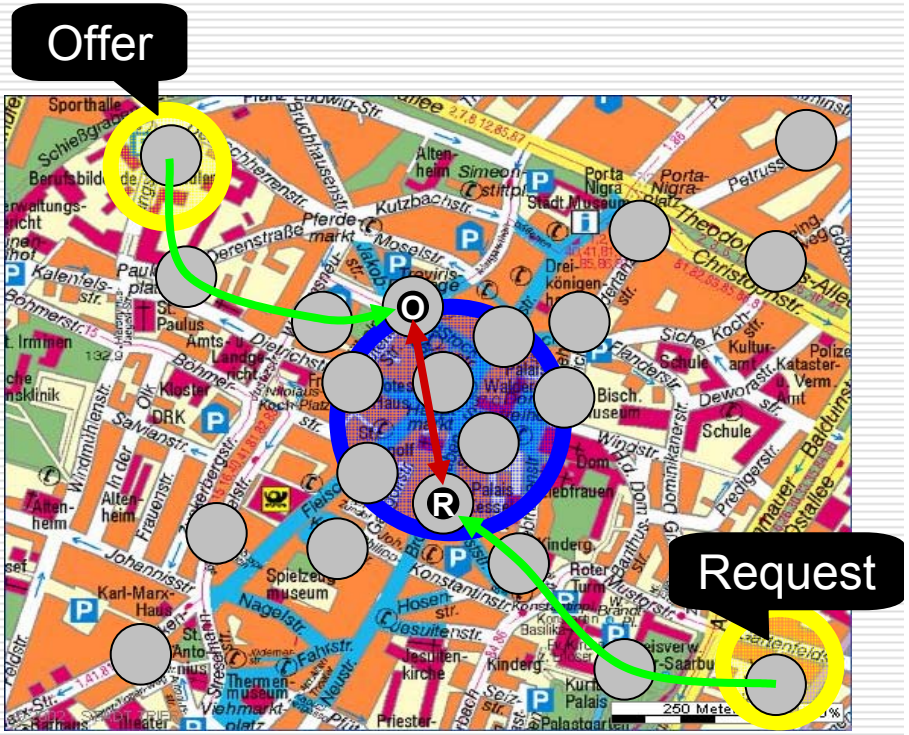
A scalable workbench for implementing and evaluating distributed applications in mobile ad-hoc networks

Johannes K. Lehnert, **Daniel Görgen**, Hannes Frey, Peter Sturm

System software and distributed systems
University of Trier
Germany

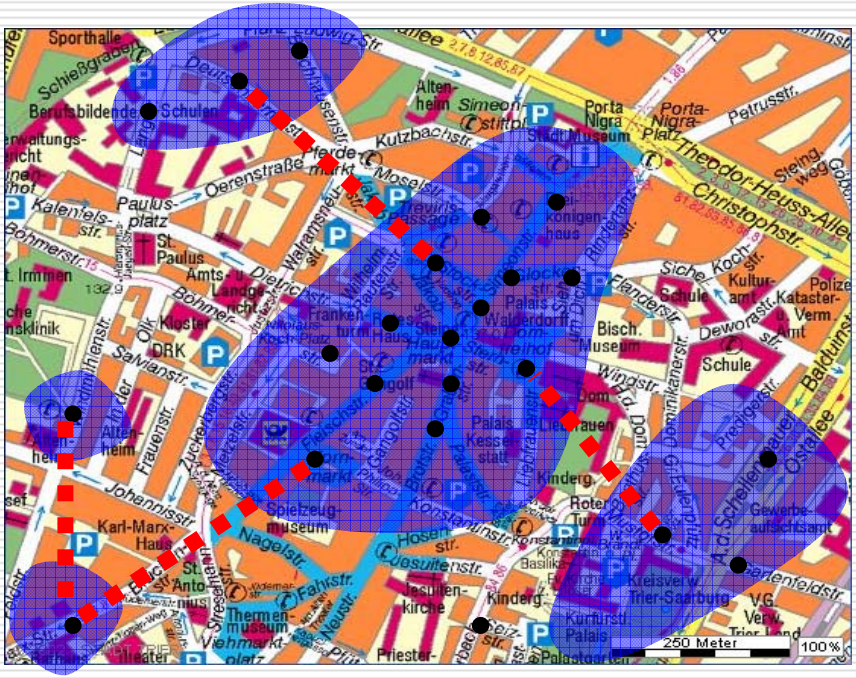
Mobile multihop ad-hoc networks

- Metropolitan sized networking
- Mobile devices
  - Wireless communication facilities
  - Localized location computation
- Direct communication only within transmission range
- Unpredictable network topology changes due to mobility
  - Network partitions
  - Permanent link failures



