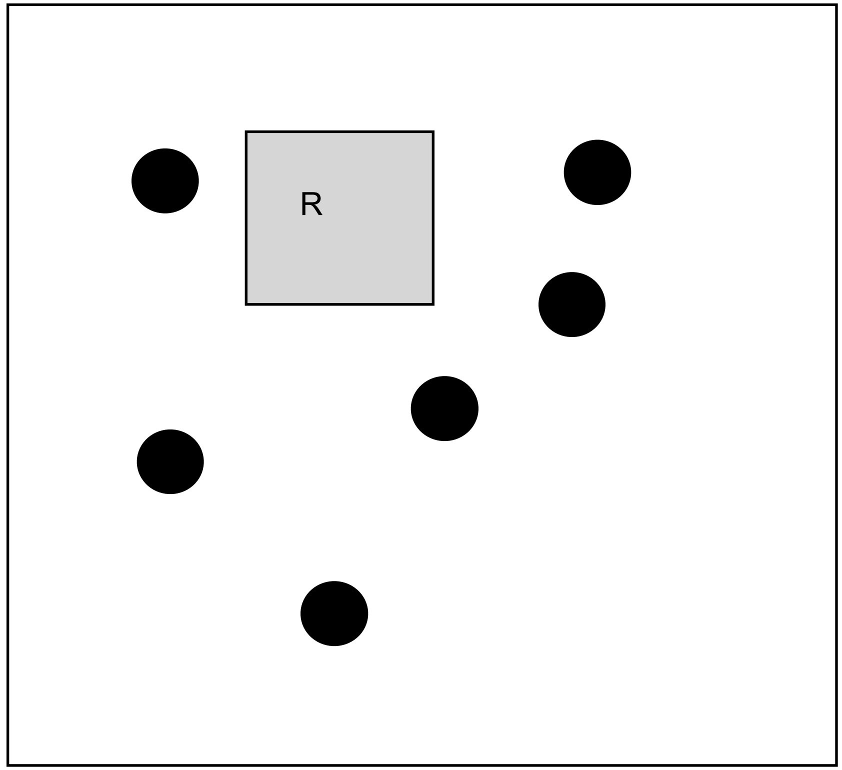
An Area Not being Hit

- **Lemma**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have $P_{R,n} \leq e^{-n \mathrm{Vol}(R)}$
- **Proof**
  - Let x=Vol(R)
  - The probability that a peer does not fall into R is $1 - x$
  - The probability that n peers do not fall into R is $(1-x)^n$
  - So, the probability is bounded by

$$(1-x)^n = ((1-x)^{\frac{1}{x}})^{nx} \leq e^{-nx}$$

  - because

$$m > 1 \ : \ \left(1 - \frac{1}{m}\right)^m \leq \frac{1}{e}$$

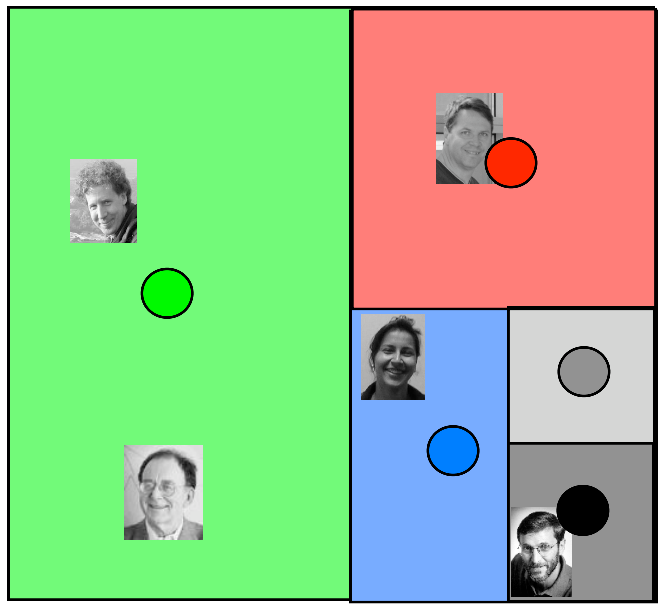


How Fair Are the Data Balanced

- **Lemma**
  - With probability n^{-c} a rectangle of size (c ln n)/n is not further divided
- **Proof**
  - Let $P_{R,n}$ denote the probability that no peers falls into an area R. Then we have $P_{R,n} \leq e^{-n \mathrm{Vol}(R)}$

