Exposing Resource Tradeoffs in Region-Based Communication Abstractions for Sensor Networks

Matt Welsh

Harvard University

Division of Engineering and Applied Sciences

mdw@eecs.harvard.edu

Wireless Sensor Networks

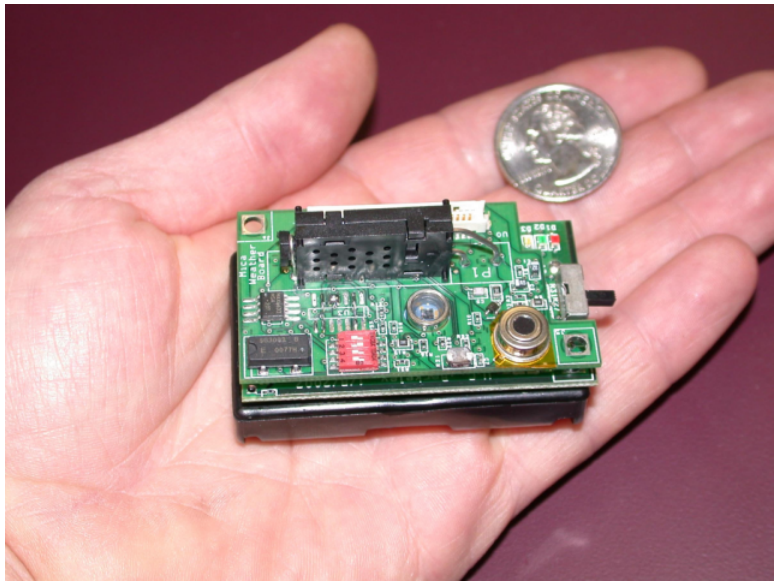
Exciting emerging domain of deeply networked systems

- Low-power, wireless “motes” with tiny amount of CPU/memory
- Large federated networks for high-resolution sensing of environment

Drive towards miniaturization and low power

- Ride Moore’s law to bring down cost and size
- Eventual goal - complete systems in 1 mm^3, MEMS sensors

Example: Berkeley MICA2 Mote



- ATMEGA 128L (8 MHz 8-bit CPU)
- 128KB code, 4 KB data SRAM
- 512 KB flash for logging
- 433/916 MHz 19.2 Kbps radio (100m)
- Sandwich-on sensor boards
- Powered by 2AA batteries

Typical Applications

Moving vehicle tracking and pursuit

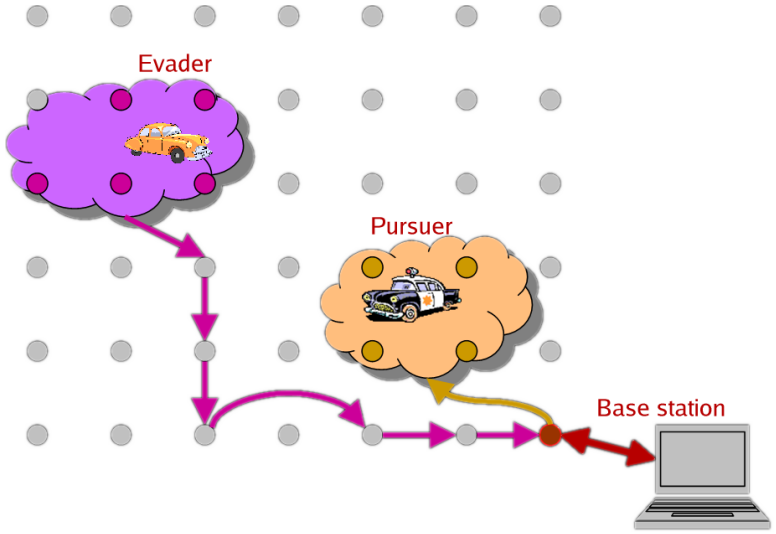
- Sensors take magentometer readings, locate object using centroid of readings
- Base station informs automated pursuer of object location

