Introduction to Multi-Agent Programming

9. Peer-to-Peer Networks for team coordination

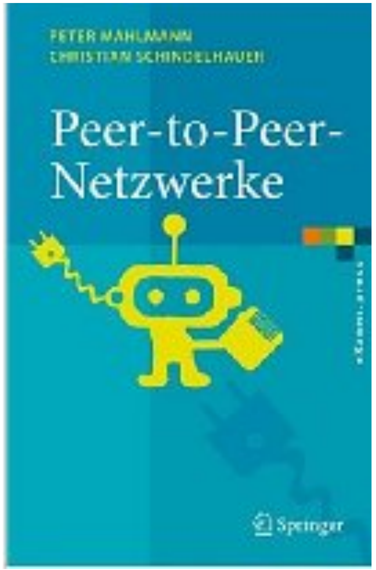
Napster, Gnutella, DHTs, Case Study:
DHT-based team coordination

Alexander Kleiner, Bernhard Nebel

Contents

- Introduction
- P2P systems: Napster, Gnutella, & Co.
- Distributed Hash Tables (DHTs)
- *Case-study*: DHT-based team coordination in logistics
- Summary

Lecture Material



P2P Netzwerke: Algorithmen Und Methoden

By Peter Mahlmann und Christian Schindelhauer

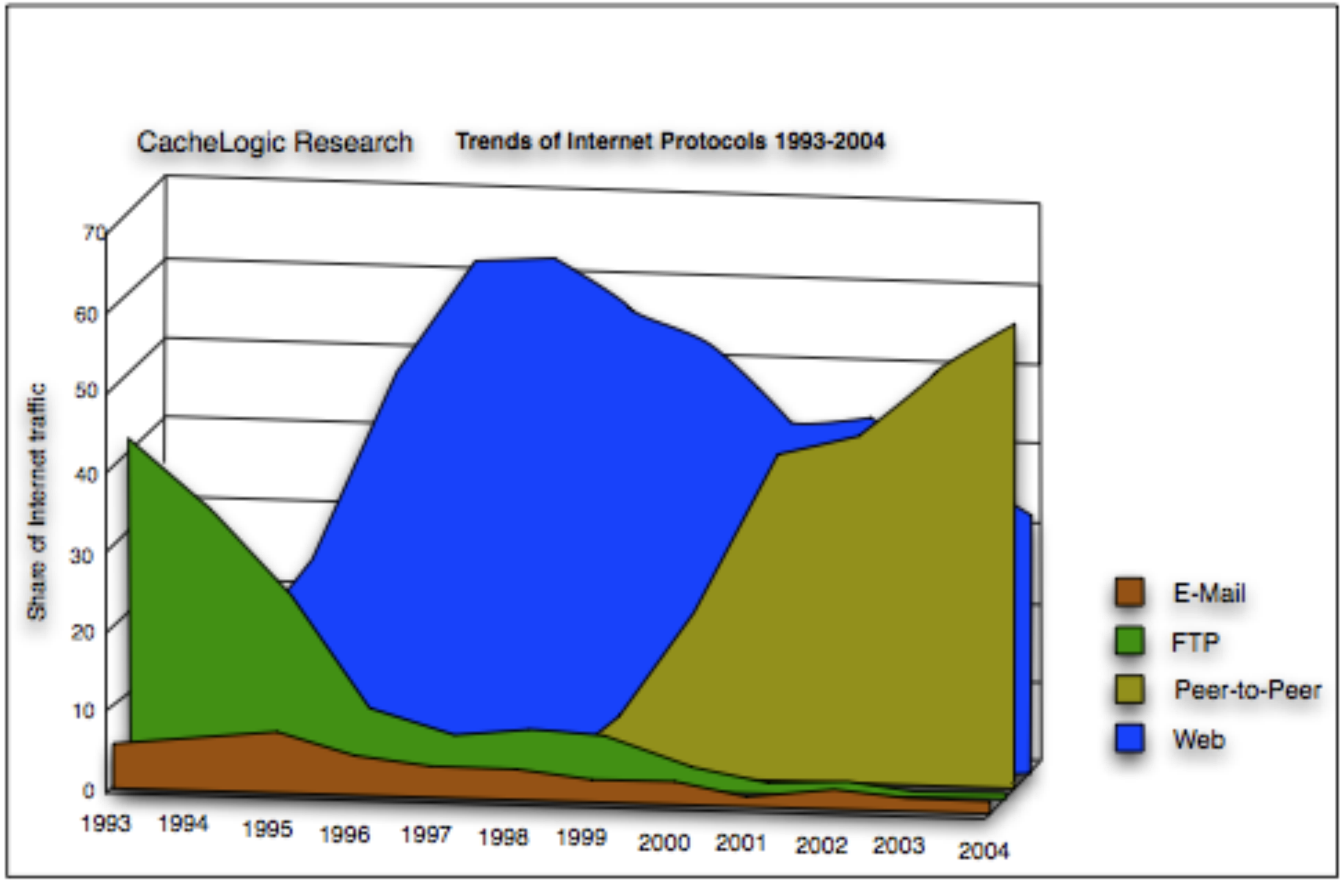
Springer, Berlin (Gebundene Ausgabe - 12. Juli 2007)

Many illustrations have been taken from the book above.

Introduction

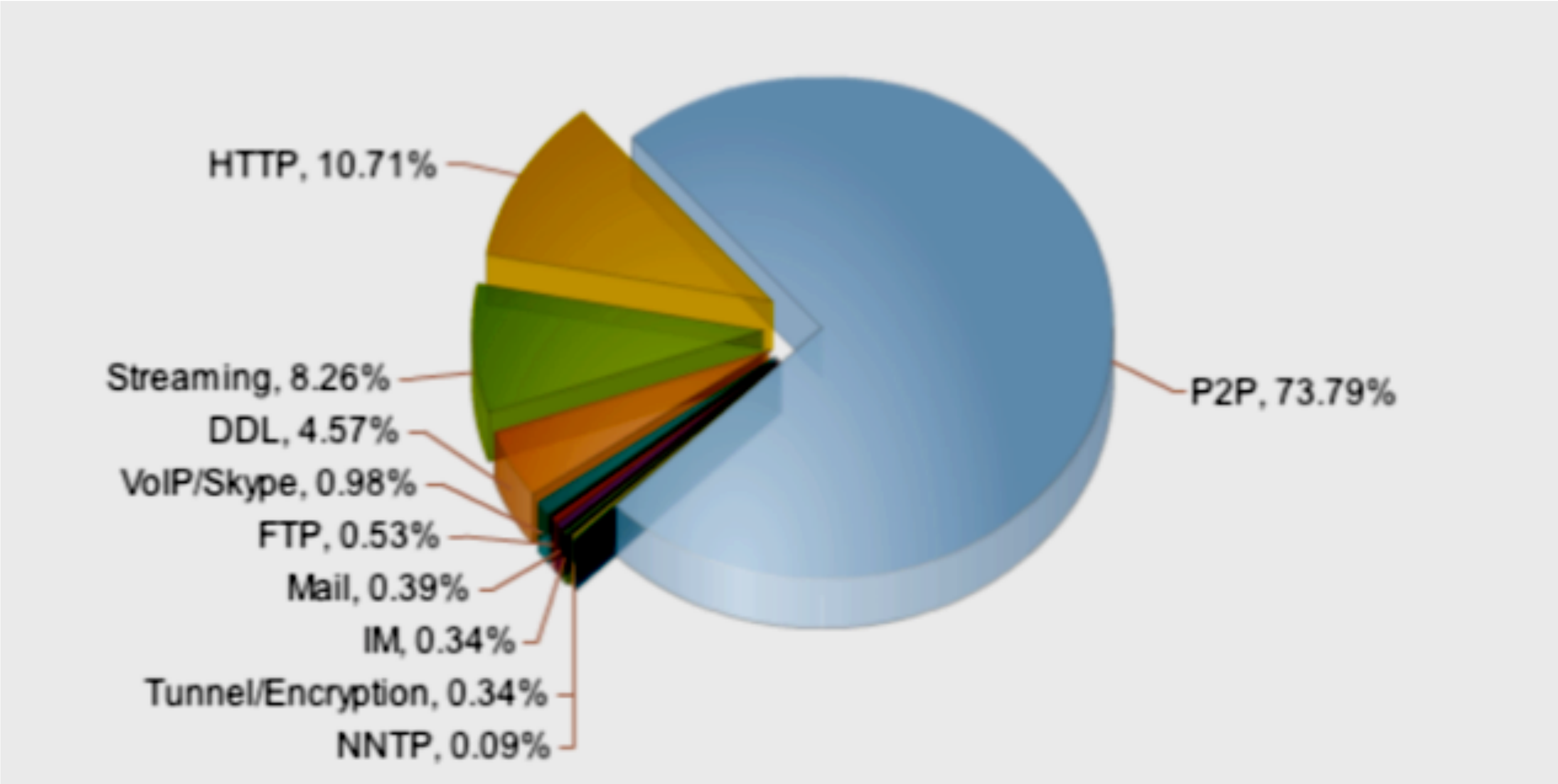
- What are peer-to-peer networks ?
- Mainly known from file sharing systems in the Internet, such as Napster, Gnutella, and BitTorrent
- "Peers" are equally ranked partners, i.e., no one is above the others and no one centrally controls information exchange
- Peer-to-Peer networks are **not** client-server networks!
  - Here a privileged node (server) controls the other nodes (clients)
- Peer-to-Peer networks are overlay networks of the Internet
  - A network protocol in the *application layer* of the OSI model located above the *network layer* (e.g. IP) and *transport layer* (e.g. TCP)
- In contrast to client-server, peer-to-peer scales-up with the number of nodes!

