Mobile Emulab: A Robotic Wireless and Sensor Network Testbed

D. Johnson, T. Stack, R. Fish, D.M. Flickinger, L. Stoller, R. Ricci, J. Lepreau

School of Computing, University of Utah
(Jointly with Department of Mech. Eng.)
www.emulab.net

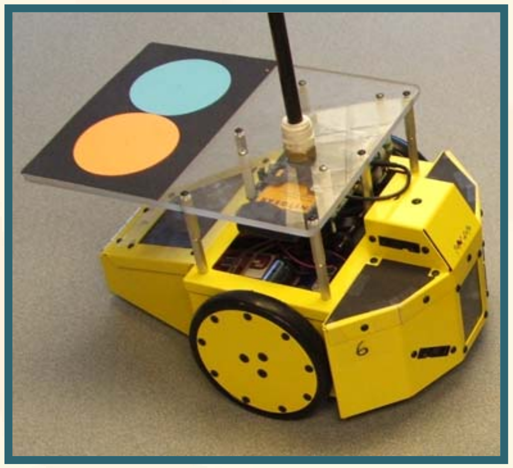
IEEE Infocom, April 2006

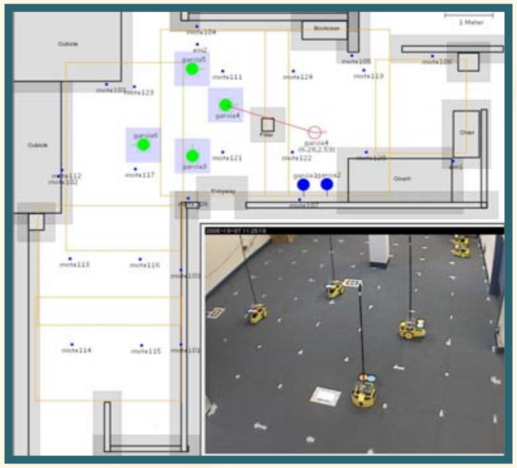
Need for Real, Mobile Wireless Experimentation

- Simulation problems
  - Wireless simulation incomplete, inaccurate (Heidemann01, Zhou04)
  - Mobility worsens wireless sim problems
- But, hard to mobilize real wireless nodes
  - Experiment setup costly
  - Difficult to control mobile nodes
  - Repeatability nearly impossible
- Must make real world testing practical!

Our Solution

- Provide a real mobile wireless sensor testbed
  - Users remotely move robots, which carry sensor motes and interact with fixed motes





Key Ideas

- Help researchers **evaluate WSN apps under mobility with real wireless**

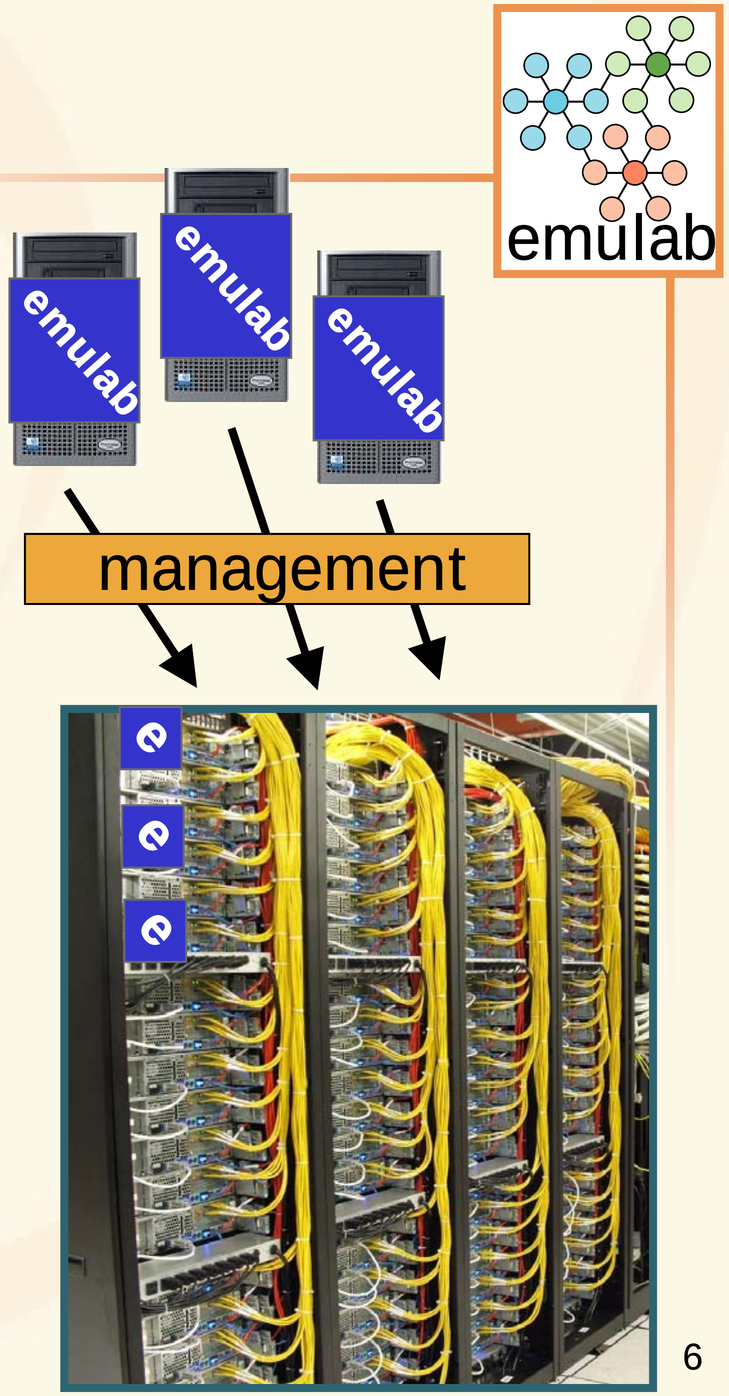
- Provide easy remote access to mobility
- Minimize cost via COTS hardware, open source
- Subproblems:
  - Precise mobile location tracking
  - Low-level motion control