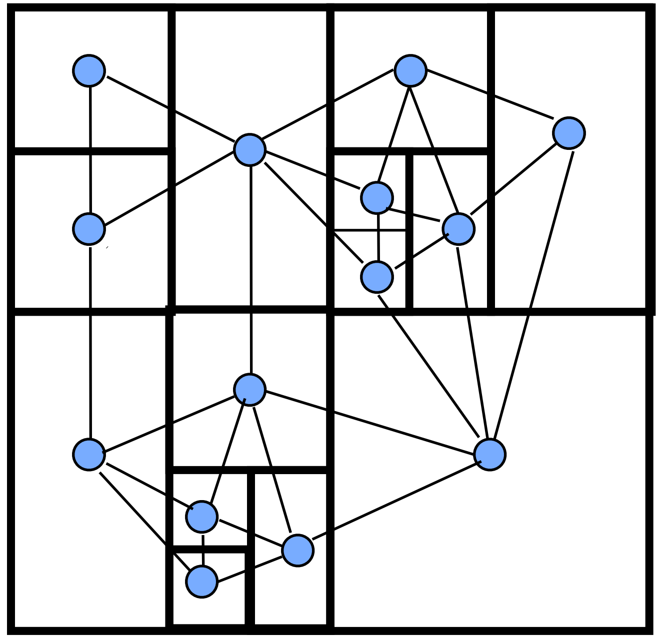
$$P_{R,n} \leq e^{-n \frac{c \ln n}{n}} = e^{-c \ln n} = n^{-c}$$

- **Every peer receives at most c (ln n) m/n elements**
  - if all m elements are stored equally distributed over the area
- **While the average peer stores m/n elements**
- **So, the number of data stored on a peer is bounded by c (ln n) times the average amount**



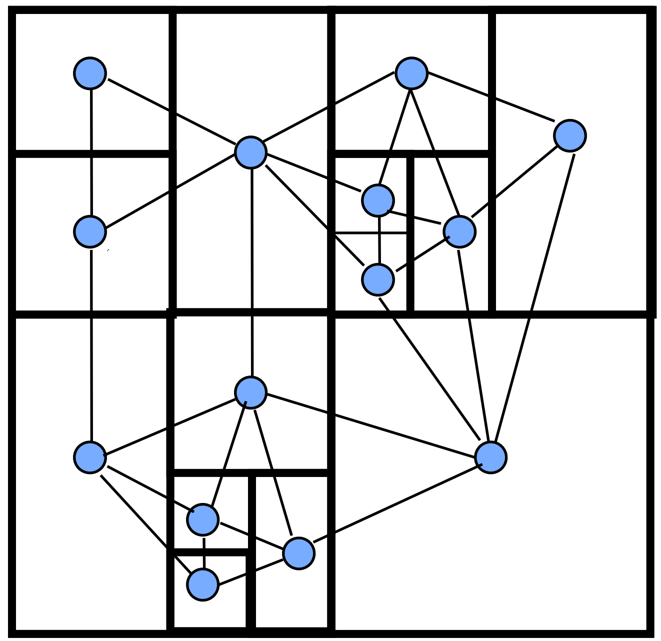
Network Structure of CAN

- **Let d be the dimension of the square, cube, hyper-cube**
  - 1: line
  - 2: square
  - 3: cube
  - 4: ...
- **Peers connect**
  - if the areas of peers share a (d-1)-dimensional area
  - e.g. for d=2 if the rectangles touch by more than a point