Introduction
Global Internet Traffic Shares 1993-2004



Source: Ipoque 2007 / Lecture Slides C. Schindelhauer

Introduction
P2P share Germany 2007



Source: Ipoque 2007 / Lecture Slides C. Schindelhauer

Development of P2P Networks

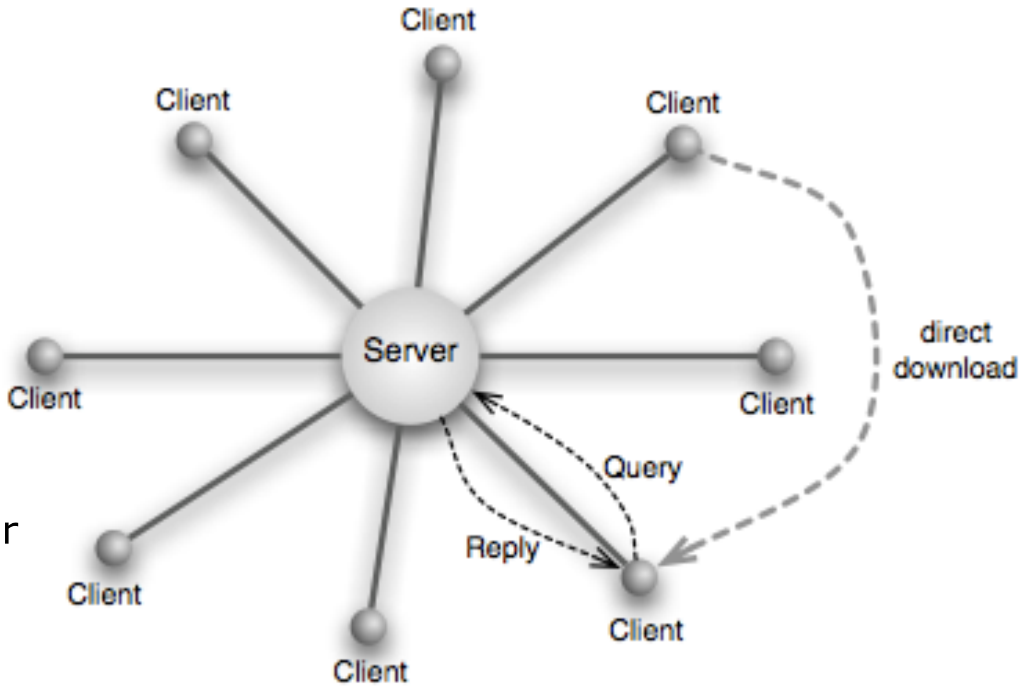
- Napster (1999-2000)
  - Central index server (all queries to server)
  - Guarantee to find files
  - Stopped by court decision

- Gnutella (2000)
  - No single point of failure
  - No guarantee that files are found
  - Flooding query model (queries involve many nodes!)
- FreeNet (2000)
  - Fully decentralized
  - Heuristic key based routing
  - No guarantee that files are found

- BitTorrent (2001)
  - First one to adopt DHT technology
  - **Attains both the decentralization of Gnutella and Freenet, but also efficiency and completeness of Napster**

Napster

- Client-Server structure (not really a P2P network)
- Server stores lists of clients an files
- Files themselves are stored on each client's local hard disk
- Downloading a file:
  - Client queries filename on server
  - Server replies the owner of the file
  - Client downloads directly from owner



- Comments:
  - Central structure enables censorship and is vulnerable
  - Napster does not scale up!
  - \+ Files are always found if they are in the network

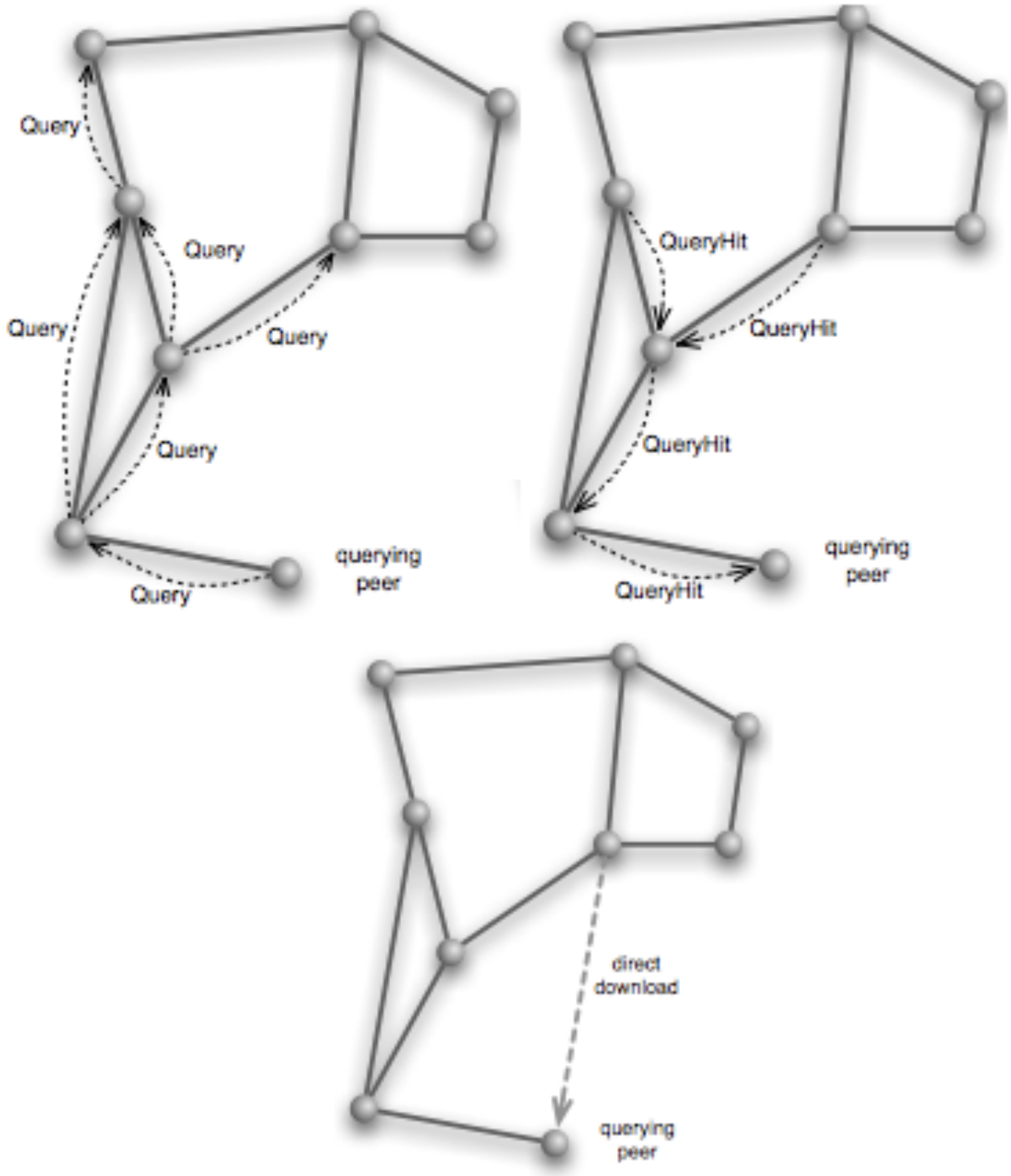
Gnutella

Bootsrapping / Connecting

- Initially, the client software holds a list of peers
- When bootsrapping, the client tries to connect to one node of the list
- From a found active node, up to *N* neighbors are queried (by sending a *ping message*)
  - The ping message contains a number named Time To Live (TTL) entry, which is decreased each time when passing a node
  - The message is not further routed when TTL=0
- From the returns of active peers (pong message) the list of peers is updated (and stored for the next bootstraping process)
- From the list the client randomly selects *k* peers as its neighbors

Gnutella
Query

- Queries (for files to download) are sent to all neighbors
- Queries are forwarded by each node until a maximum hop-distance (TTL entry)
- After a successful query (a node returned a queryHit message), the file is directly downloaded from that node
- Comments:
  - \+ Gnutella is scalable, and very robust and failsafe against attacks
  - – The depth-limited search (TTL value) only queries a fractions of the network
  - – When increasing the search depth, high messages density is the result, and thus high latency of queries



Distributed Hash-Table (DHT) I

- What is hashing in general?
  - To assign keys (e.g. filenames) evenly to a much smaller set (e.g. peers in the network)
  - In general, a hash function $f: K \rightarrow Q$ maps keys from K to hash values from Q
  - Example: We want to map filenames of songs to 5 nodes:
    - Hash function f(x) = x mod 5, since we have 5 nodes
    - The ASCII string of music.mp3 corresponds to the decimal number 870920545682538843149
    - Therefore, the hash value can be computed by f(870920545682538843149) = 4
- However, (conv.) Hash Tables can not be applied to P2P nets:
  - Nodes cannot be directly addressed such as memory. They can only be addressed by following the links of the network
  - Inserting and deleting a peer also implies readjusting the hash function
  - Therefore, inserting and deleting nodes is inefficient