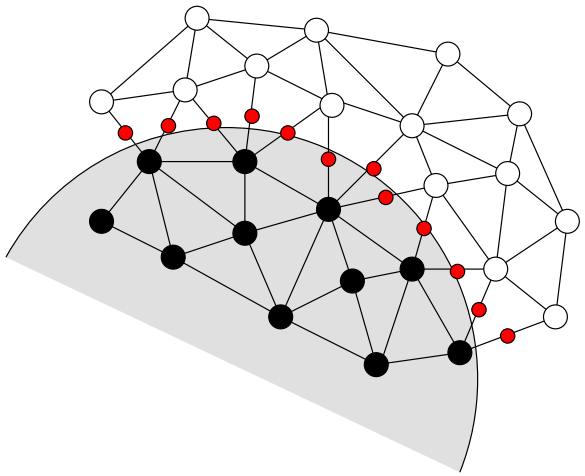
Great Duck Island - habitat monitoring

- Gather temp, humidity, IR readings from petrel nests
- Determine occupancy of nests to understand breeding/migration behavior

Spatial contour/region detection

- Detect frontier of phenomenon of interest (e.g., contaminant flow in groundwater)
- Sensors communicate locally to detect contour





Programming Challenges

Bandwidth and energy limitations force in-network processing

- Infeasible to send all data to central location for processing
- App designers must decompose program into complex distributed task

Spatial coordination within **local regions** of nodes

- Nodes wish to coordinate with local “neighbors”
- Coordinated detection of localized phenomena
- Aggregate sensor readings for bandwidth reduction

Inherent tradeoff between resource consumption and accuracy

- More messages $\rightarrow$ increased energy and bandwidth $\rightarrow$ greater precision
- Cannot consume arbitrary energy to achieve reliable communication
- Limited energy and bandwidth budget mandates statistical design
- Applications must deal with lossy communication, imperfect results

Our Goals

Develop a programming model for **aggregate programs** across an entire sensor network

- Current programming models are node centric and low level
- Scientists don't want to think about gronky details of radios, timers, battery life, etc.
- Like implementing Linux by toggling switches on a PDP-11

Flexible communication primitives for sensor networks

- Reduce programming effort to construct applications
- Abstract low-level details of local coordination
- Focus on spatial computation within local neighborhoods
- Neighborhood maintenance, routing, and collective communication

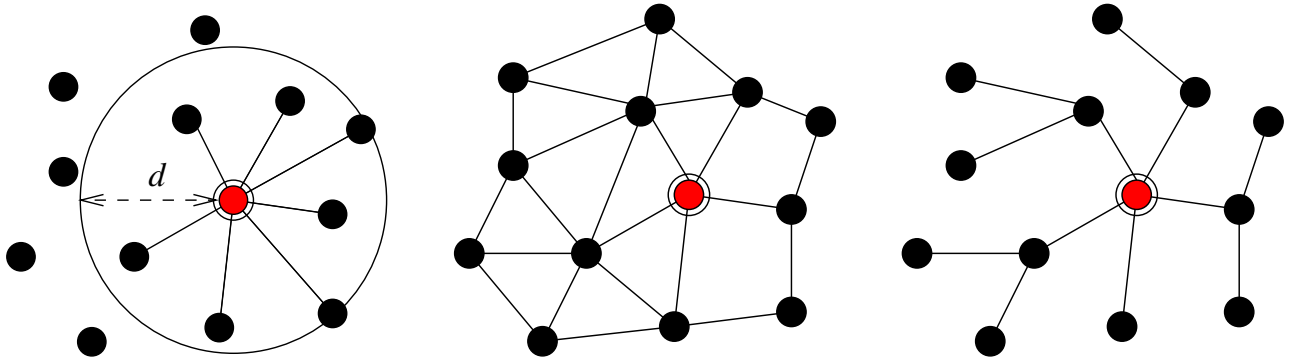
Allow application to tune resource/accuracy tradeoff