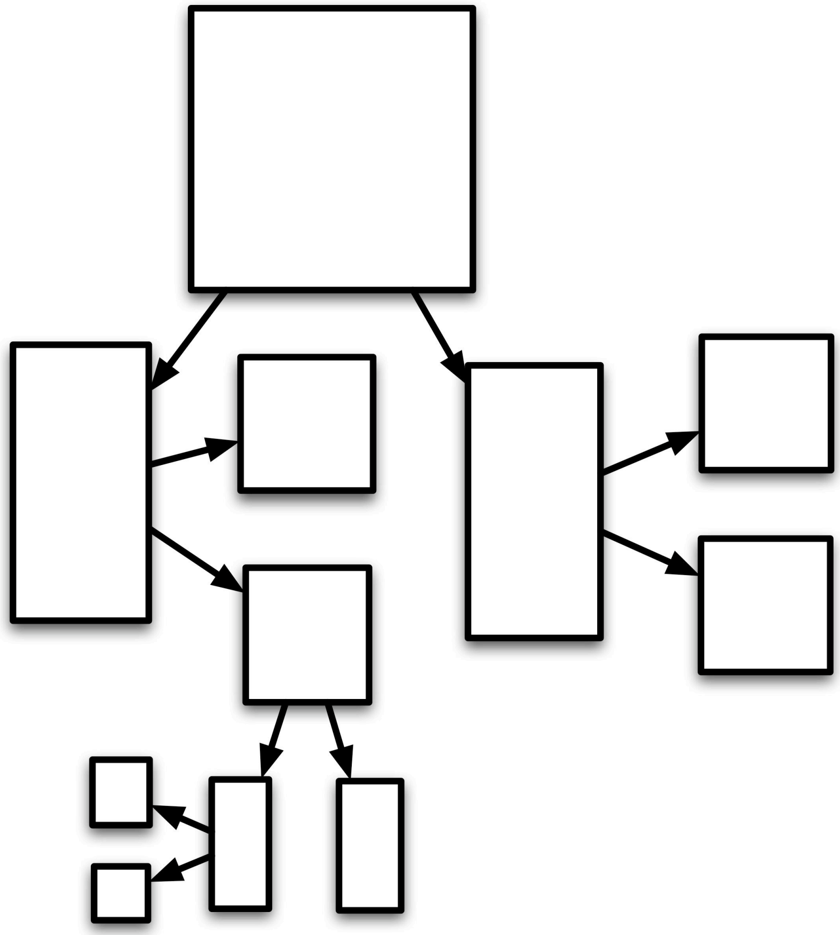
Lookup in CAN

- Compute the position of the index using the hash function on the key value
- Forward lookup to the neighbored peer which is closer to the index
- Expected number of hops for CAN in d dimensions:
  - $O(n^{1/d})$
- Average degree of a node
  - $O(d)$