Distributed Hash-Table (DHT) II

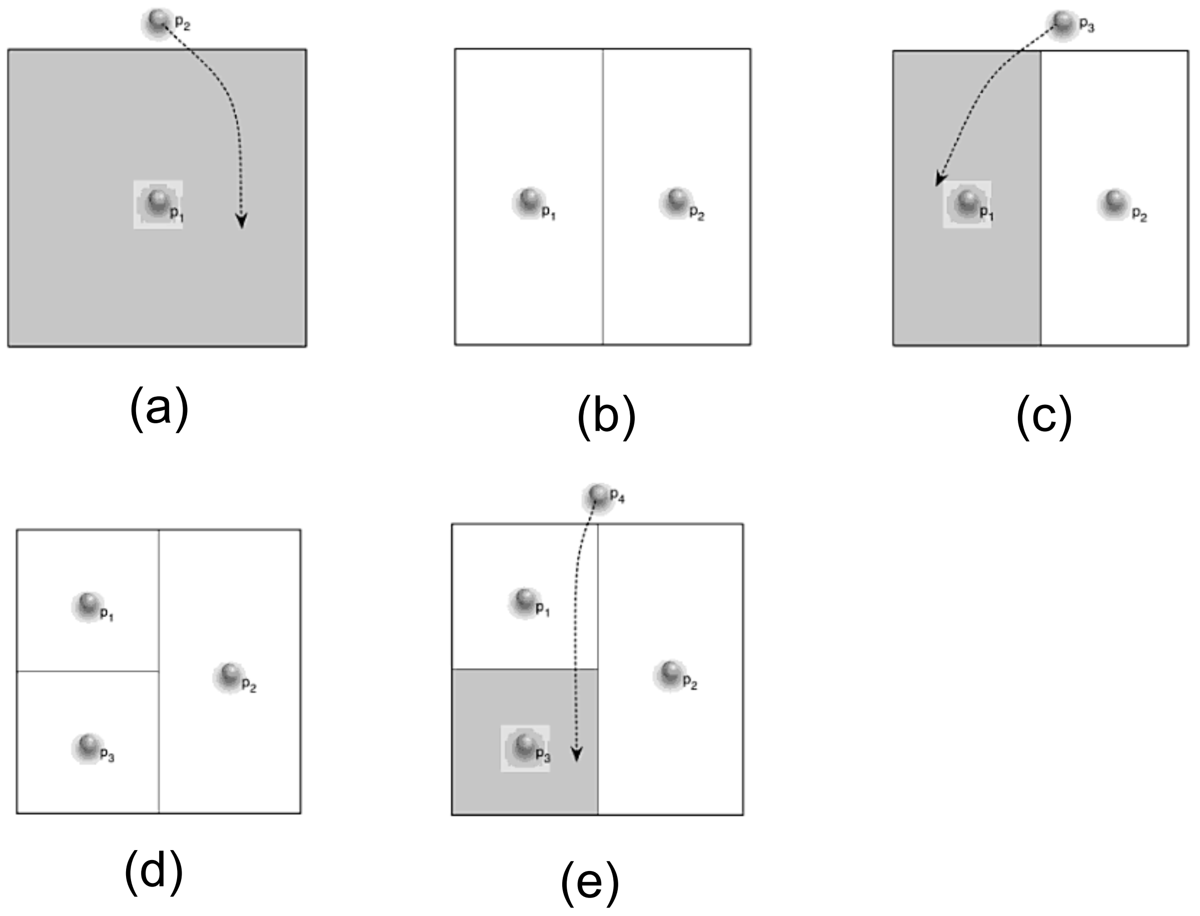
Implementation in CANs

- A DHT is a distributed data structure holding the mapping from keys (e.g. MD5 sums) to file locations (e.g. IPs)
- In Content Addressable Networks (CANs) Distributed Hash Tables are used were
  - hash values are in the two dimensional space of a square Q:

$$(x,y)\in[0,1)\times[0,1)$$

  - Note: values can also be defined for a hyper cube, i.e. d>2
  - Q is partitioned into rectangles where each rectangle belongs to one peer
  - Each peer is responsible for all files assigned to its rectangle
  - Bootstraping:
    - Initially, the whole square is owned by the first peer
    - When inserting a new peer p, a point z in Q is chosen randomly
    - The owner p’ of the rectangle around z is queried
    - The rectangle of p’ is halved and the network structure (list of neighbors of each peer) is adjusted

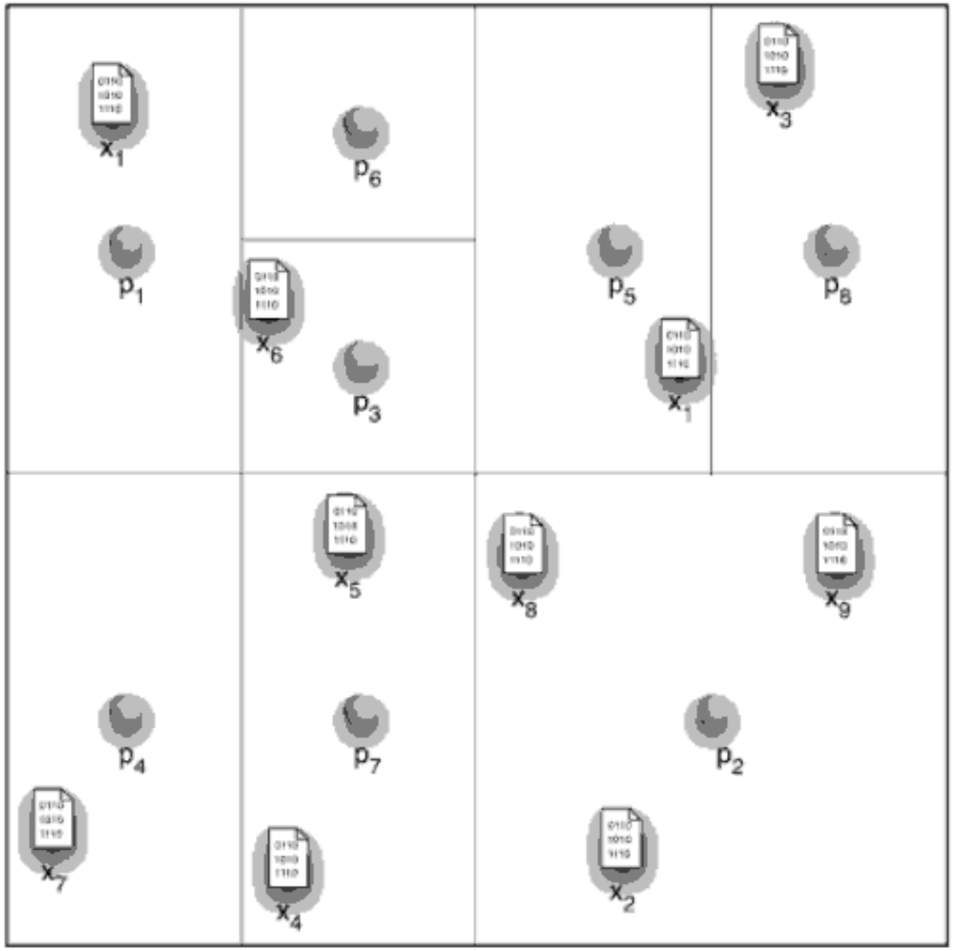
Distributed Hash-Table (DHT) IV
Inserting Nodes into the CAN



Distributed Hash-Table (DHT) III

Assignment of data to nods in CAN

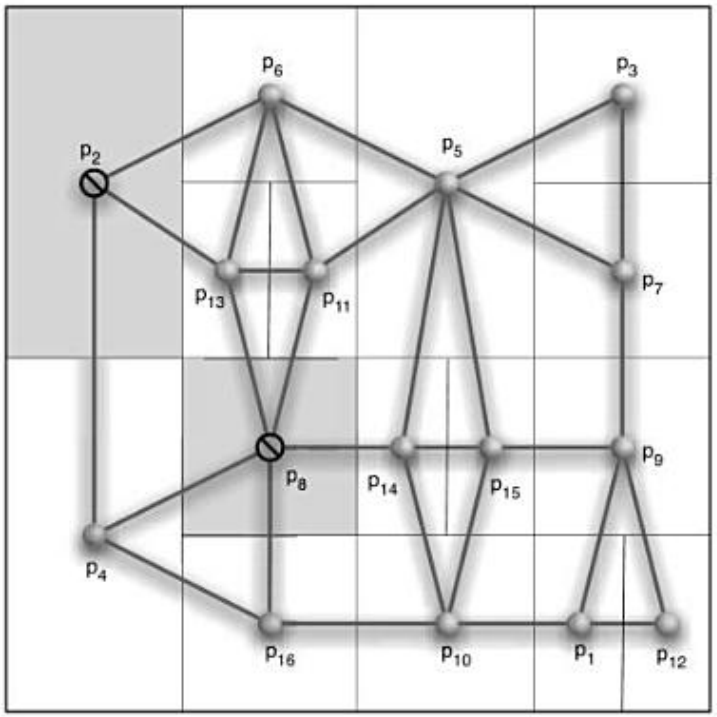
The Hash function has to guarantee that data (e.g. filenames, MD5s) are equally distributed on the square



