P2P Networks – Exercise
Solution For Exercise # 6

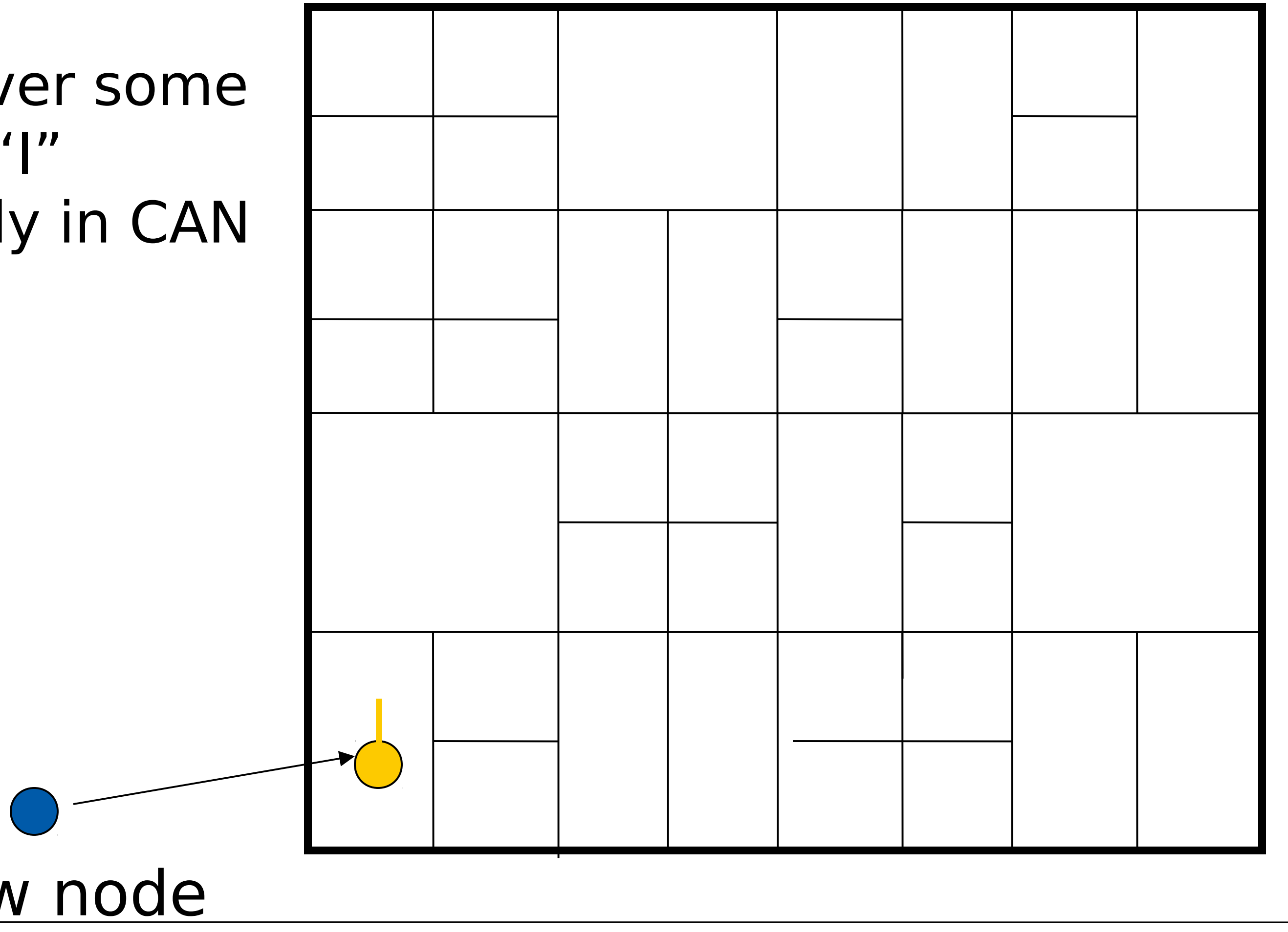
Ikram M. Khan, P2P Networks Group,
TU Darmstadt