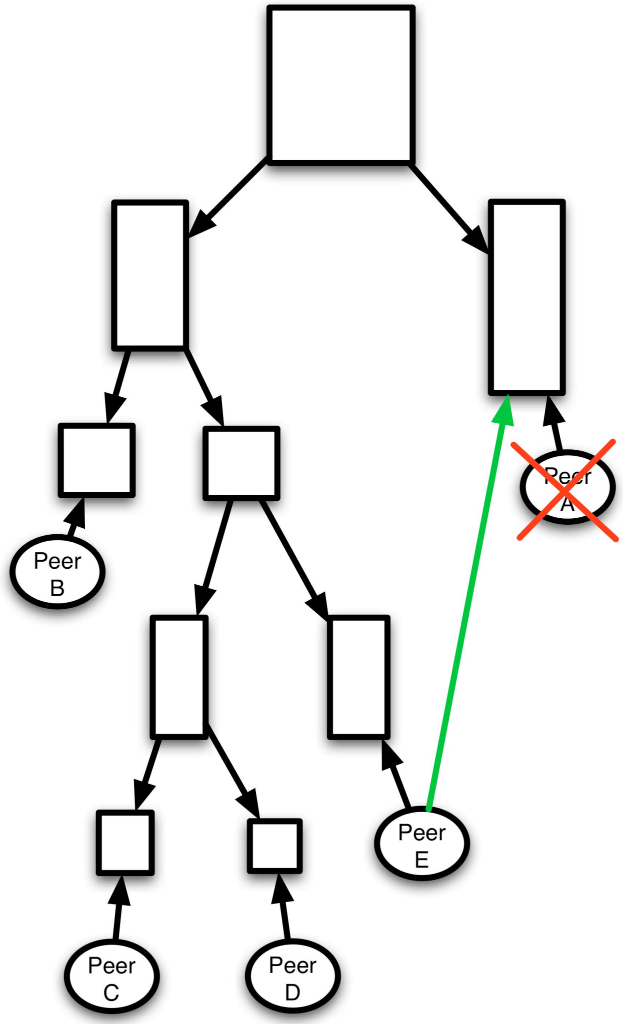
Insertions in CAN = Random Tree

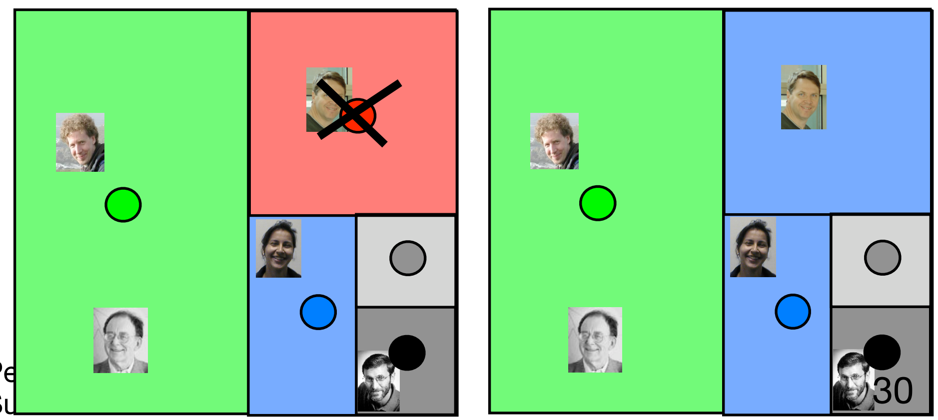
- **Random Tree**
  - new leaves are inserted randomly
  - if node is internal then flip coin to forward to left or right sub-tree
  - if node is leaf then insert two leafs to this node
- **Depth of Tree**
  - in the expectation: O(log n)
  - Depth O(log n) with high probability, i.e. $1-n^{-c}$
- **Observation**
  - CAN inserts new peers like leafs in a random tree



Leaving Peers in CAN

- **If a peer leafs**
  - he does not announce it
- **Neighbors continue testing on the neighborhood**
  - to find out whether peer has left
  - the first neighboir who finds a missing neighbor takes over the area of the missing peer
- **Peers can be responsible for many rectangles**
- **Repeated insertions and deletions of peers leed to fragmentation**