Outline

- Introduction
- **Context & Architecture**
- Key Problem #1: Localization
- Key Problem #2: Robot Control
- Evaluation
  - Microbenchmarks
  - Data-gathering experiment
- Summary

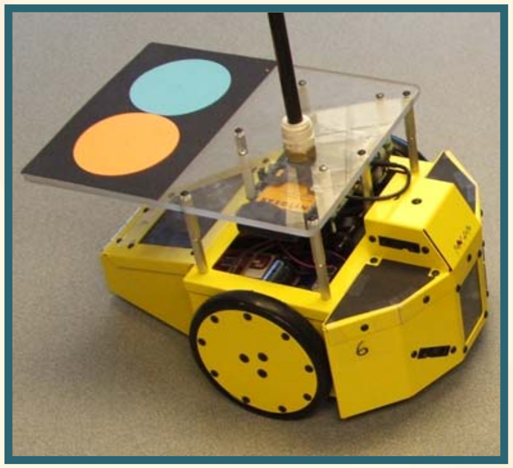
Context: Emulab

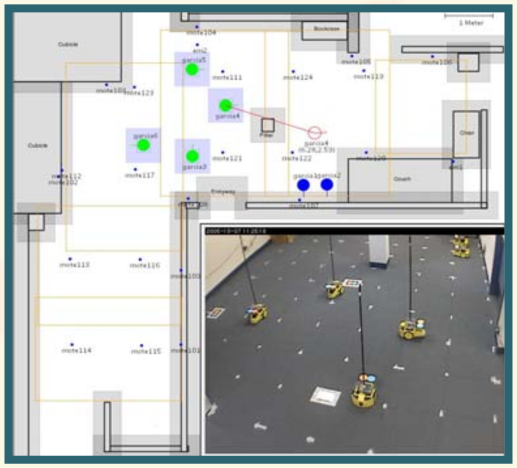
- Widely-used network testbed
  - Provides remote access to custom emulated networks
- How it works:
  - Creates custom network topologies specified by users in NS
  - Software manages PC cluster, switching fabric
- Powerful automation, control facilities
- Web interface for experiment control and monitoring
- Extended system to provide mobile wireless…



Mobile Sensor Additions

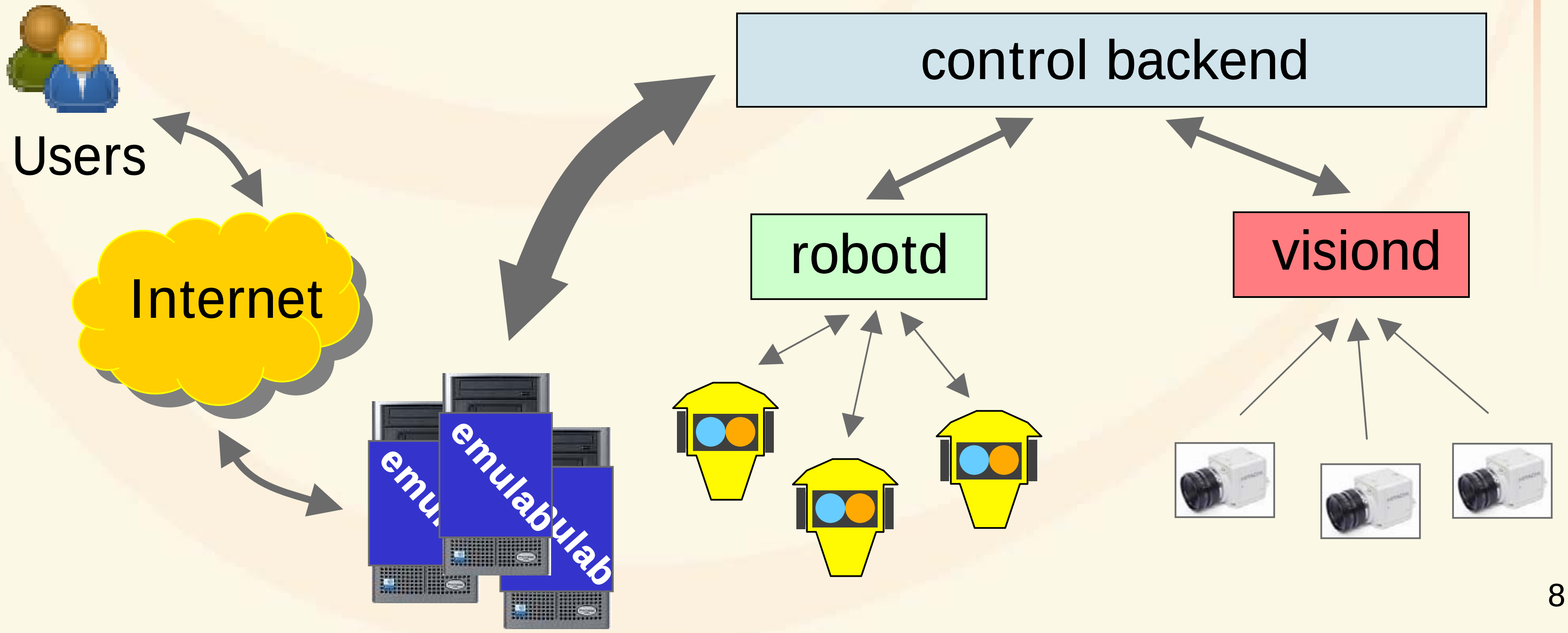
- Several user-controllable mobile robots
  - Onboard PDA, WiFi, and attached sensor mote
- Many fixed motes surround motion area
  - Simple mass reprogramming tool
  - Configurable packet logging
  - … and many other things
- New user interfaces
  - Web applet provides interactive motion control and monitoring
  - Other applets for monitoring robot details: battery, current motion execution, etc