Date: Dec. 13th, 2011

CAN

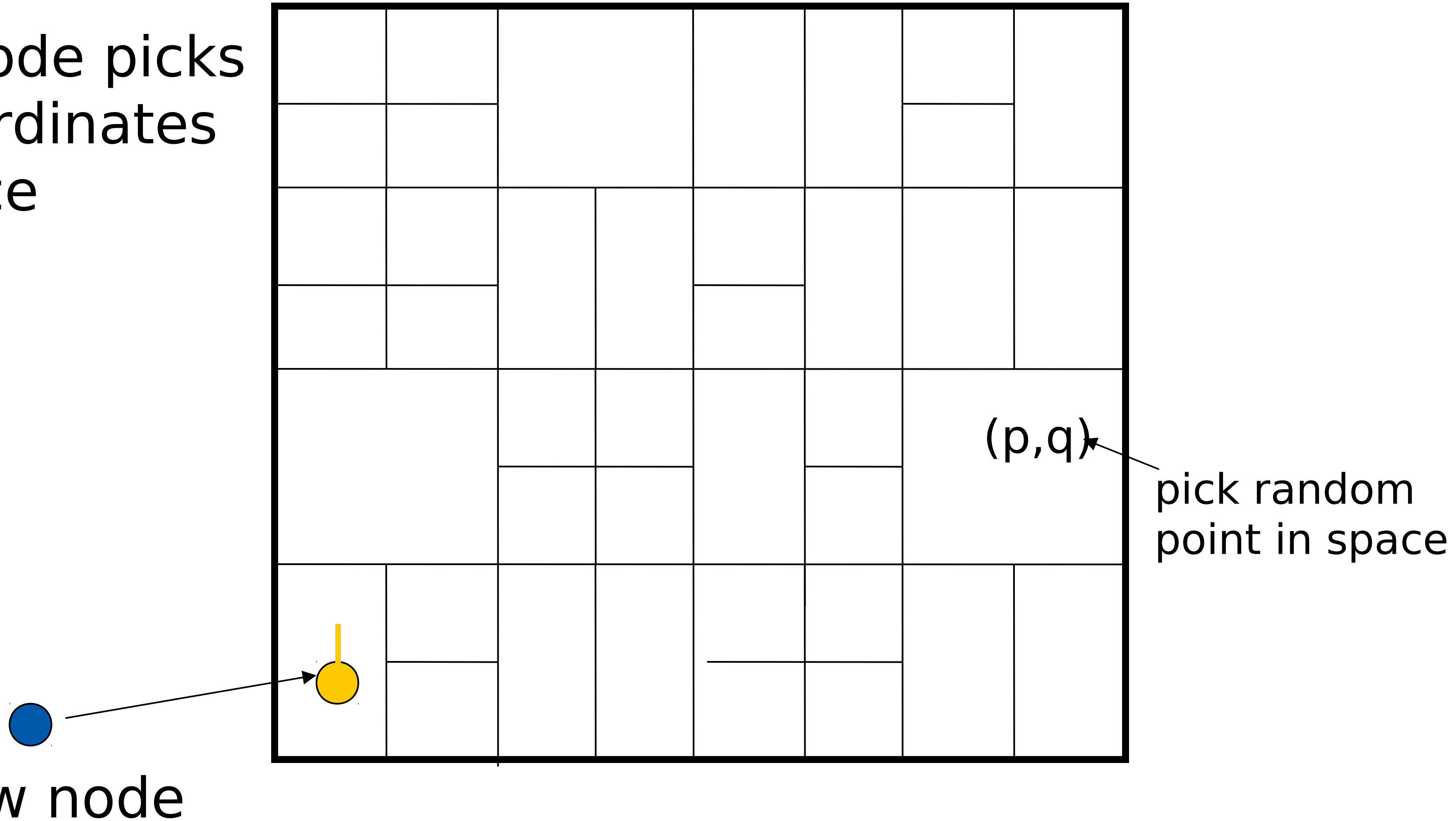
1.1 CAN: Join Procedure

Discover some node "I" already in CAN



1.1 CAN: Join Procedure