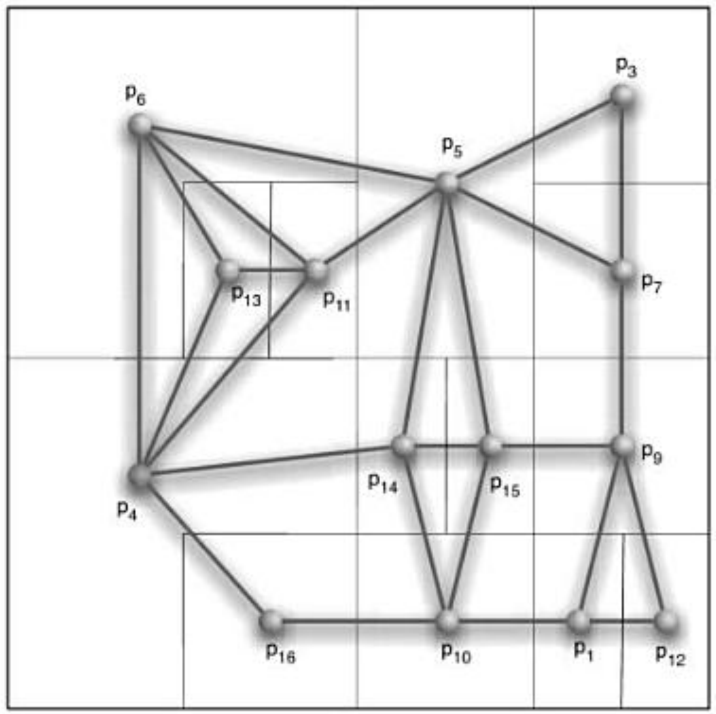
CAN Structure and Routing

- Local connections
  - Each peer maintains connections to other peers neighboring its rectangle
  - When inserting a new node, neighbor peers are adjusting their information accordingly
- Routing in CAN
  - First, compute the position P of the data by the hash function
  - Second, forward the message to the neighbor closest to P until reaching the maintaining node
  - Expected number of hops when squares are equally sized: $O\left(\sqrt{n}\right)$
- When peers are leaving…
  - … they typically do not announce it
  - Thus, peers continuously test their neighborhood with a ping message
  - The first neighbor that detects a missing peer takes over its area
  - Therefore, peers can be responsible for many rectangles
  - However, repeated insertions and deletions lead to a fragmentation of the network!

CAN Fragmentation When Nodes Are Leaving



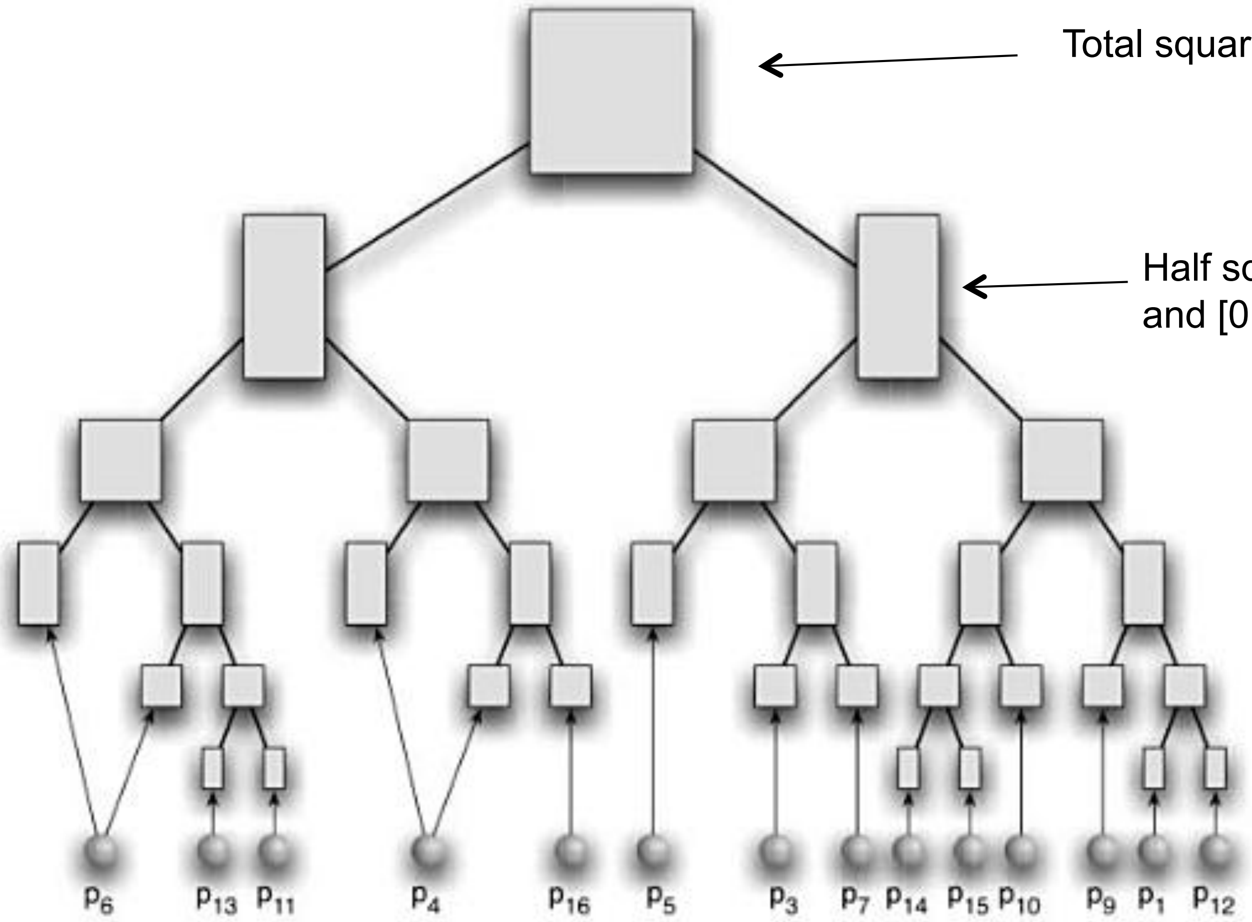
Peer 2 and Peer 8 are leaving the network



Peer 4 takes over the area of Peer 8 and Peer 6 the one of Peer 2

CAN Defragmentation I

Network from last slide represented as binary tree



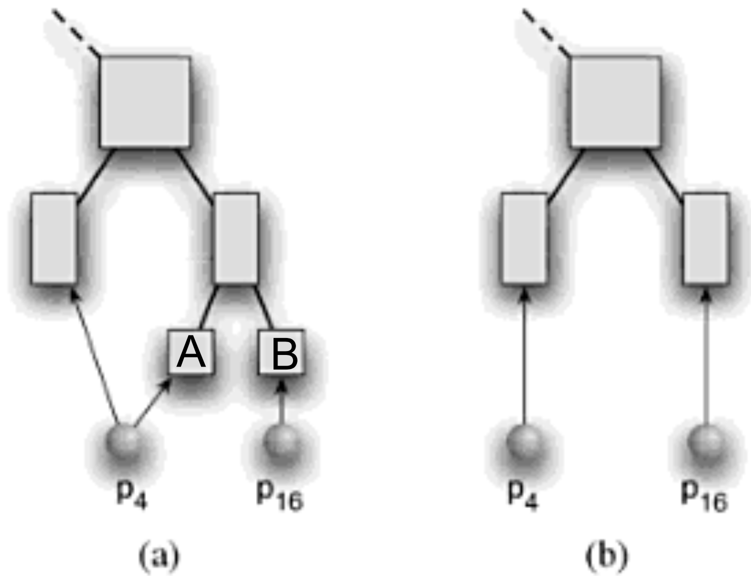
Total square Q

Half squares [0.5,0)x [0,1) and [0,0.5)x [0,1)

Leaves containing peers

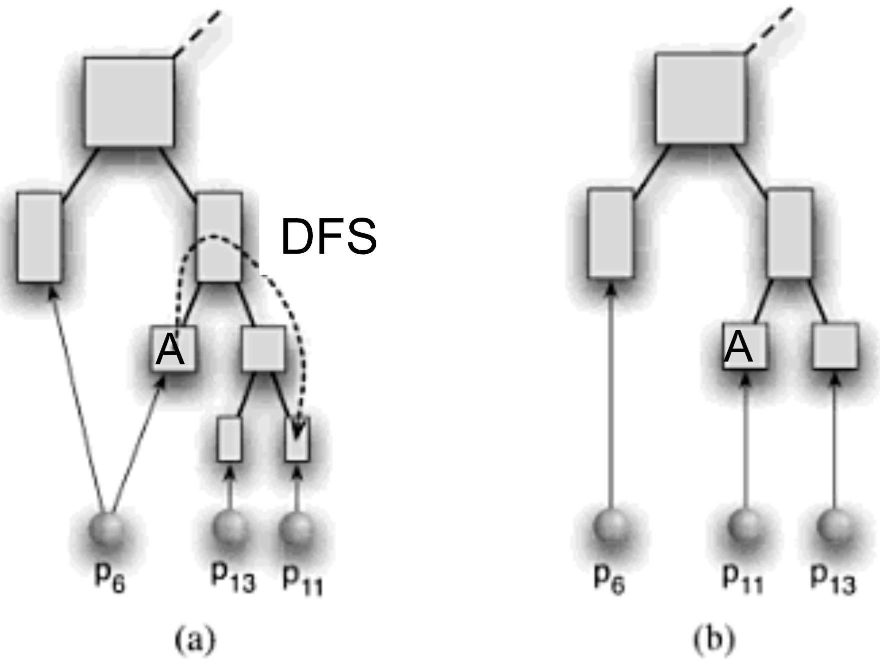
CAN Defragmentation II
Simple Case

- Defragmentation can be initiated by peers having more than one rectangle
- This is done by handing over their smallest rectangle A to another peer
- Simple case:
  - the brother tree of A consists of a single leaf B only
  - Then A can be handed over to the peer (P16) in charge of B
  - Both rectangles are subsequently merged by this peer