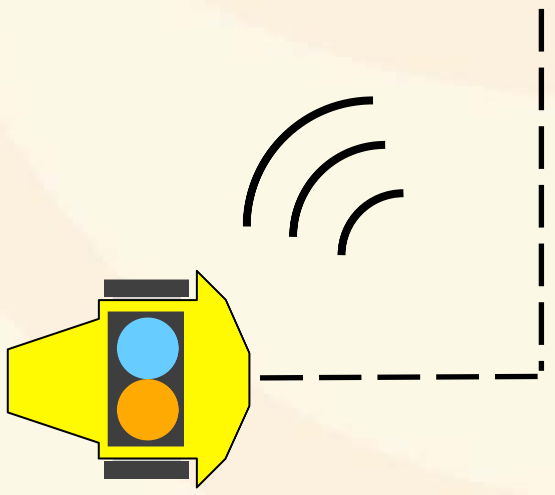
Mobile Testbed Architecture

- Emulab extensions
  - Remote users create mobile experiments, monitor motion
- Vision-based localization: *visiond*
  - Multi-camera tracking system locates robots
- Robot control: *robotd*
  - Plans paths, performs motion on behalf of user
  - Vision system feedback ensures precise positioning



Motion Interfaces

- Drag’n’drop Java applet, live webcams
- Command line
- Pre-script motion in NS experiment setup files
  - Use event system to script complex motion patterns and trigger application behavior



```
set seq [ $ns event-sequence {
    $myRobot setdest  1.0  0.0
    $program run -time 10
           “/proj/foo/bin/pkt_bcast”
     $myRobot setdest  1.0  1.0
    …
} ]
$seq run
```