- Application must have control over resource usage
- Don't hide settings of complex parameters inside lower layer
- Provide feedback to applications:
  - *Timeouts on communication operations*
  - *Accuracy and completeness of collective operations*
- Feedback used to adapt to changing network conditions

Abstract Regions

A *region* is a group of nodes with some geographic or topological relationship

- e.g., All nodes within distance d from node k
- Neighbors forming a planar mesh based on radio connectivity
- Spanning tree rooted at node k



Shared variables support coordination

- Tuple-space like programming model within regions
- Implementation may broadcast, pull requested data, or gossip

Reductions support aggregation of shared variables

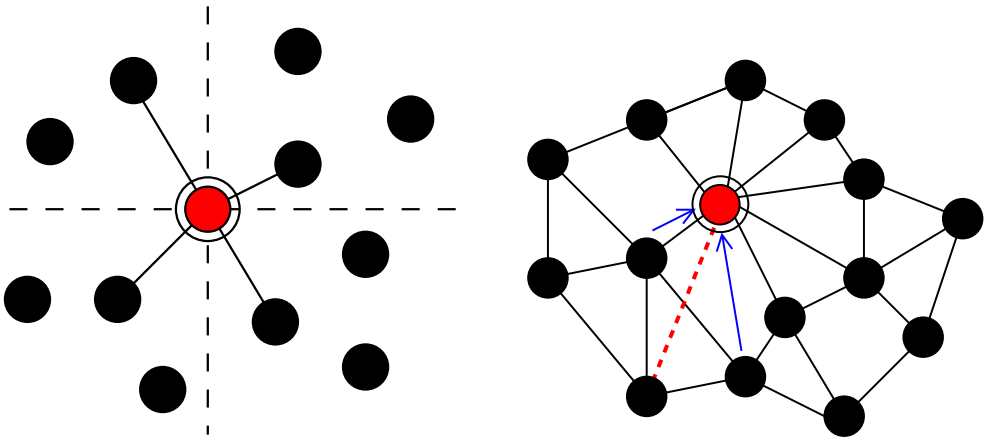
- Combine shared variables in region to a single value

Region Implementations

Radio and geographic neighborhoods

- k nearest neighbors, all nodes within k hops, etc.
- Nodes emit periodic beacons with node ID and (optionally) location
- Filter received beacons to determine neighbors (e.g., k nearest nodes)

Approximate planar mesh



- Goal: Constuct connectivity mesh with minimal number of crossed edges
- Based on pruned Yao graph: set of m nodes in distinct sectors
- Advertise chosen Yao neighbors and prune crossing edges from set

Spanning tree

- Useful for data collection and aggregation to a sink (e.g., TinyDB, directed diffusion)