New node picks its coordinates in space

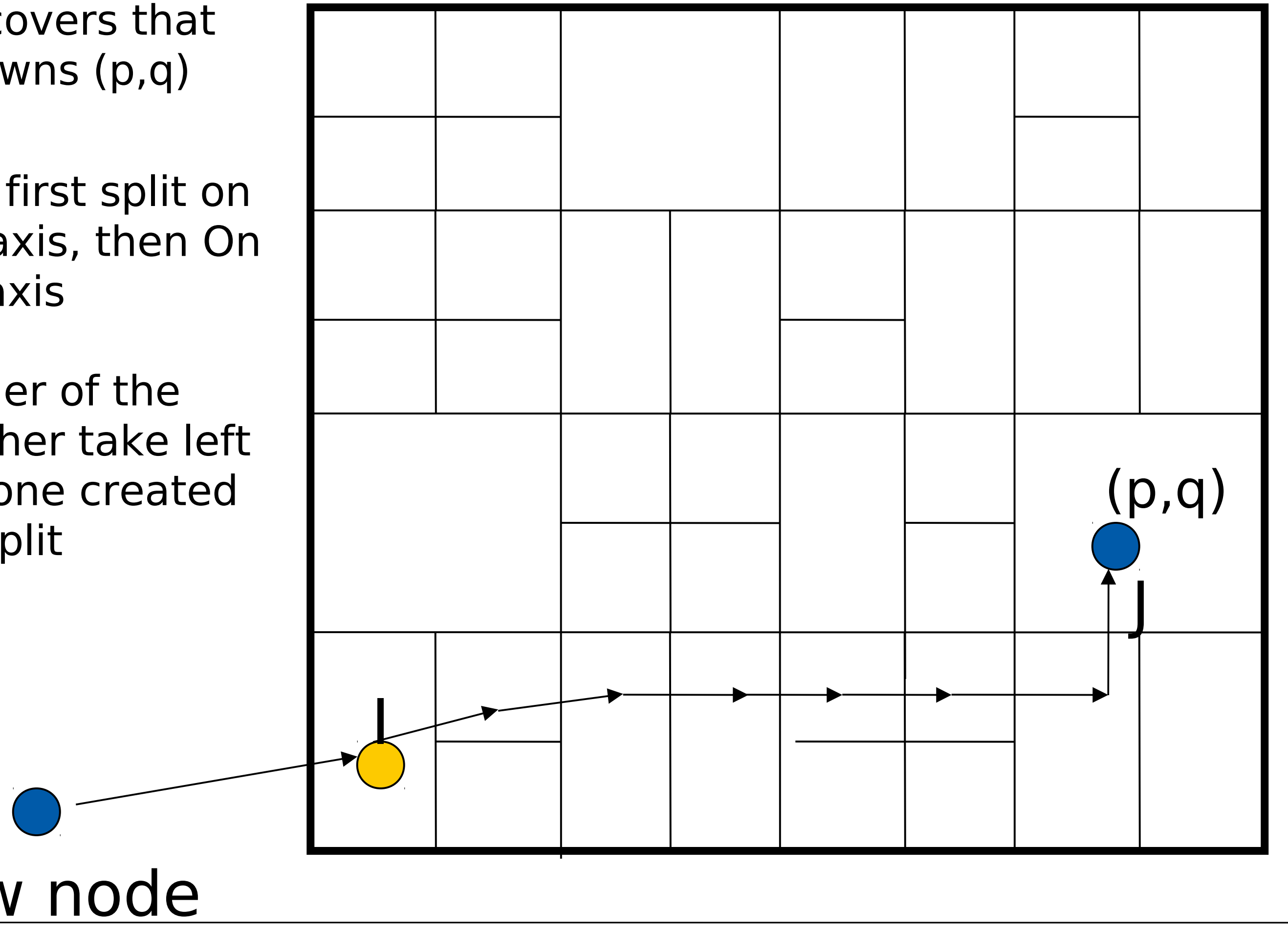


1.1 CAN: Join Procedure

I routes to (p,q), and discovers that node J owns (p,q)