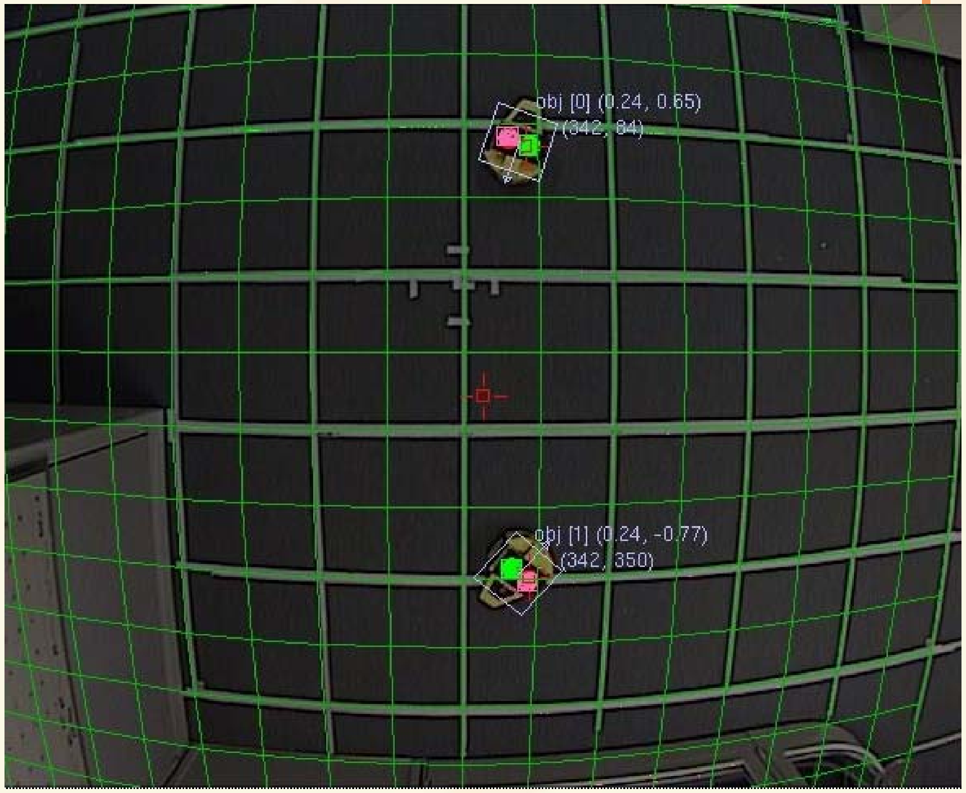
Outline

Key Problem #1: Robot Localization

- Need precise location of each robot
  - Needed for our control and for experimenter use in evaluation
- System must minimize interference with experiments
  - Excessive node CPU use
  - Wireless or sensor interference
- Solution: obtain from overhead video cameras with computer vision algorithms (*visiond*)
  - Limitation: requires overhead lighting

Localization Basics

- Several cameras, pointing straight down
  - Fitted with ultra wide angle lens
- Instance of Mezzanine (USC) per camera "finds" fiducial pairs atop robot
  - Removes barrel distortion ("dewarps")
- Reported positions aggregated into tracks
- But…



Localization: Better Dewarping

- Mezzanine's supplied dewarp algorithm unstable (10-20 cm error)
- Our algorithm uses simple camera geometry
  - Model barrel distortion using cosine function

$$loc_{world} = loc_{image} \;/\; \cos(\, a * w \,)$$

(where a is angle between optical axis and fiducial)

  - Added interpolative error correction
- Result: ~1cm max location error
- No need to account for more complex distortion, even for very cheap lenses

Outline

- Introduction
- Context & Architecture
- Key Problem #1: Localization
- **Key Problem #2: Robot Control**
- Evaluation
  - Microbenchmarks
  - Data-gathering experiment
- Summary

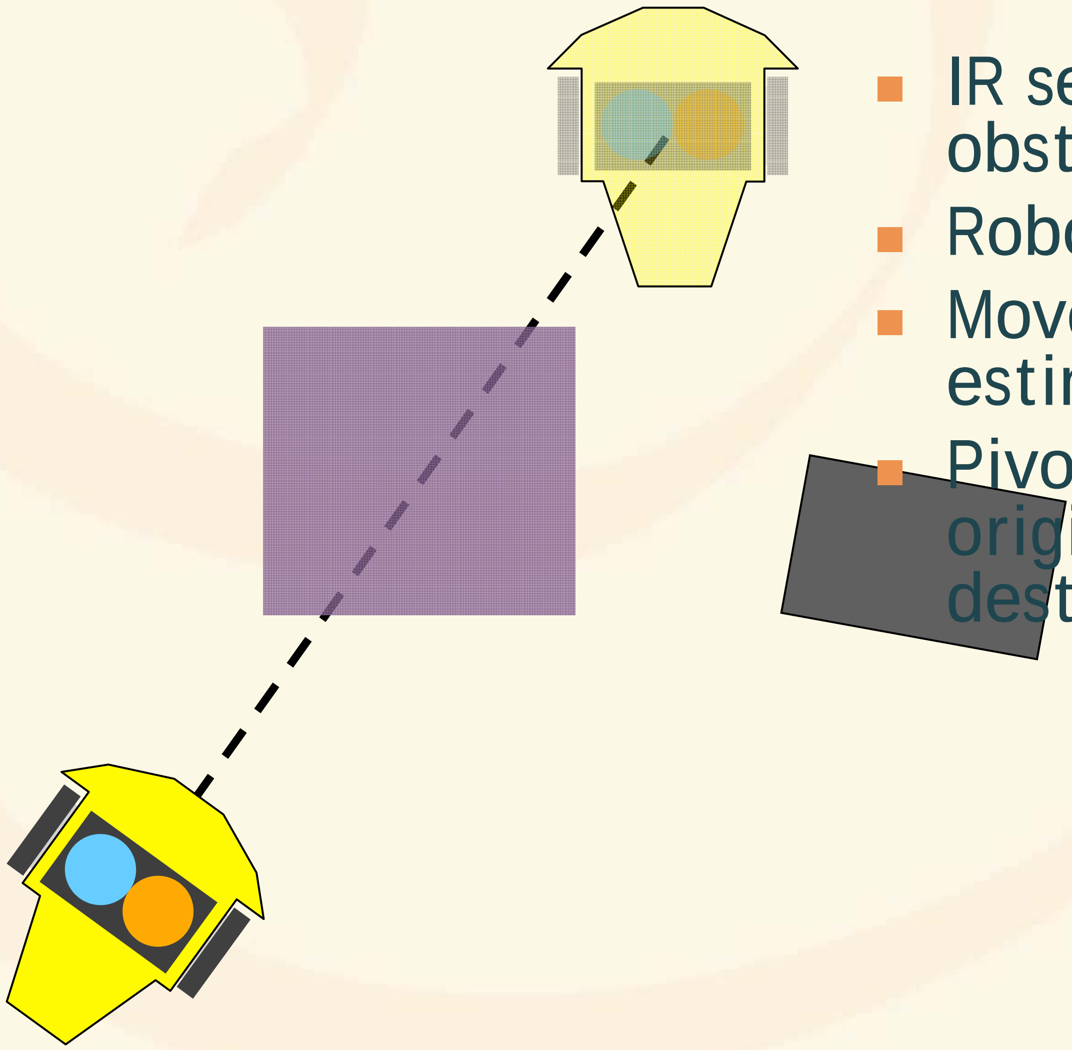
Key Problem #2: Robot Motion

- Users request high-level motion
  - Currently support waypoint motion model (A->B)
- *robotd* performs low-level motion:
  - Plans reasonable path to destination
  - Avoids static and dynamic obstacles
  - Ensures precise positioning through vision system feedback