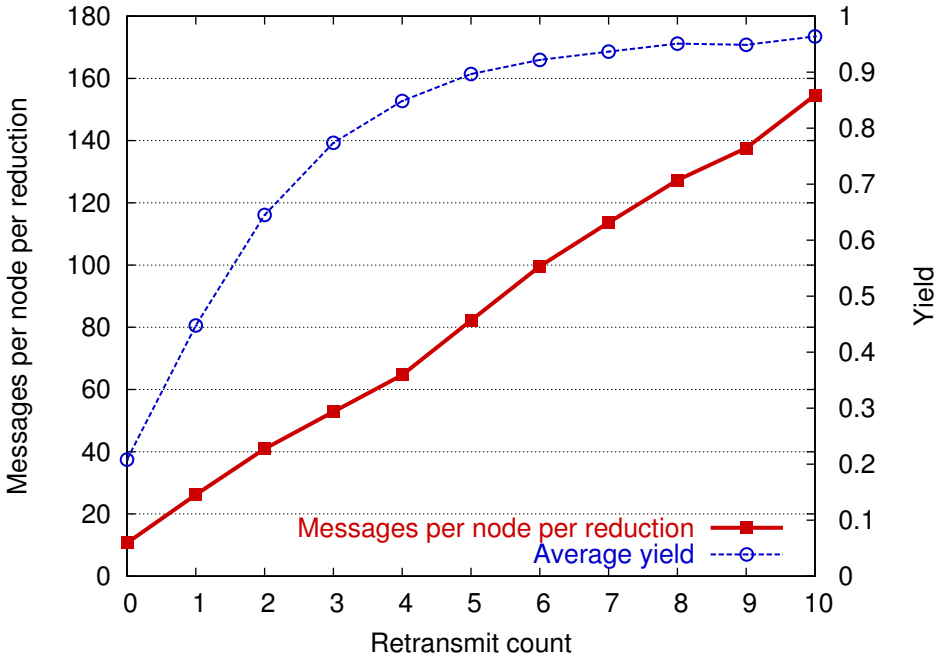
Quality feedback and tuning

Region operations are inherently statistical

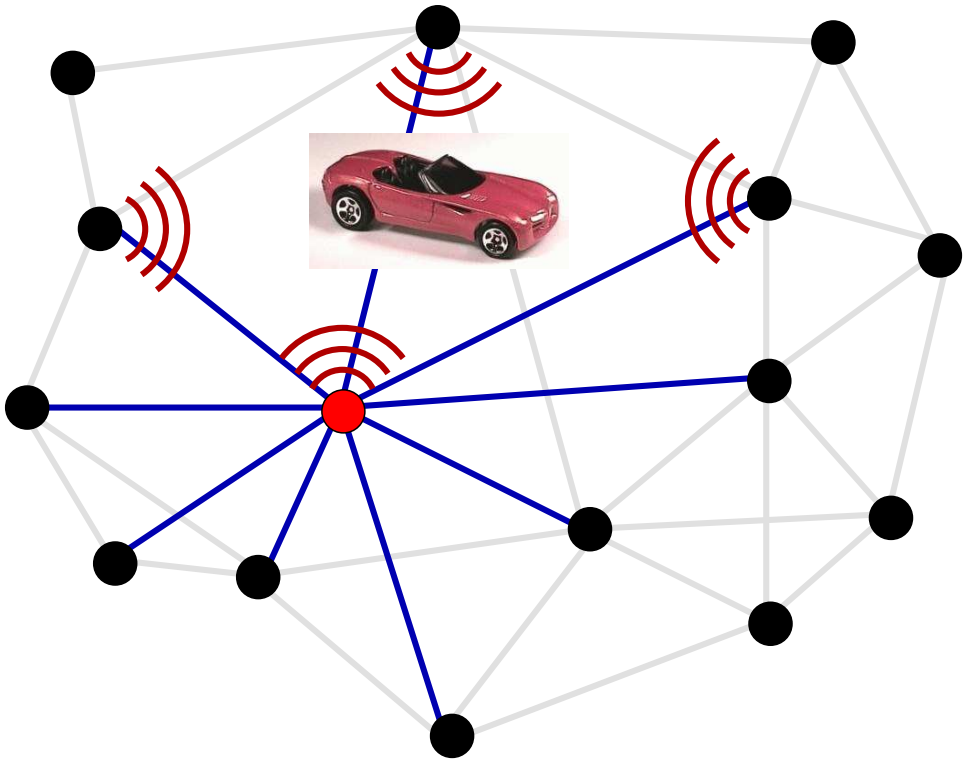
- Shared variable operations may time out
- Reductions may only contact subset of nodes
- Collective operations report **yield**: fraction of nodes that responded to a request

Regions provide control over overhead-accuracy tradeoff

- Programmer can tune parameters affecting resource usage of region operations
- Examples: retransmission count, timeouts, number of neighbors in region
- Quality feedback can be used to drive adaptation to changing network conditions