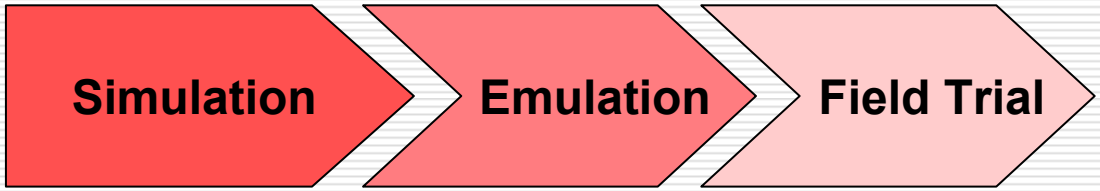
Mobile multihop ad-hoc networks

- Metropolitan sized networking
- Mobile devices
  - Wireless communication facilities
  - Localized location computation
- Direct communication only within transmission range
- Unpredictable network topology changes due to mobility
  - Network partitions
  - Permanent link failures



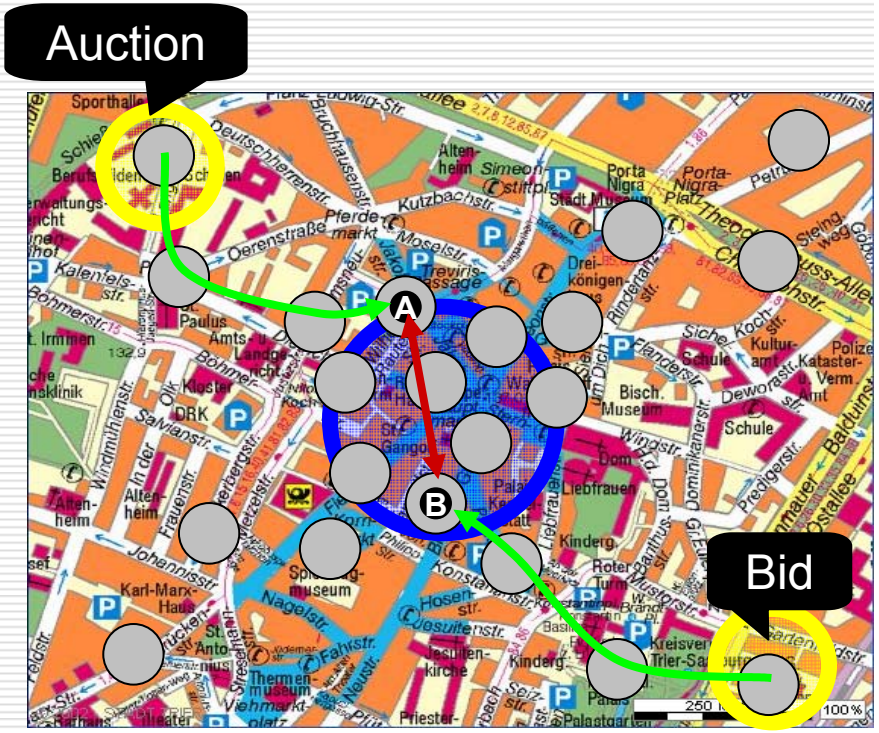
Application development in mobile multihop ad-hoc networks

- Challenging area
  - State-of-the-art still an open question
  - Self-organization
  - Small devices with many limitations
- Field trials expensive
  - Time, money, hardware, people
  - Critical mass needed for serious tests
- Uniform workbench
  - Develop and test in simulation first
  - Evaluate application in emulation
  - Use the *same* code in field trials