For 2D, first split on the X---axis, then On the Y---axis

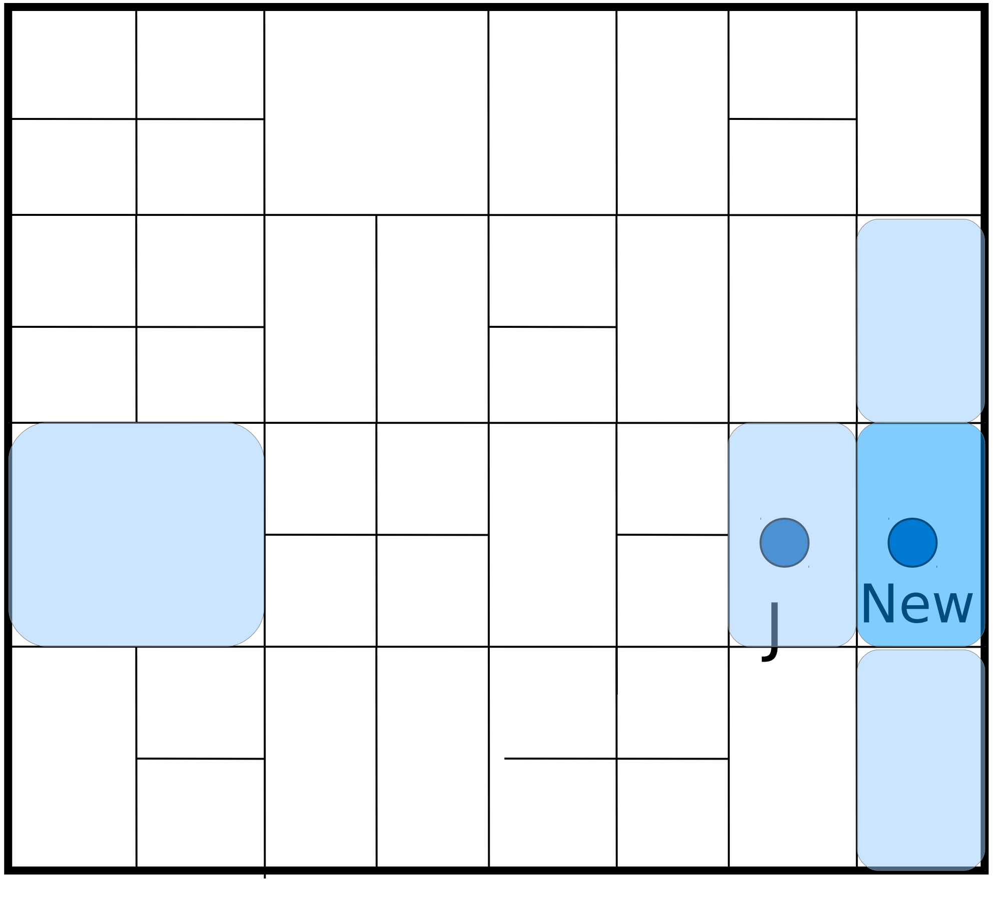
Controller of the zone either take left or top zone created by the split