Object tracking using regions



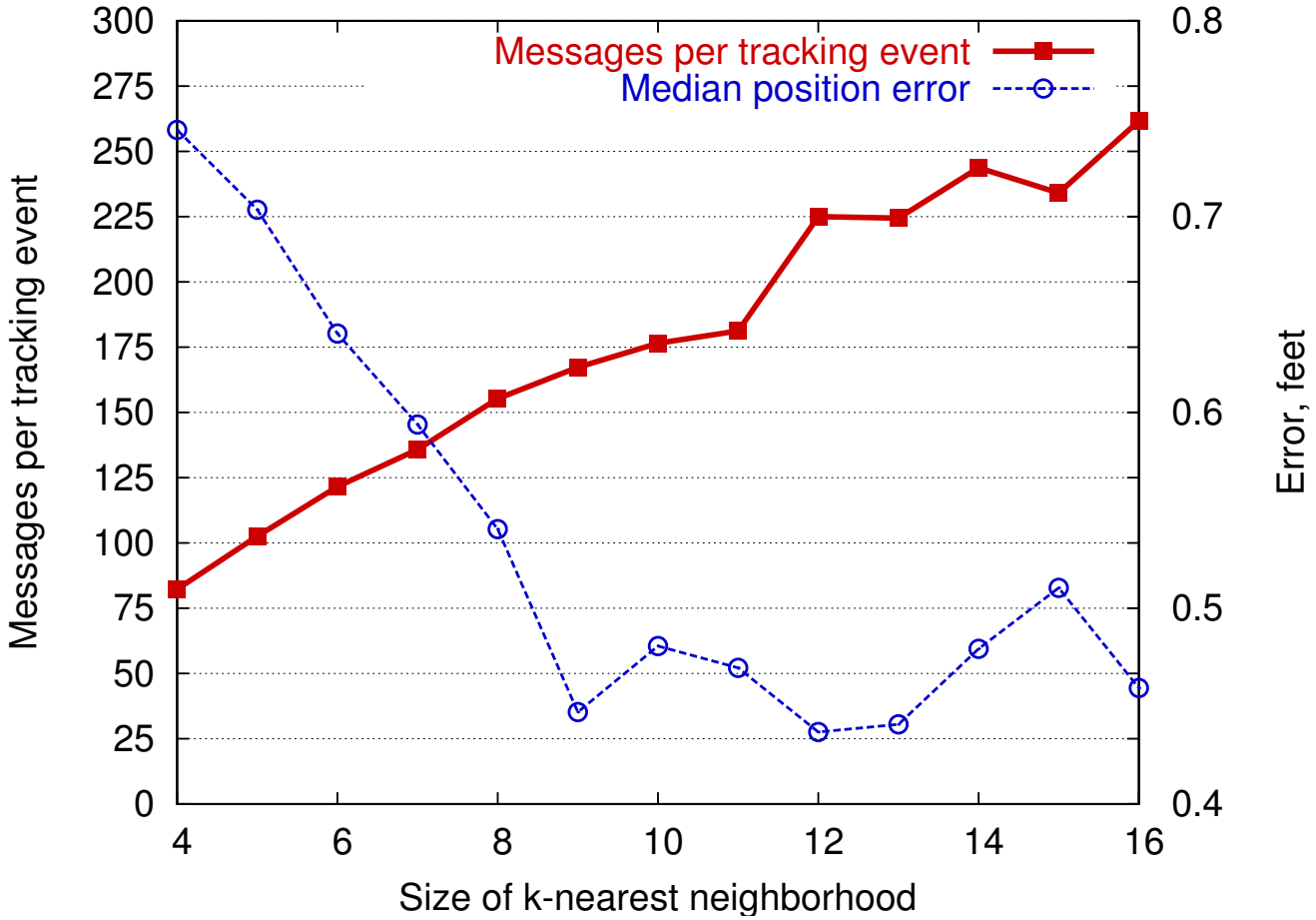
Localize vehicle in a sensor field using magnetometer readings

- Nodes store local sensor reading as **shared variable**
- **Reduction** used to determine node with the largest value
- Max node performs **reductions** to determine centroid of sensor readings