CAN Defragmentation III
Difficult Case

- Difficult case:
  - the brother tree of A consists **not** of a single leaf
  - Then the peer (p6) performs Depth First Search (DFS) until it finds two neighboring leaves
  - Both leaves are merged and now controlled by a single peer (p13)
  - The released peer (p11) is then assigned to A



Case-study: DHTs For Mobile Robot Teams Solving Intra-Logistics Tasks

Motivation

- A remarkably high degree of automation has been reached in production and intra-logistics nowadays
- However, handcarts and forklifts manually steered by humans are still indispensable in many of situations
  - For example, boxes filled with small parts by a automated picking system have to be delivered to packing stations
- Fixed installations exists, for example, conveyors either overhead- or floor-based
  - Drawback: when the business model of the company changes existing installations have to be redesigned
- The Vision:
  - To build-up a team of autonomous and decentralized units communicating on a low-range-basis with each other
  - A team consisting of hundreds of robots organizes material flows autonomously and decentralized

Existing Intra-Logistics System: KIVA

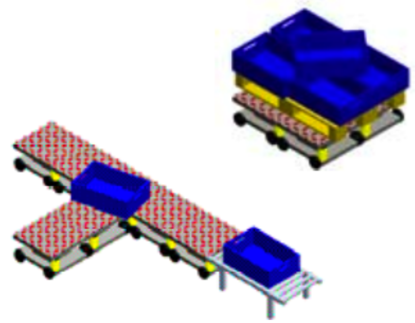
The KIVA System

- Strengths:
  - Comparably cheap robots (no laser scanner)
    - Cheap localization via barcodes in the ground
  - Robots optimize their controller online for reducing misalignments
  - Simplified path planning due to grid structure of the shelves
  - Virtual highways: Multiple paths are joined to one highway
- Drawbacks:
  - Centrally controlled (might not scale-up)
  - Cannot operate in environments with humans
  - Cannot be integrated in arbitrary environments, i.e. needs a large hangar-like structure
  - Environment has to be engineered (barcodes)

Karis (Kleinskalige Autonomes Redundantes Intralogistiksystem)

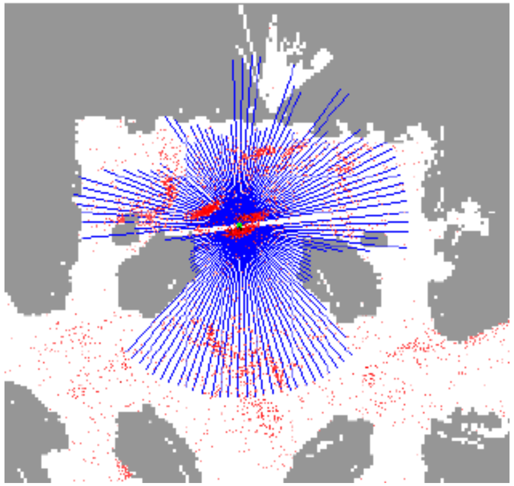
- Goal:
  - Team of 100 decentralized “elements” to accomplish autonomously transportation tasks
- Features:
  - Automatic load and unload at assembly chains
  - Automatic battery recharging via the ground
  - Mechanism to couple with stations or other vehicles
- Challenges:
  - Navigation and coordination of decentralized teams

KARIS element with conveyer

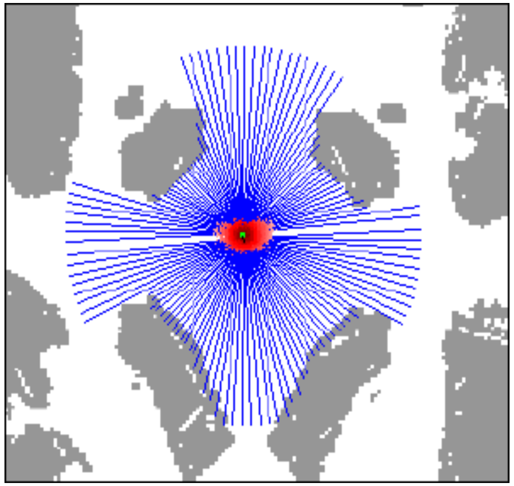


The Vision: KARIS elements teaming up for carrying larger goods or building assembly lines

Karis Navigation I
Monte-Carlo Localization (MCL)



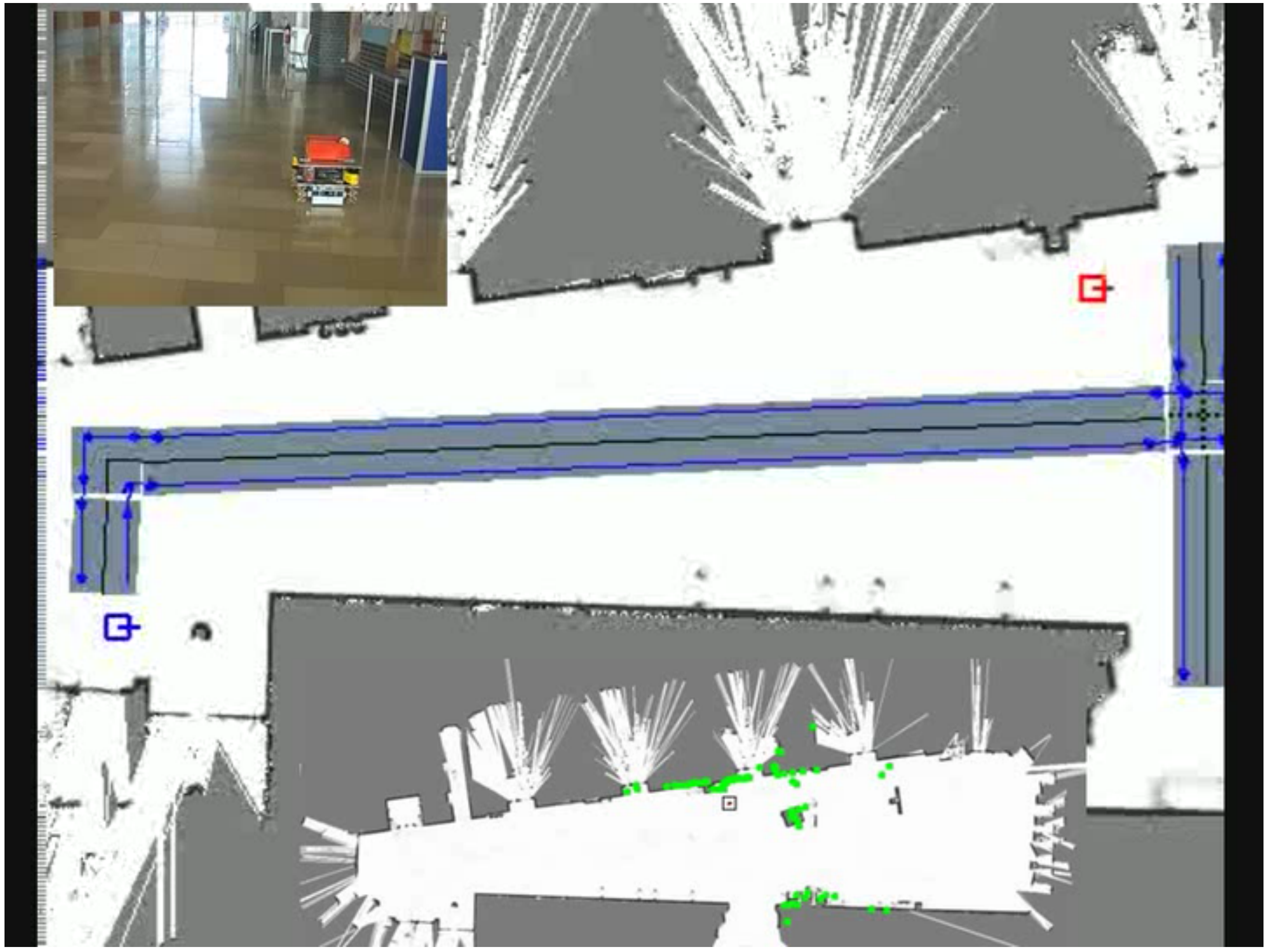
- **Measuring** the distance (blue) to surrounding objects (grey) with a laser range finder
- **Particel-Filter**: Method to compute the robot pose, where the estimate represented by a set of particles (red). Each particle represents a posssible pose of the robot