1.1 CAN: Join Procedure

Split J’s zone in half. New owns one half

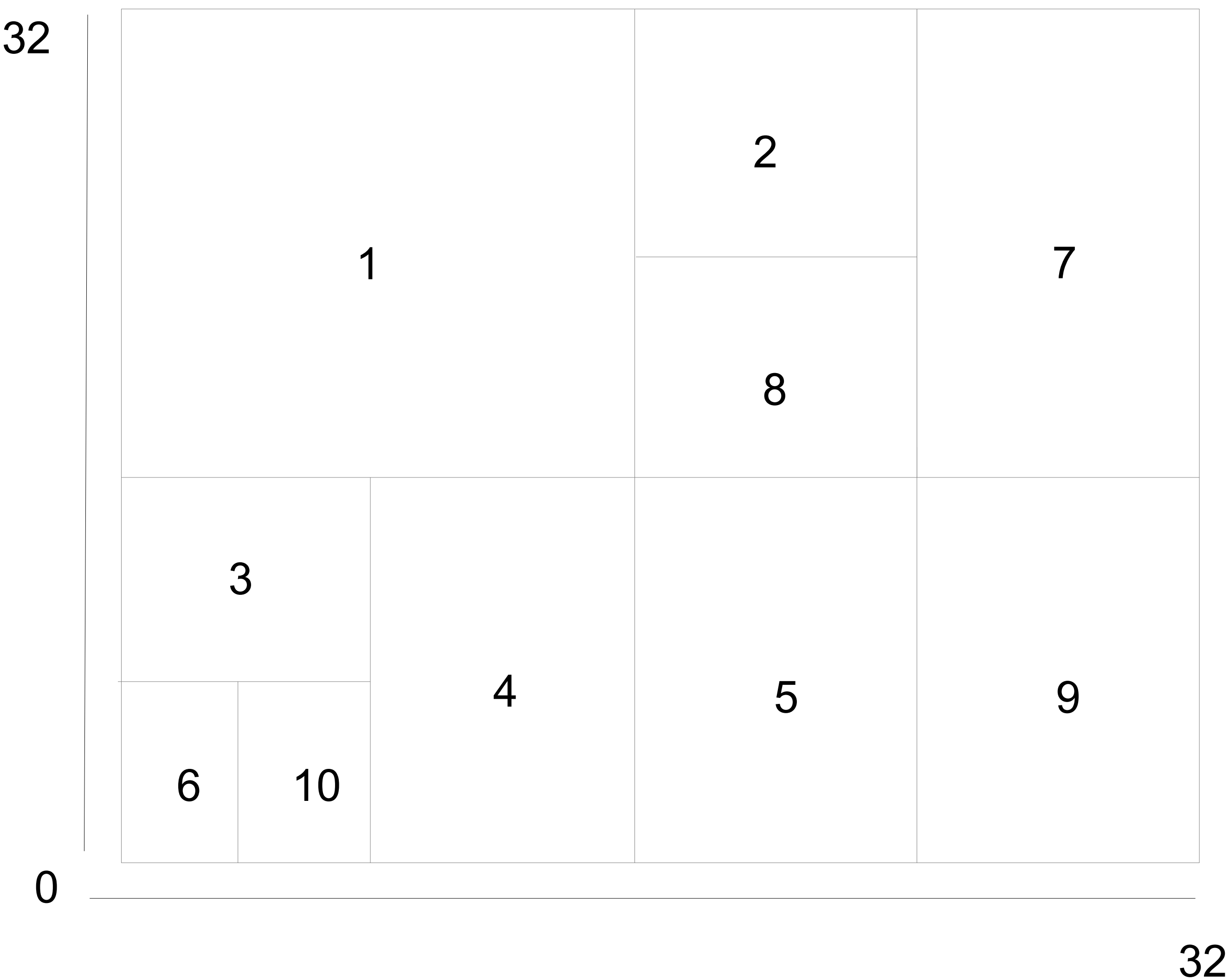
NEW node extablishes connection with its neighborhod