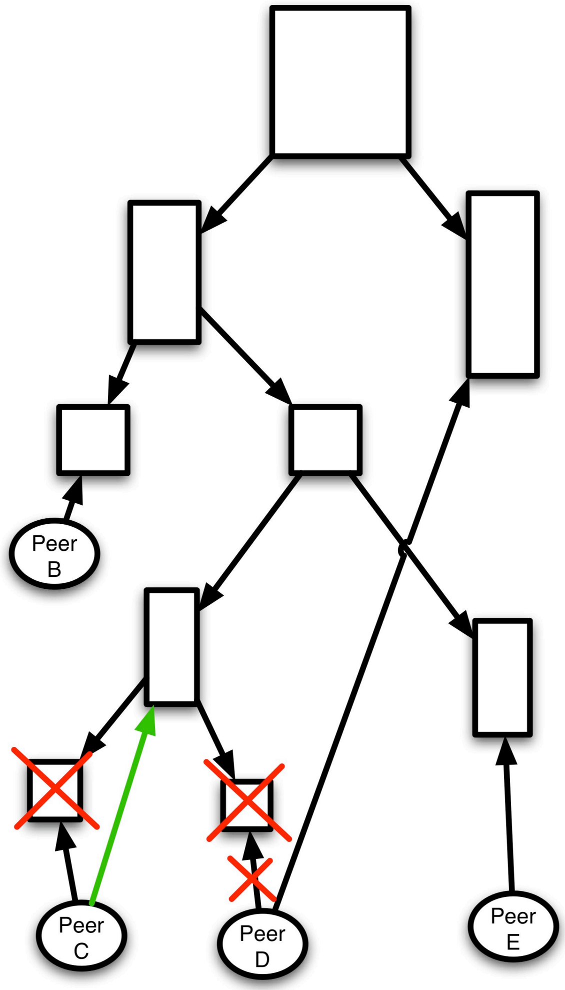




Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

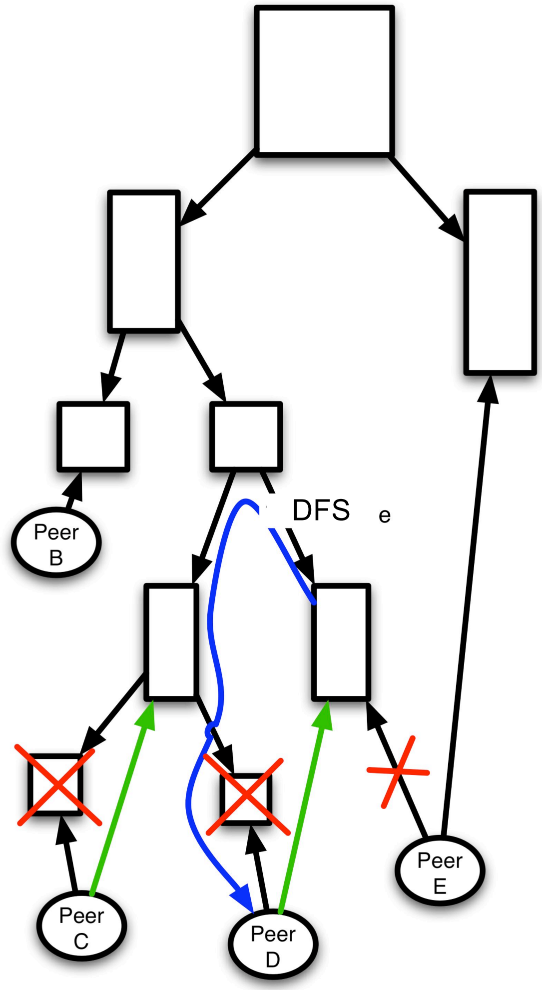
Defragmentation — The Simple Case

- **To heal fragmented areas**
  - from time to to time areas are freshly assigned
- **Every peer with at least two zones**
  - erases smalles zone
  - finds replacement peer for this zone
- **1. case: neighboring zone is undivided**
  - both peers are leafs in the random tree
  - transfer zone to the neighbor



Defragmentation — The Difficult Case

- **Every peer with at least two zones**
  - erases smalles zone
  - finds replacement peer for this zone
- **2. case: neighboring zone is further divided**
  - Perform DFS (depth first search) in neighbor tree until two neighbored leafs are found
  - Transfer the zone to one leaf which gives up his zone
  - Choose the other leaf to receive the latter zone