- **Prediction step**: For each particle a position is sampled according to the motion model
- **Corection step:** Each particle is weighted according to the current observation (LRF) and the sensor model
- **Selection:** New particles are chosen with a probability proportional to their weight

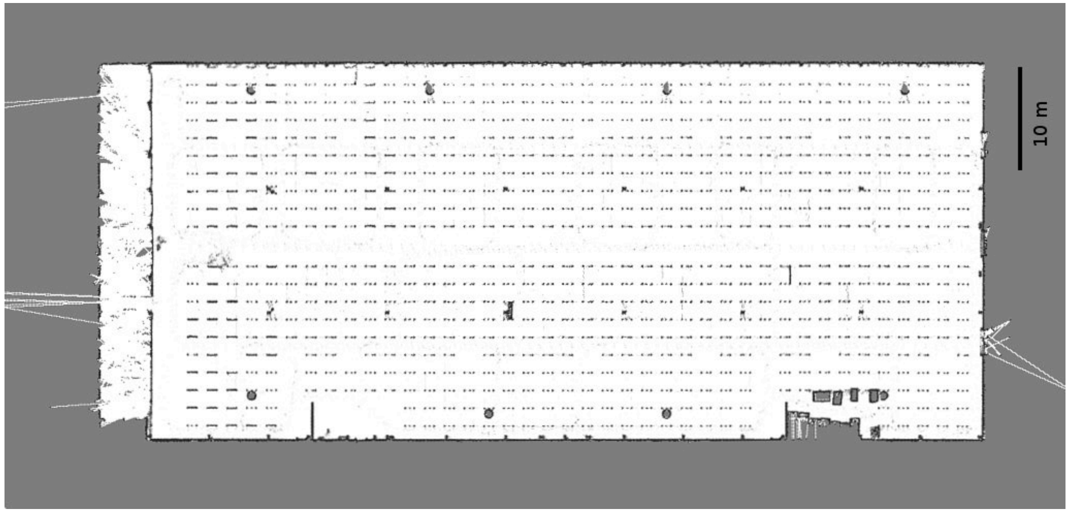
Karis Navigation II

Monte-Carlo Localization (MCL) & A* Planning

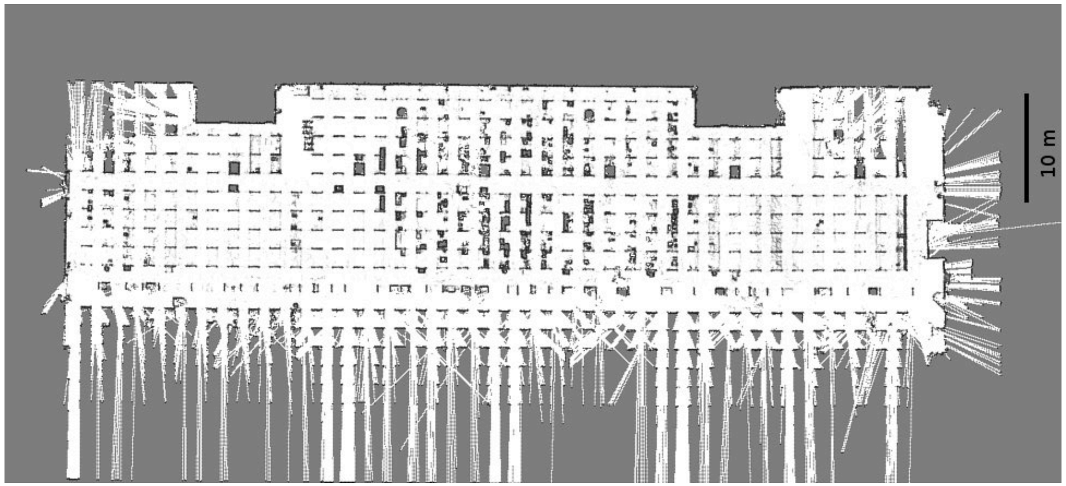


Karis Navigation III

Larger GridMap generated at a logistics company



3rd floor



2nd floor

Google Map Image

DHTs for assigning robots to stations

Problem Description

- Boxes are queued at loading stations
  - coming from an outer infrastructure such as trucks or automated shelves
- Robots have to deliver boxes between loading stations
- Wish list:
  - Minimal *worst case time* delivery
  - Maximal *efficiency* (e.g. minimize waiting or blocking of robots)
  - Truly decentralized & autonomous to avoid single point of failure
  - Low network traffic, (i.e. no broadcasts ála Gnutella)
- Challenges:
  - Travel times between stations can change (i.e. new obstacles in the path, wheel malfunctions, etc.)
  - Robots can be inoperable
  - Load, i.e., number of boxes arriving at stations can vary
- Claim:
  - DHT solution can solve these three problems

DHTs for assigning robots to stations

Performance Metric

Throughput rate T_r:

number of boxes dispatched per minute
(can simply be counted over time)

Max. possible throughput rate $T_{r\text{-}max}$:

MIN(# boxes arriving , max due to latency)

Computation of efficiency e_i : $e_i = \frac{T_r}{T_{r_{\max}}}$

➡ *In other words: relation between current performance and max possible performance.*

DHTs for assigning robots to stations

Weighted Distance

Weight expresses how eligible a station is for being served →

$$w_i = \frac{1}{e_i} \frac{N_Q - N_C}{N_{delivered}}$$

Computed & published by stations (SSI)

N_Q	current queue length of station i
N_C	# of robots assigned to the station i
$N_{delivered}$	# of totally delivered packages (bounded 20min)
e_i	efficiency of station i

$$D_i = \frac{\log(d_i)}{w_i}, for\ d_i > d_{\min}$$

Mobile nodes selects at each time the station with **min(D_i)**

Mobile Content Addressable Network (MCAN)

Message Traffic and Network Repair

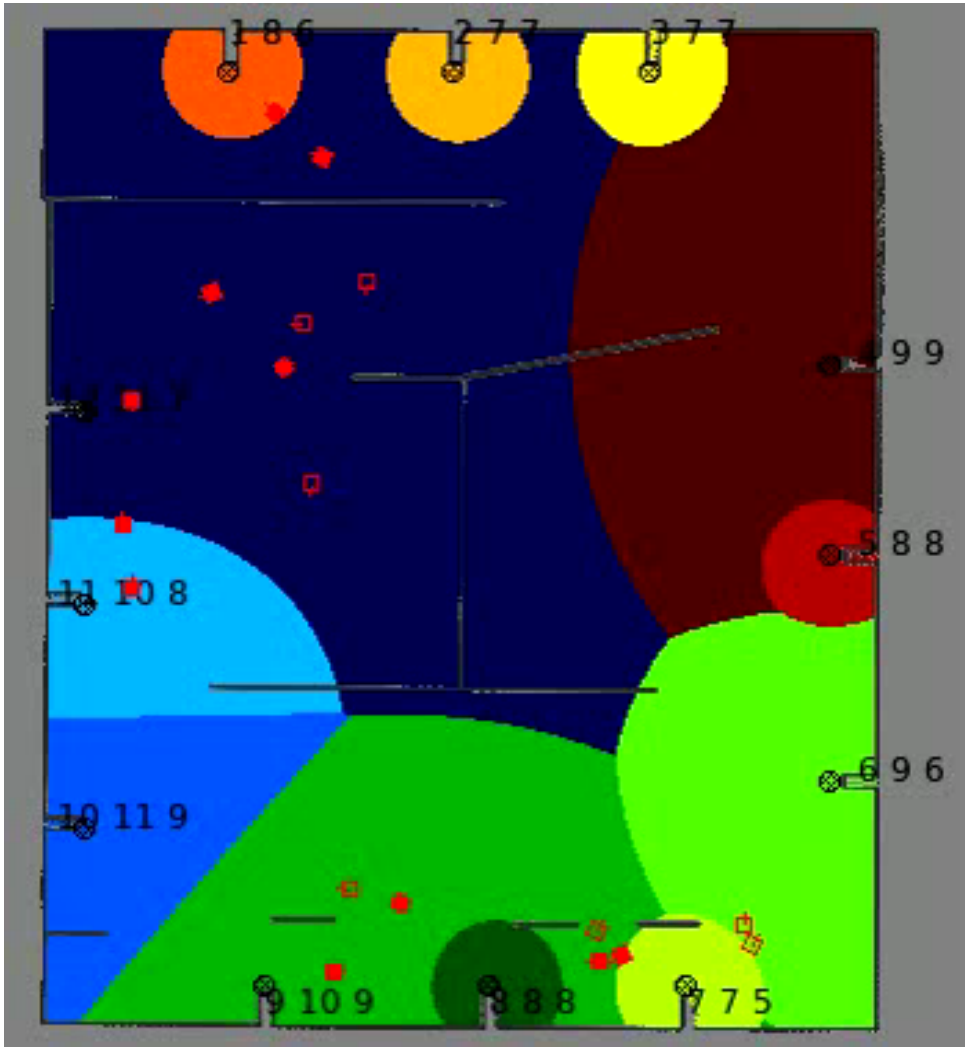
- Geographic Routing:
  - Locations of stationary nodes (loading stations) given
  - Routing to neighbor which is nearest to destination
  - Can generally not route to mobile nodes since their location changes!
- Stations broadcast Station Status Info (SSI) reflecting their statistics
  - Each mobile node forwards the SSI to its neighbors
  - However, TTL of SSI messages is limited to the area defined by D_i
  - Therefore, no network wide broadcast!
- Automatic network repair
  - When memorized SSI_j of station j is too old, move towards station j