1.2 Partitioning of CAN’s identifier space

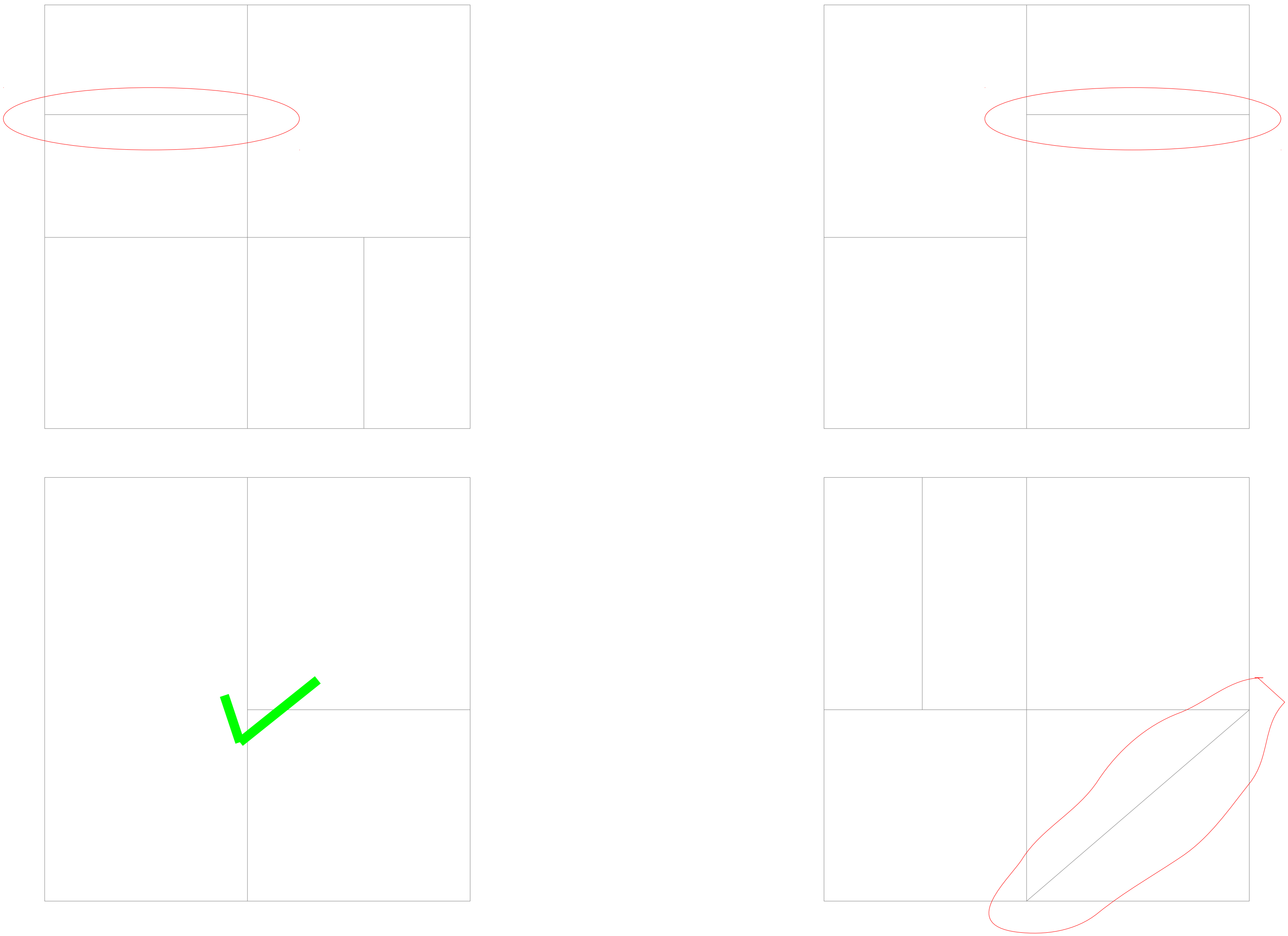
Node	ID
1	17/3
2	20/10
3	15/31
4	7/2
5	27/5
6	2/3
7	16/18
8	21/19
9	25/11
10	7/2

1.2 Partitioning of CAN’s identifier space (6 P.)



Node	Neighboring nodes
1	2,3,4,6,7,8,10
4	1,3, 5, 10
7	1, 2, 8, 9
10	1, 3, 4, 6

1.3 Incorrect Paritioning of Identifier Space (2 P.)



Kademlia

2.1 XOR Metric

- Nodes receive requests from the same distribution of nodes contained in their routing tables
  - Over-hearing each query they receive

- XOR's undirectionality ensure lookup convergence
  - Enable caching of <key, value> pairs along a lookup path

- Requests can be sent to any node in an interval
  - Hope selection based on latency
  - Query sending parallelly and asynchronously

2.2 Routing State Information

- Kademlia routing tables consist of 160 k-buckets
  - Contain nodes at distance $2^i \leq d \leq 2^{i+1}$
  - Each bucket contains at most k entries
  - For small i, k-buckets most probably empty