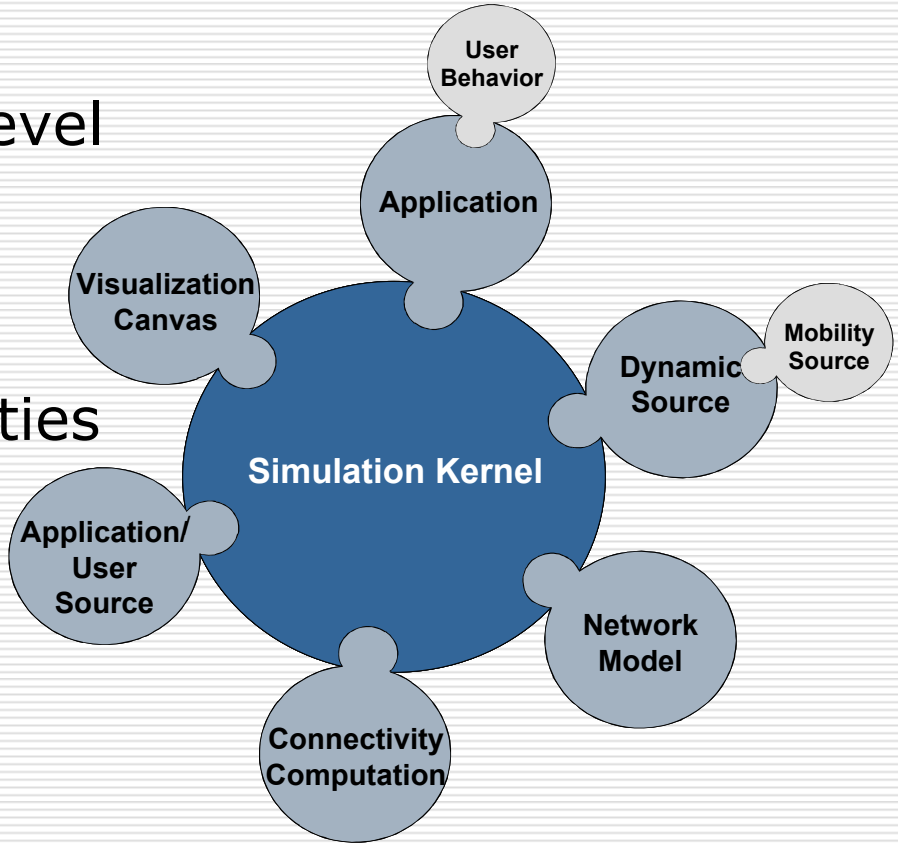
Case Study: UbiBay

- Self-organizing auction system for mobile multihop ad-hoc networks
  - Find running auctions
  - Bid on auctions
  - Start own auctions
  - Intended for low-value goods
- Based on *marketplace communication pattern*
  - Devices act on behalf of others
  - Negotiation takes place at central marketplace
- Developed using workbench & proposed development process
  - Simulation ✓
  - Emulation ✓
  - Field trials ✓ (for few devices), larger tests planned



Workbench: Simulation

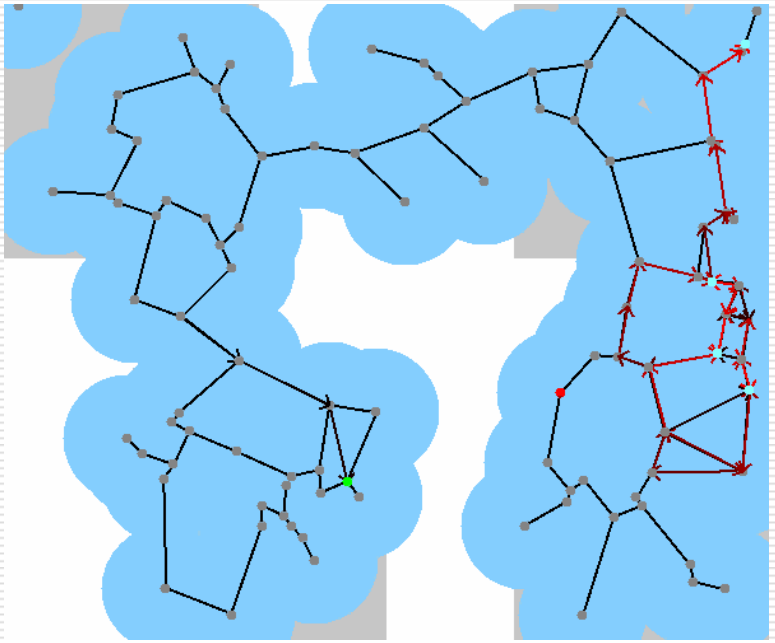
- Scalable
- Intuitive, high abstraction level
- Powerful visualization
- Extensible
- Focus on topological properties
- Code reuse



- "Concentrate on development, not on the simulator!"
- "Faster than real-time"