Mobile Content Addressable Network (MCAN)

Bootstraping / Construction

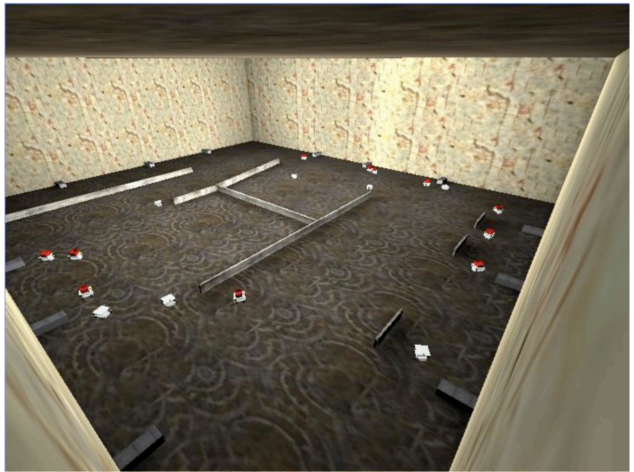
- MCAN construction:
  - Mobile node started in the network area
  - Search for the network, i.e., contact the nearest node in communication range
  - Receive SSIs from all stations via the contact node and compute for each SSI the D_i **=** $f(r_x, r_y, SSI_i)$
  - Select station s_j with $\min(D_i)$
  - Send REQ to s_j and go towards region of s_j
  - When in neighbor range of s_j: negotiate for a delivery task (Contract-Net Protocol)

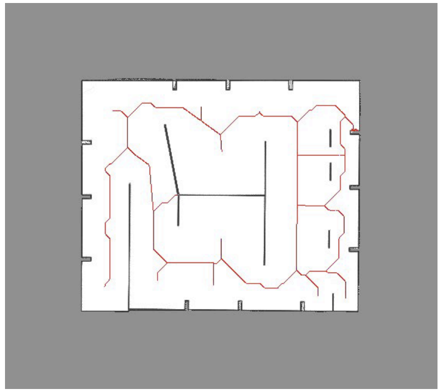
Visualization of max(D_i)

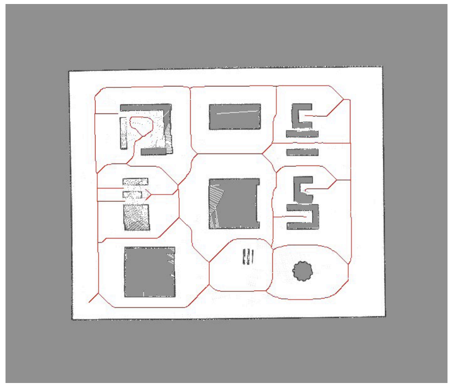


Simulation Results I

Environments In USARSim

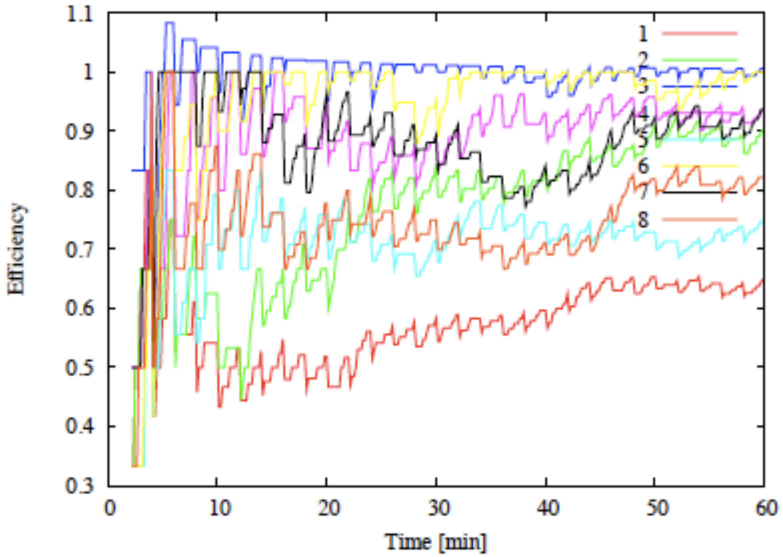




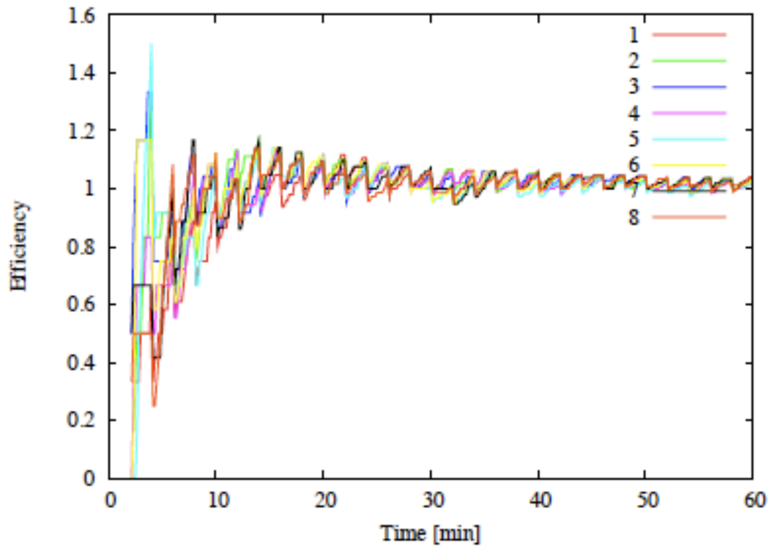


Simulation Results II

Comparing DHT Solution with the Baseline



Baseline

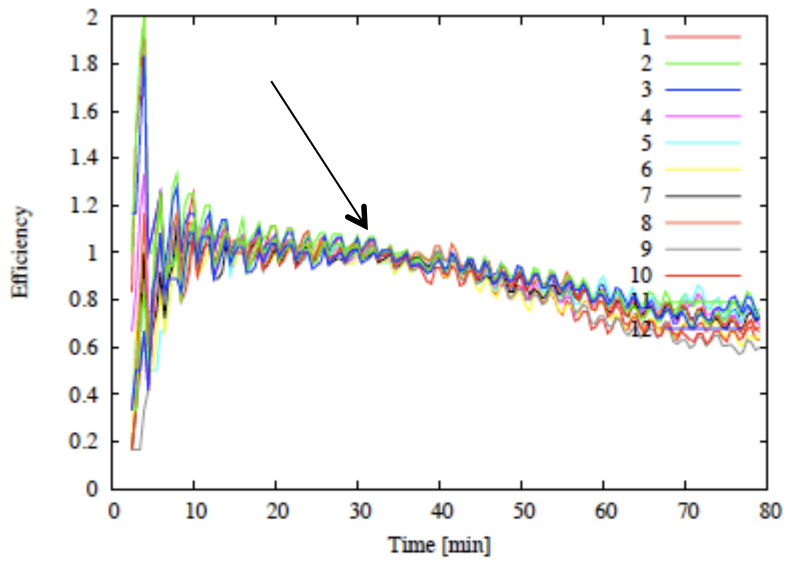


DHT

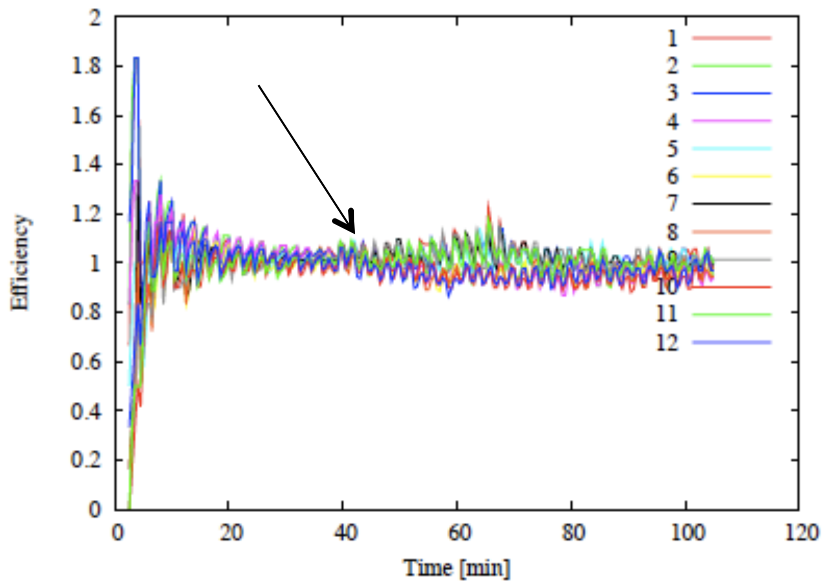
The baseline approach assigns robots according to their distance to stations. Robots receive task offers from all stations and decide for the station with the shortest distance.