- Maximum amount of routing state information (RSI_{max})
  - RSI_{max} = 160 * k * X, X: memeory required for each entry
  - X = |<IP address, UDP port, Node ID>|
  - X = 4 Byte + 2 Byte + 20 Byte = 26 Byte
  - RSI_{max} = 160 * k * 26 => k * 4160 Byte ≈ k * 4 KB

2.3 Update Policy

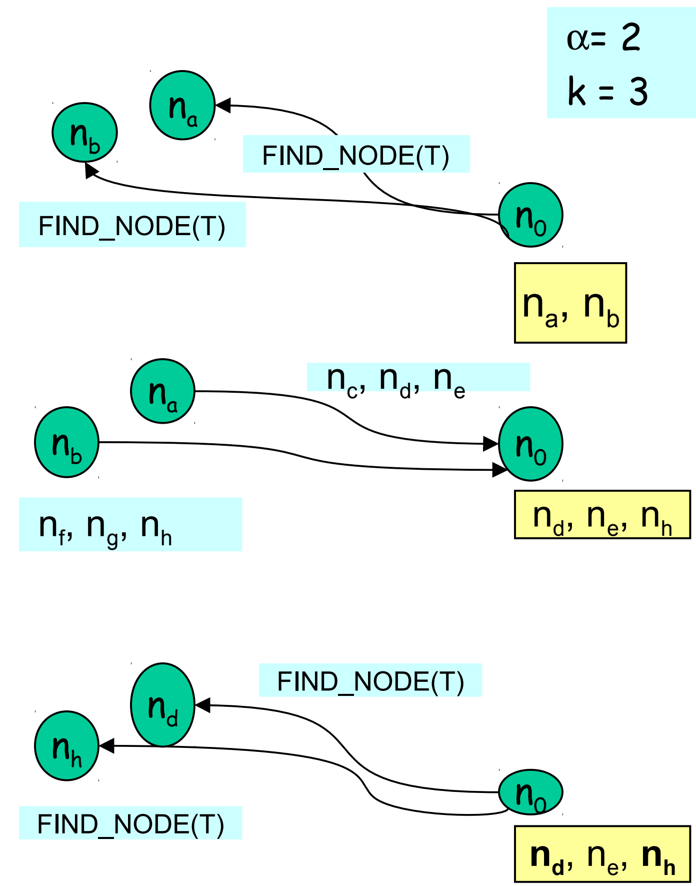
- Least-recently seen eviction policy
  - Except for not removing live nodes

- Ping k-buckets least-recently seen entry
  - If Pinged node responds
    - Discard the new node's information
  - Else
    - Remove the unresponsive node from the head of the list
    - Insert the new node at the tail of the list

- Preferences old nodes
  - K-bucket maximizes the probability that the nodes they contain will remain online

2.4 Lookup Algorithm

- **Goal**: Find k nodes closest to ID T
- **Initial Phase**:
  - Select α nodes closest to T from n_o's routing table
  - Send FIND_NODE(T) to each of the α nodes in parallel
- **Iteration**:
  - Select α nodes closest to T from the results of previous RPC
  - Send FIND_NODE(T) to each of the α nodes in parallel
  - Terminate when a round of FIND_NODE(T) fails to return any closer nodes
- **Final Phase:**
  - Send FIND_NODE(T) to all of k closest nodes not already queried
  - Return when have results from all the k-closest nodes.



2.5 Kademlia and BitTorrent

- Used by trackerless torrents
  - hop://www.bioorrent.org/beps/bep_0005.html
- Used by Azureus in 2005
- Adopted by BitTorrent client
  - µTorrent, BitComet, BitSpirit, and Transmission

Programming Exercise

- Implement the Kademlia routing table according to the Kademlia Paper. Ping requests are messages that only transfer the value 0, 1 byte wide. Ping answers are also 1 byte wide but transfer the value 1.

- Prepare a test setup for the implementation to present on 20th December. If you are not available on this date please contact us via the submission mail address.

Announcement

Exam: Tuesday 14:30-16:10, 21.02.2012

Exam Hall will announced

Please check course web-page for updates

Next Exercise

- Exercise # 8
- Due date 21.12.2011
- 14:25 – 16.05
- S2|02 - C110