Workbench: Simulation II

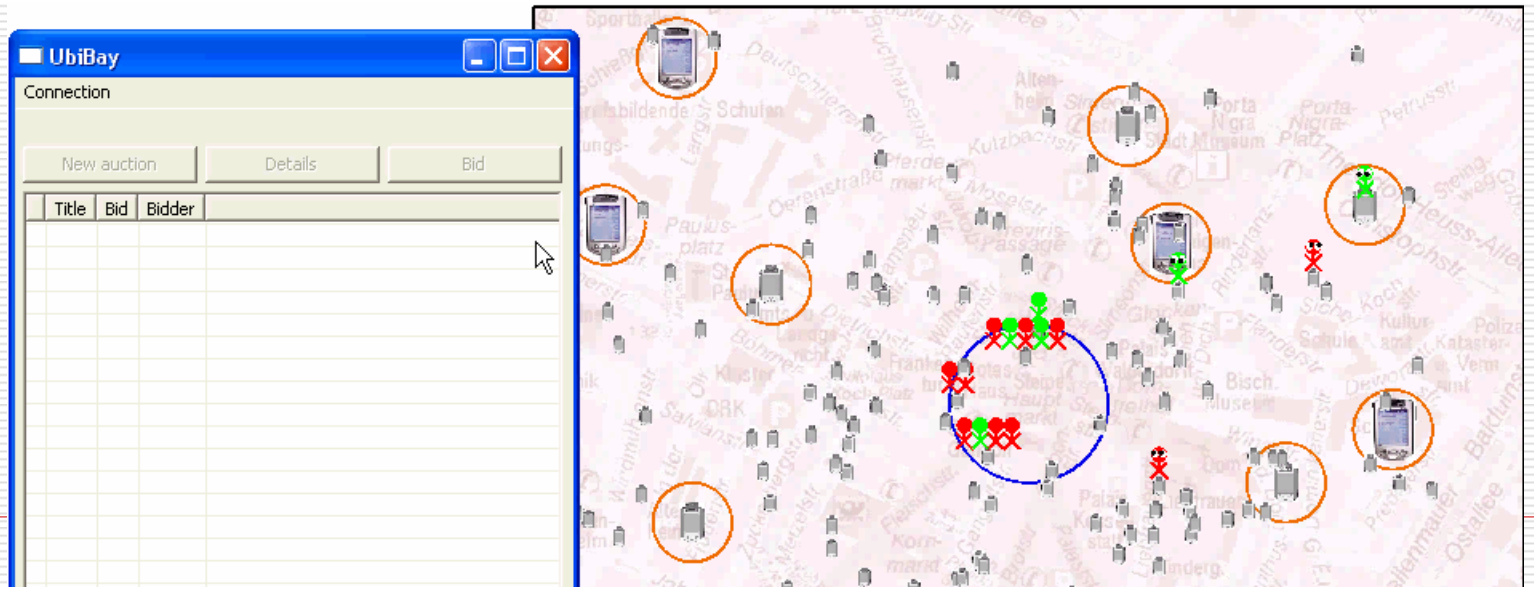
- Extensible
  - Components defined as interfaces
  - Many default implementations (mobility, connectivity, network)
- High abstraction level
  - Register as listener for neighbor discovery
  - Network messages = Java objects
- Scalable
  - 10000 devices possible
  - Precomputation for mobility and connectivity
- Visualization
  - Freely definable
  - Multiple output targets: Swing/Java2D, OpenGL, PostScript, …



Protocol: GPSR
Mobility Model: Restricted Random Waypoint
Traffic Source: CBR

Workbench: Hybrid mode

- Simulate network and devices
- Connect workstations or other devices to simulation
  - Replace simulated user behavior with GUI
  - RMI server controls simulation kernel
  - Mix of simulated and real user behavior possible
- Valuable for debugging
- “Get a feeling for the application”



Workbench: Real hardware

- Execution environment identical to simulation
  - Multiple threads, synchronization queues
  - Network implementation: WLAN + UDP unicast/broadcast
  - Positioning: GPS receivers
  - Neighbor discovery: periodic broadcasts
  - GUI: reused from hybrid mode
- Current implementation: PocketPC with IBM J9 VM

Summary

- Workbench approach works
  - Scalable: simulate thousands of devices in real-time
  - Intuitive and productive programming environment
  - Code reuse very effective
- Java is the right choice
  - Fast, powerful environment
  - Available even on small devices
  - “Write once, run anywhere” facilitates uniform workbench approach
  - Eclipse IDE makes it even more attractive
- It’s not finished:
  - Provide more mobility models
  - “Realistic” network model
  - Allow feedback from visualization