Abstract Regions hide details of communication and coordination

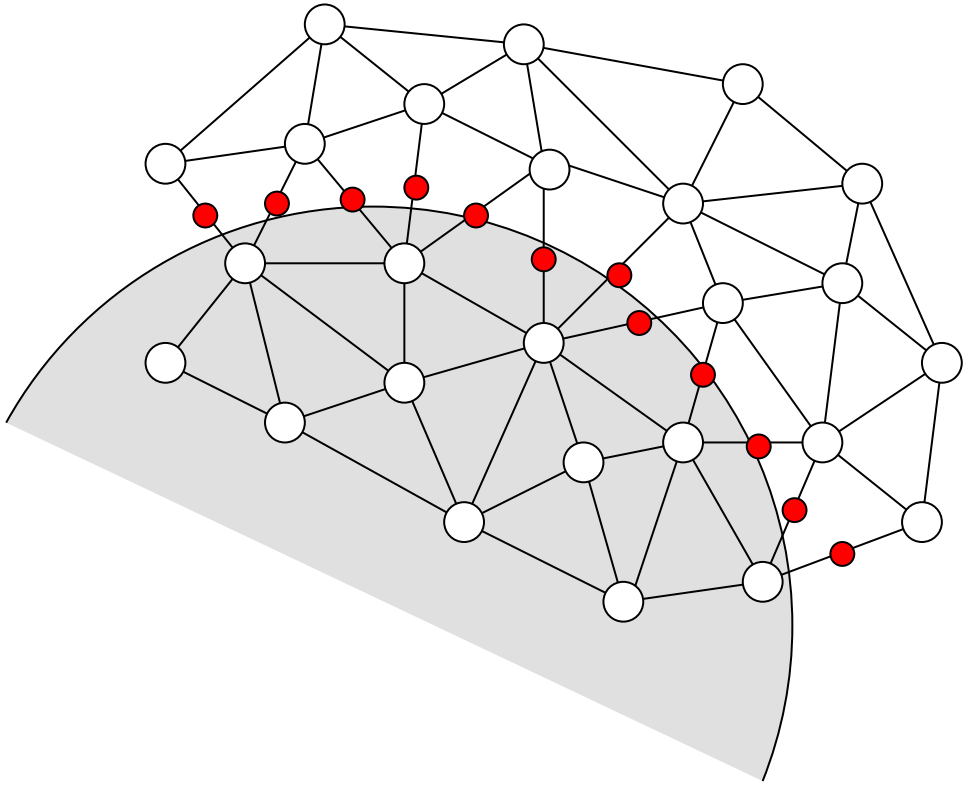
- Resulting application code is very concise

Object tracking accuracy and overhead



- TOSSIM sensor network simulator with realistic radio model
- Object moving in circular path through sensor net
- Tuning knob: Number of neighbors in k-nearest neighbor region
- Size of neighborhood increases both accuracy and message overhead