Motion: Control & Obstacles

- Planned path split into segments, avoiding known, fixed obstacles
  - After executing each segment, vision system feedback forces a replan if robot has drifted from correct heading
- When robot nears destination, motion enters a refinement phase
  - Series of small movements that bring robot to the exact destination and heading (three sufficient for < 2cm error)
- IR rangefinders triggered when robot detects obstacle
  - Robot maneuvers around simple estimate of obstacle size

Motion: Control & Obstacles



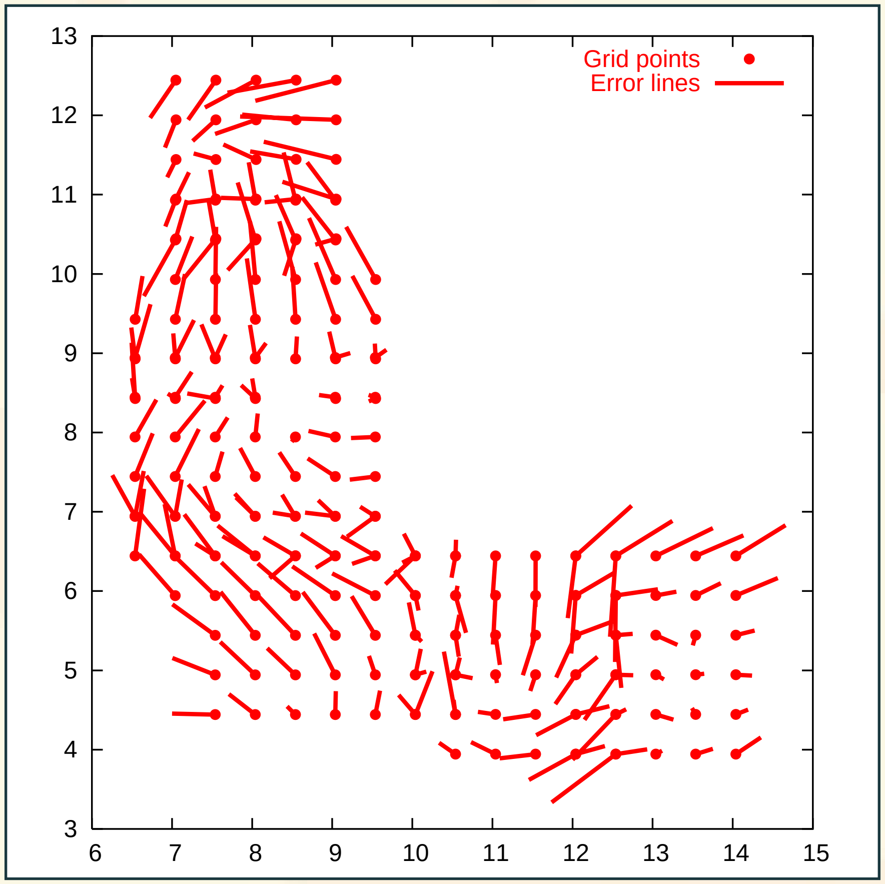
- IR sensors “see” obstacle
- Robot backs up
- Moves to corner of estimated obstacle
- Pivots and moves to original final destination