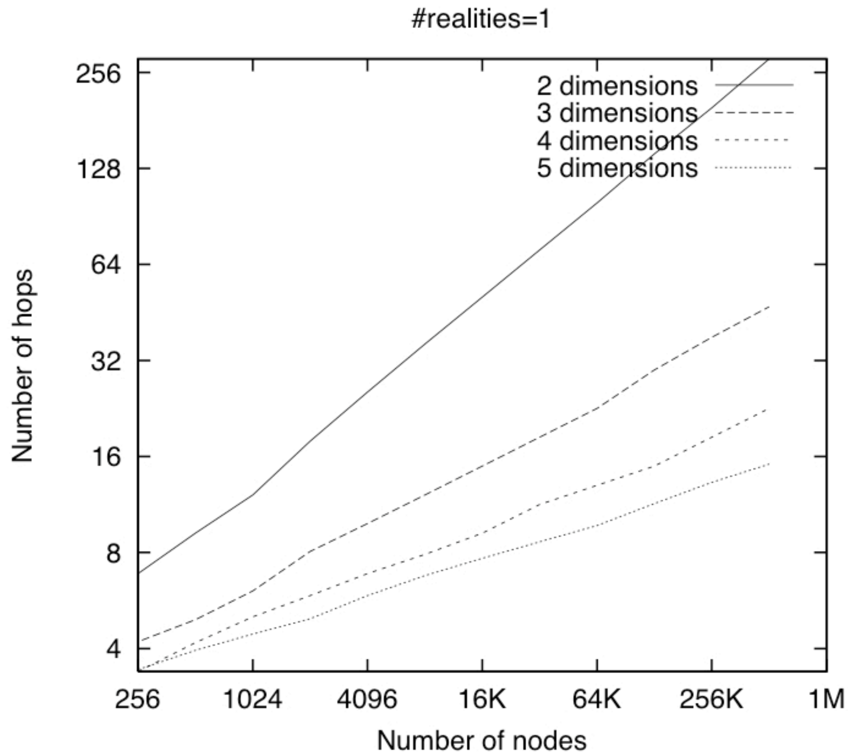


Improvements for CAN

- More dimensions
- Multiples realities
- Distance metric for routing
- Overloading of zones
- Multiple hasing
- Topology adapted network construction
- Fairer partitioning
- Caching, replication and hot-spot management

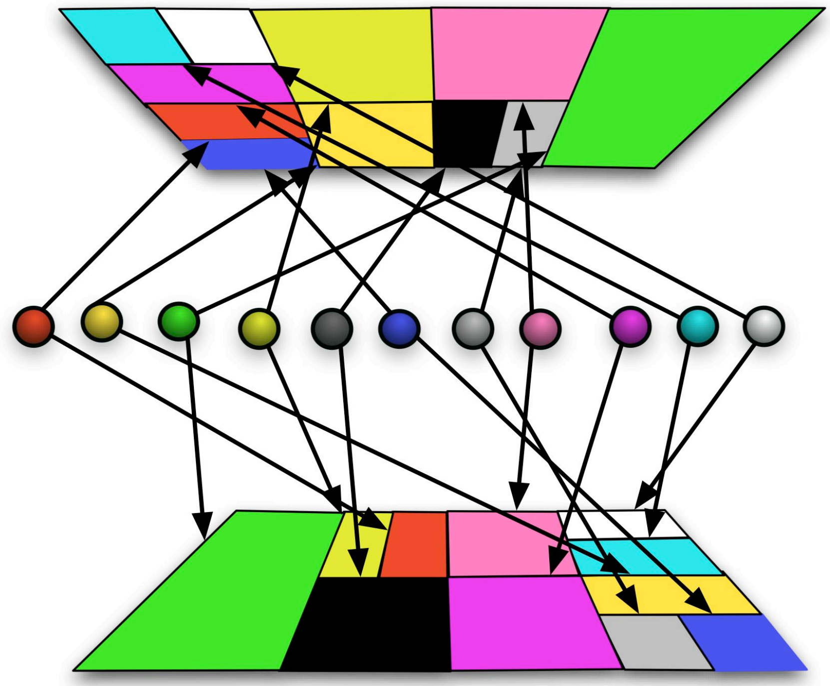
Higher Dimensions

- **Let d be the dimension of the square, cube, hyper-cube**
  - 1: line
  - 2: square
  - 3: cube
  - 4: ...
- **The expected path length is $O(n^{1/d})$**
- **Average number of neighbors O(d)**