Simulation Results III

Adaption to Sudden Change



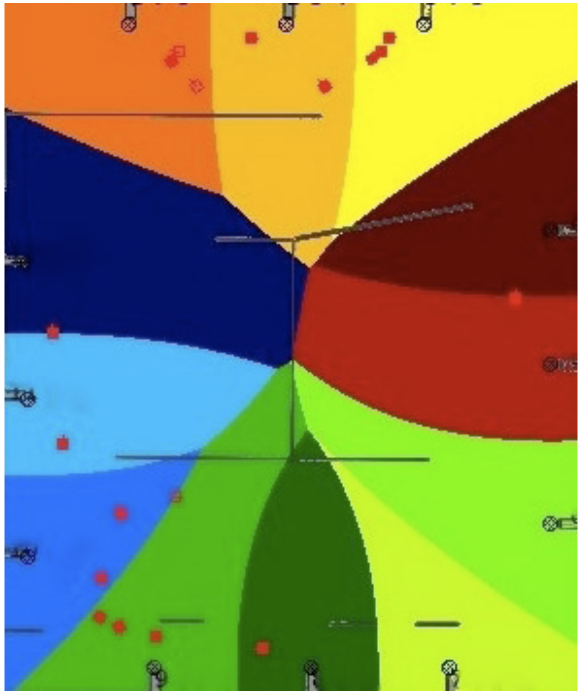
Running with 18 robots. After 30 min 10 robots were killed



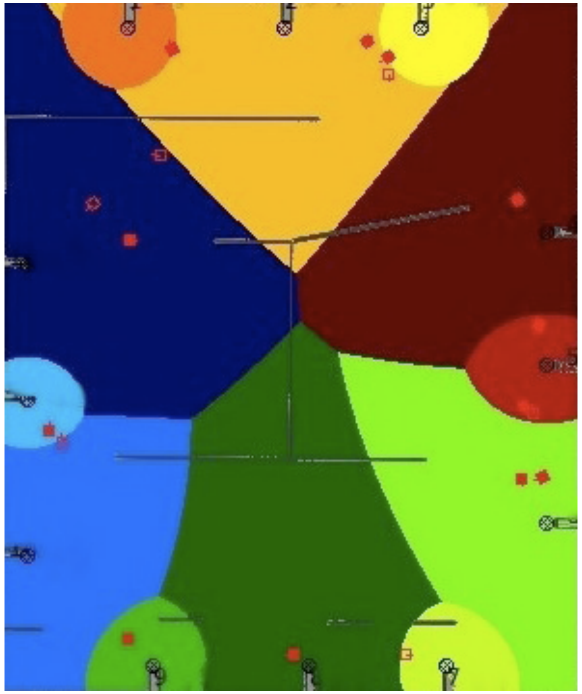
Change of station load: Till 40 min running with 4box/sec, then, 6 stations with 2/sec and 5/sec

Simulation Results IV

Visualization of max(D_i) when station load changes



12 stations with 4 boxes per minute

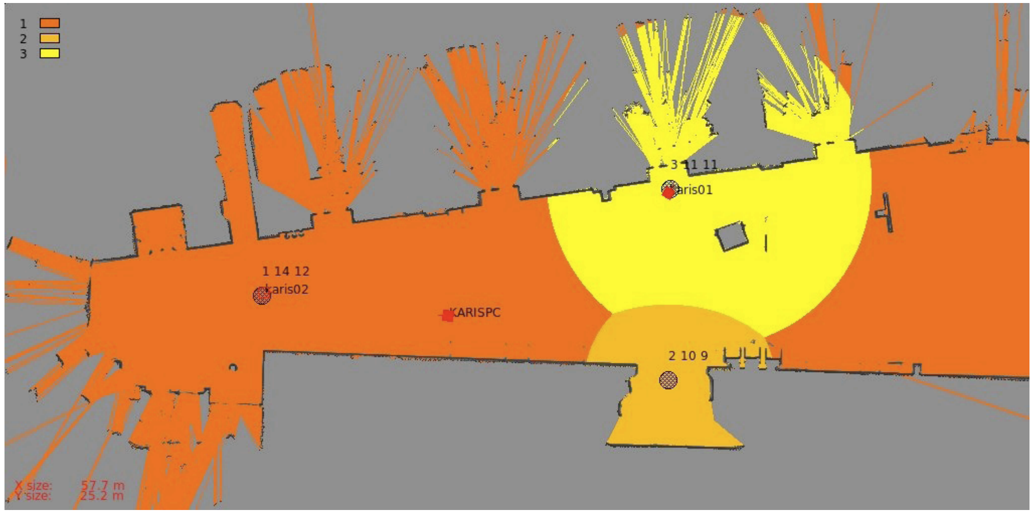


6 stations with 2 boxes per minute and six stations with 5 boxes per minute

Real-World Results I

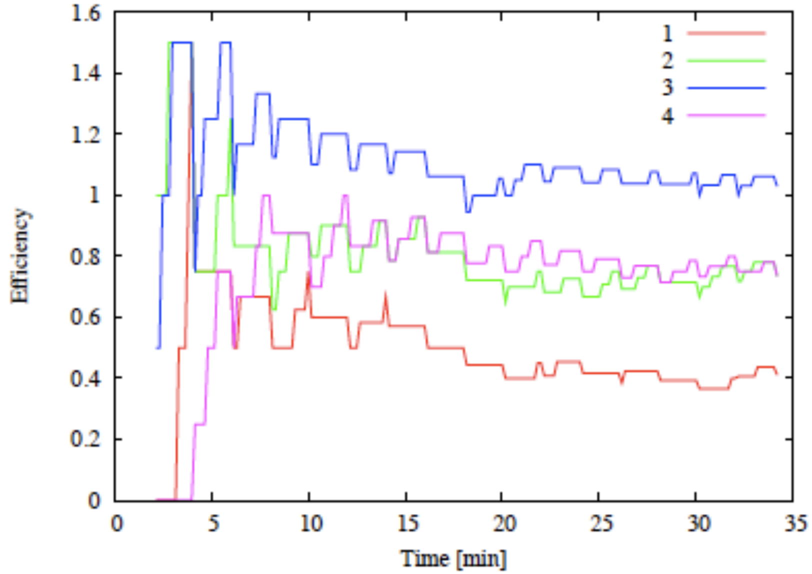
Experiments in 101 building

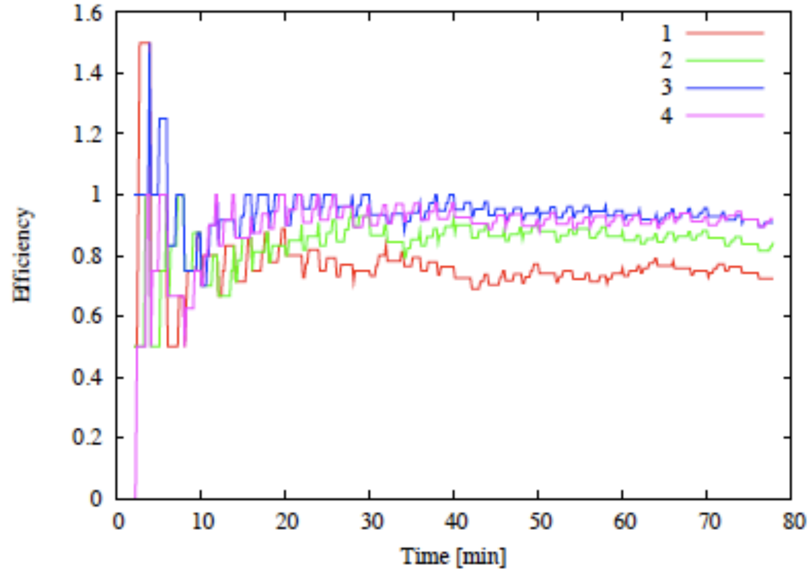
Team of 3 robots



DHT distribution of robots
to 3 stations

Real-World Results
Efficiency from 3 robots with 4 stations





	Three stations	Four stations
Base eff. [%]	0.69± 0.1	0.74 ± 0.24
WHHT eff. [%]	0.76± 0.05	0.9 ± 0.09
Base deliv. [#]	36.3± 4.9	20.3 ± 6.5
WHHT deliv. [#]	24 ± 1.6	25.5 ± 2.7
Base w. time [min.]	20.1	18.3
WHHT w. time [min.]	14.5	10.5

Real-World Results

Video

Summary

- The development of peer-to-peer systems on the Internet indicates the need for decentralized solutions when the number of clients increases
- Decentralized Hash Tables have proven to be a strong mechanism for this problem
- In the future they might also play an important role in multi agent systems, at least, when the number of agents is significantly large

Literature

- Peter Mahlmann und Christian Schindelhauer, **P2P Netzwerke: Algorithmen Und Methoden**, Springer 2007
- Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S. **A scalable content-addressable network**, *Computer Communication Review. Volume 31., Dept. of Elec. Eng. and Comp. Sci., University of California, Berkeley* (2001) 161–172.
- David Karger, Eric Lehman, Tom Leighton, Mathhew Levine, Daniel Lewin, Rina Panigrahy, **Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web**, *STOC 1997*
- D. Sun, A. Kleiner, C. Schindelhauer, **Decentralized Hash Tables For Mobile Robot Teams Solving Intra-Logistics Tasks,** to appear AAMAS 2009