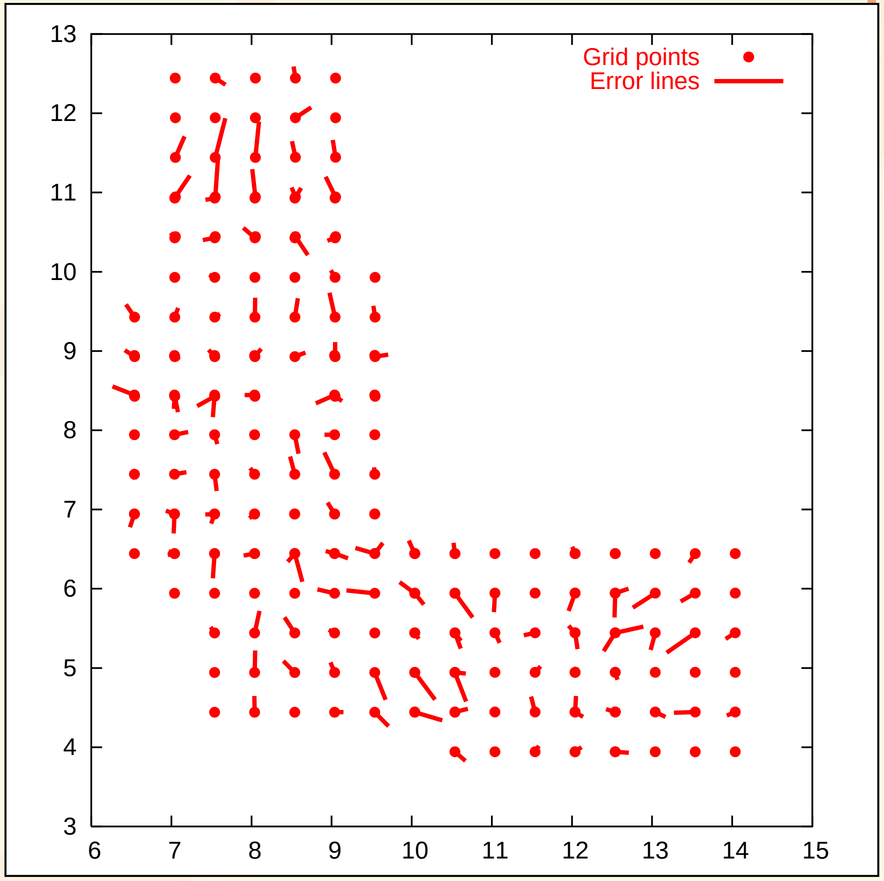
Outline

- Introduction
- Context & Architecture
- Key Problem #1: Localization
- Key Problem #2: Robot Control
- Evaluation
  - **Microbenchmarks**
  - Data-gathering experiment
- Summary

Evaluation: Localization

- With new dewarping algorithm and error correction, max error 1.02cm, mean 0.32cm