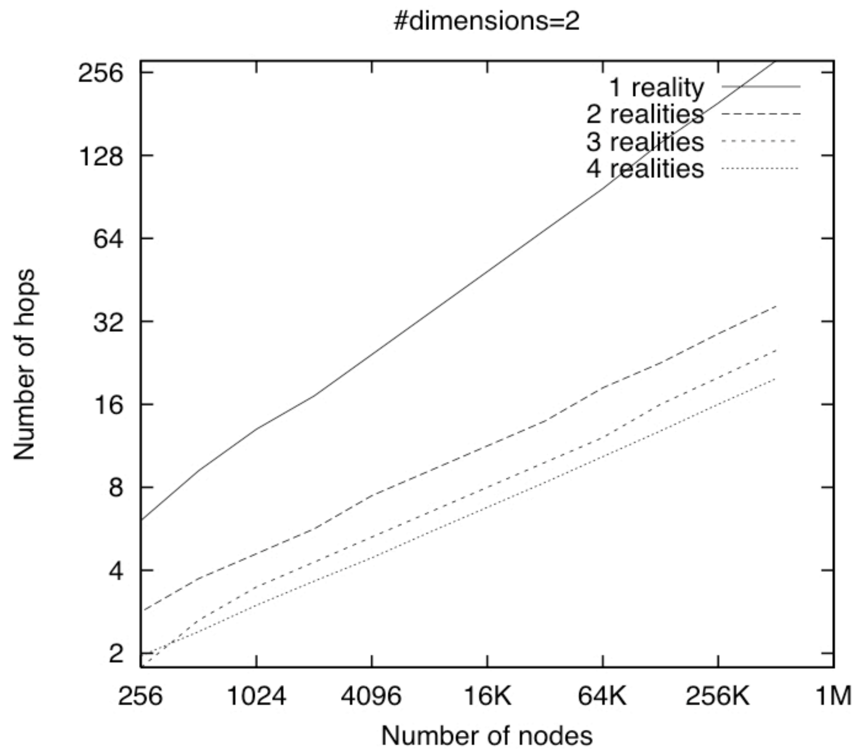
More Realities

- **Build simultanously r CANs with the same peers**
- **Each CAN is called a *reality***
- **For lookup**
  - greedily jump between realities
  - choose reality with the closest distance to the target
- **Advantanges**
  - robuster network
  - faster search



More Realities

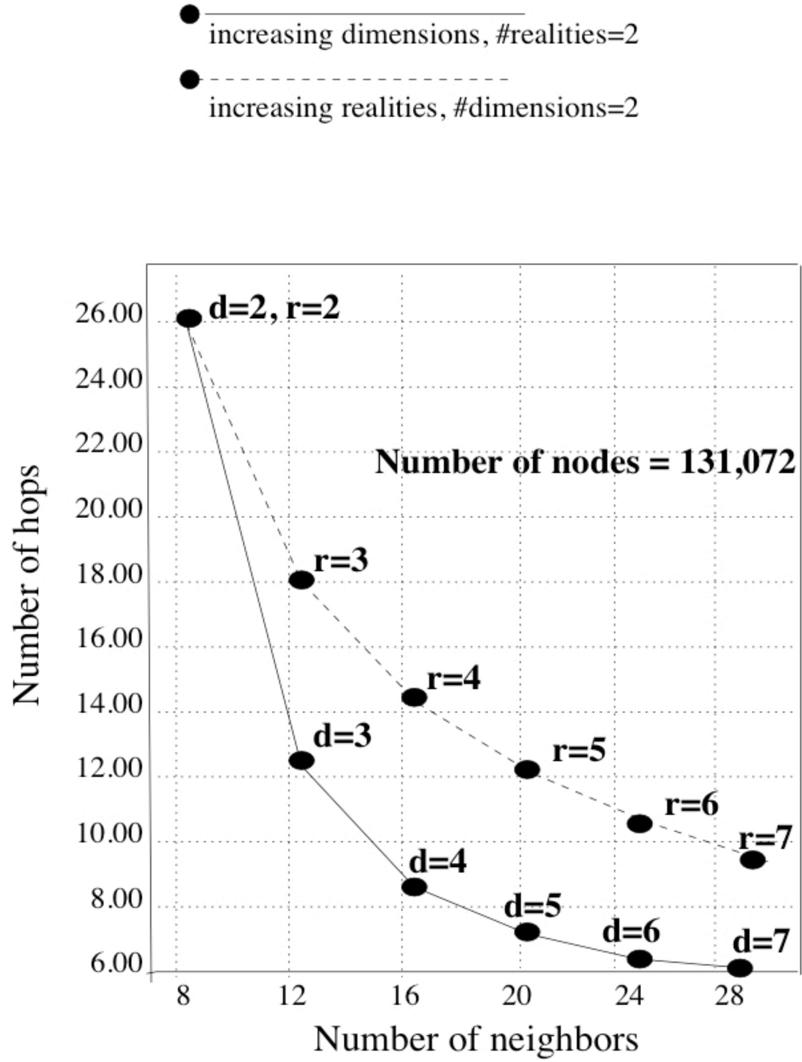
- **Advantages**
  - robuster
  - shorter paths



Computer Networks and Telematics
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

Realities vs. Dimensions

- **Dimensionens reduce the lookup path length more effciently**
- **Realities produce more robust networks**



CoNe
Freiburg

Peer-to-Peer Networks

End of 2nd Week

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer
Summer 2008