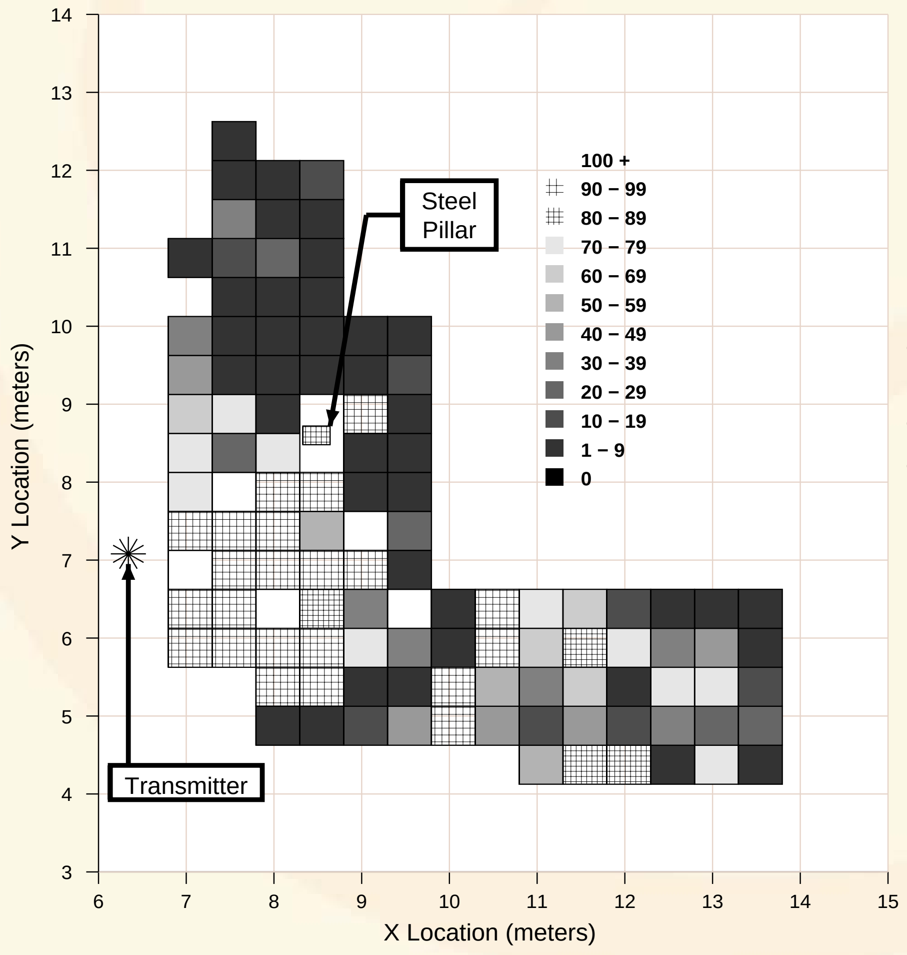


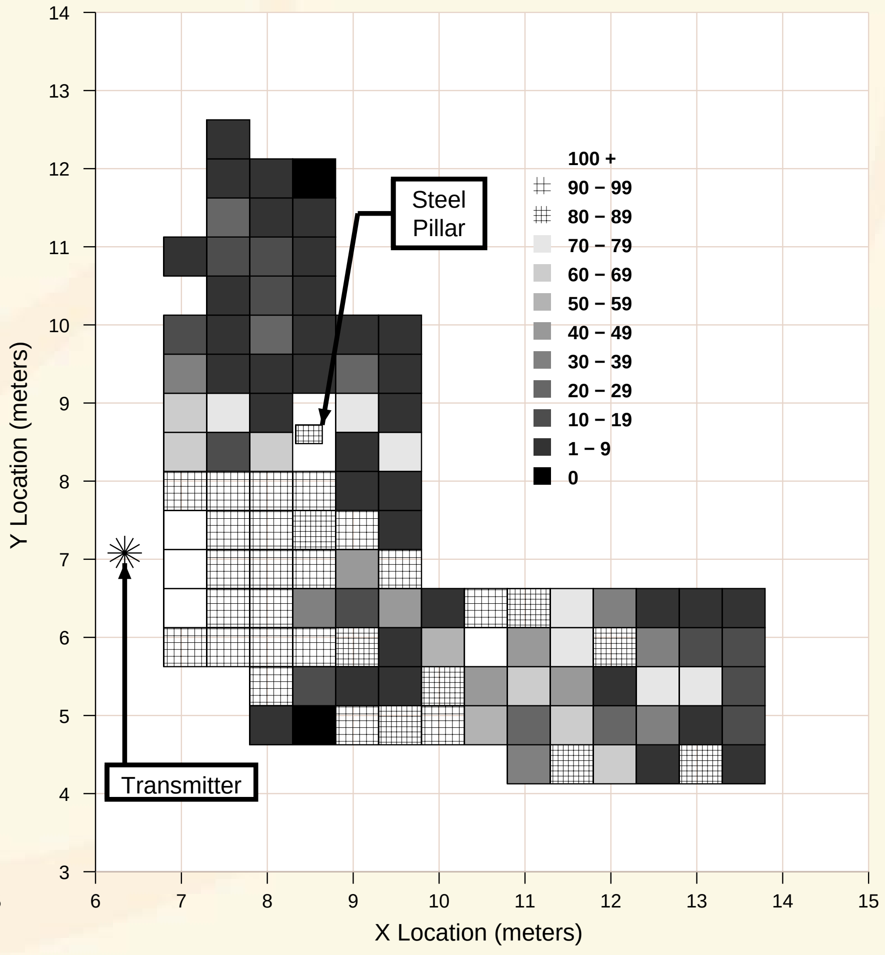
Case Study: Wireless Variability Measurements

- Goal: quantify radio irregularity in our environment
  - Single fixed sender broadcasts packets
  - Three robots traverse different sectors in parallel
  - Count received packets and RSSI over 10s period at each grid point
- Power levels reduced to demonstrate a realistic network

Wireless Variability (2)

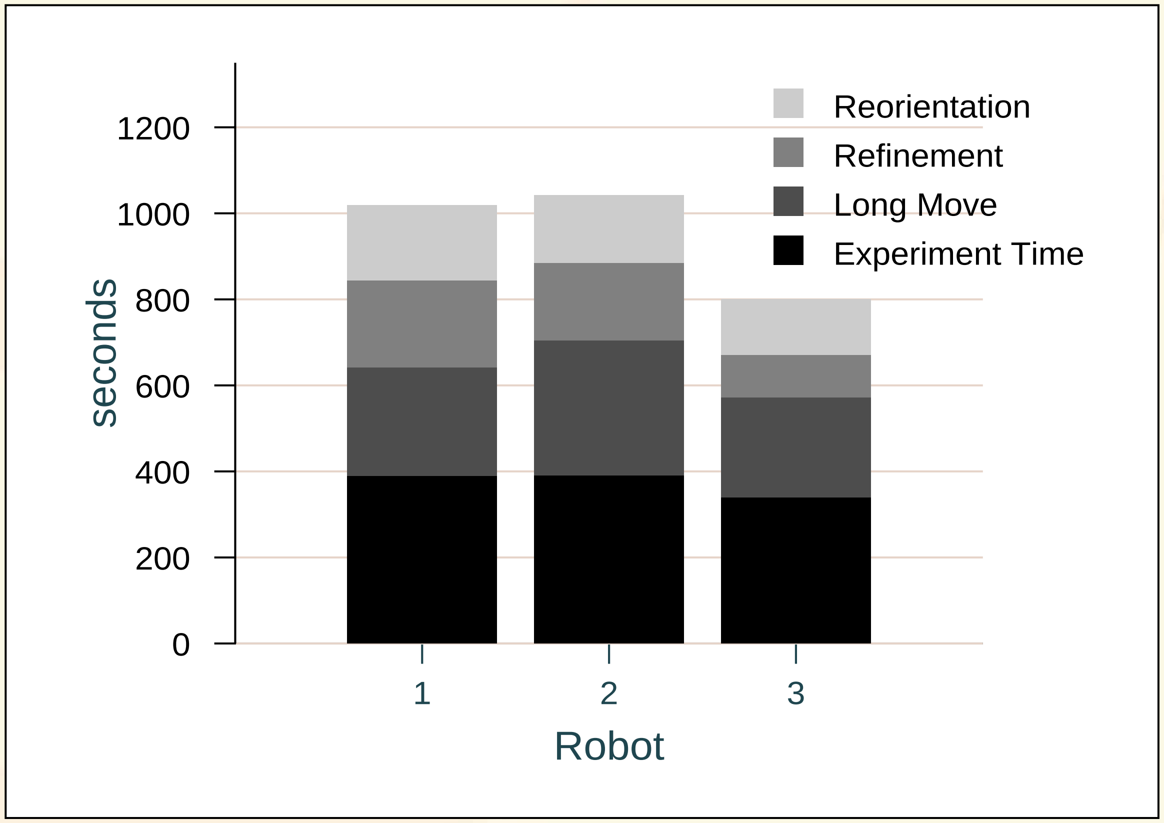
- Some reception decrease as range increases, but significant irregularity evident
- Similarity shows potential for repeatable experiments





Wireless Variability (3)

- 50-60% time spent moving robots
  - Continuous motion model will improve motion times by constantly adjusting robot heading via vision data



Outline

- Introduction
- Context & Architecture
- Key Problem #1: Localization
- Key Problem #2: Robot Control
- Evaluation
  - Microbenchmarks
  - Data-gathering experiment
- **Summary**

In Conclusion…

- Sensor net testbed for real, mobile wireless sensor experiments
- Solved problems of localization and mobile control
- Make real motion easy and efficient with remote access and interactive control
- Public and in production (for over a year!)
  - Real, useful system

Thank you!

Questions?

emulab

Related Work

- MiNT
  - Mobile nodes confined to limited area by tethers
- ORBIT
  - Large indoor 802.11 grid, emulated mobility
- Emstar
  - Sensor net emulator: real wireless devices coupled to mote apps running on PCs
- MoteLab
  - Building-scale static sensor mote testbed

Ongoing Work

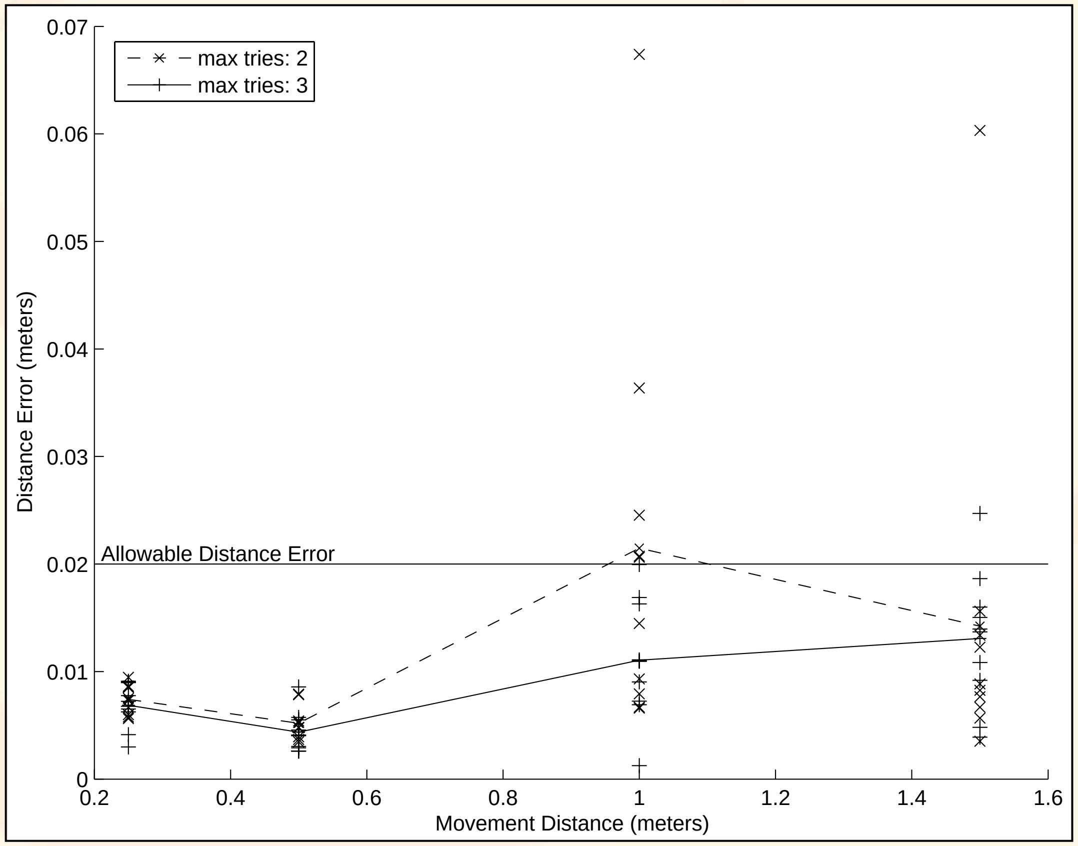
- Continuous motion model
  - Will allow much more efficient, expressive motion
- Sensor debugging aids
  - Packet logging (complete)
  - Sensed data emulation via injection (in progress)
- Interactive wireless link quality map (IP)

Evaluation: Localization

- Methodology:
  - Surveyed half-meter grid, accurate to 2mm
  - Placed fiducials at known positions and compared with vision estimates
- With new dewarp algorithm and error correction, max error 1.02cm, mean 0.32cm
  - Order of magnitude improvement over original algorithm

Evaluation: Robot Motion

- In refine stage, three retries sufficient
  - End position 1-2cm distance from requested position
- Accuracy of refine stage not affected by total movement distance



emulab