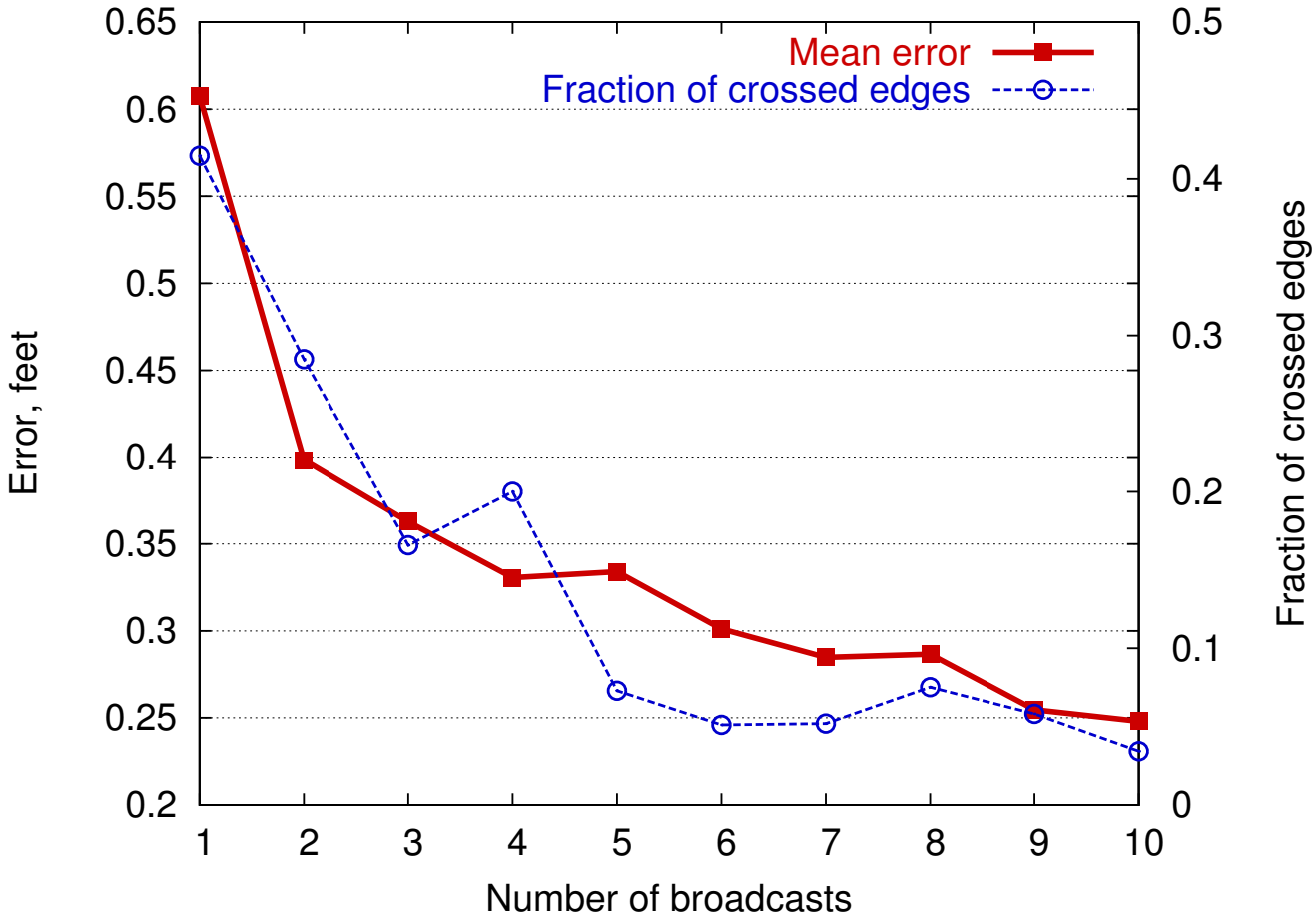
Contour finding



Determine location of threshold between sensor readings

- Construct approximate planar mesh of nodes
- Nodes above threshold compare values with neighbors
- Contour defined as midpoints of edges crossing threshold

Contour detection accuracy and overhead



Contour finding accuracy a function of node advertisements

- Form approximate planar mesh region
- More advertisements $\longrightarrow$ fewer crossed edges
- Mean error directly correllated with mesh quality

Related Work

Various communication/programming models for sensor networks

- Directed Diffusion: event detection and propagation
- GHT: Distributed hashtable-like storage within sensor network
- Others (DIFS, SPIN, DIMENSIONS, etc.) mostly focused on specific app needs
- Little work on exposing resource tradeoffs to applications

Sensor network query systems [TinyDB, Cougar]

- SQL-like queries flooded to sensor network
- Queries periodically sample and relay aggregate data to base station
- Not possible to express general-purpose local behavior

Abstract regions are intended to be lower-level and more general

- Underlying substrate for local communication and coordination
- Can be used to implement higher-level abstractions (e.g., TinyDB)
- Simplify application code considerably
- Expose resource consumption/accuracy tradeoff

Future Work

Ongoing work on abstract regions

- Generic region constructors
  - ▷ *Supply membership primitive and communication strategy*
- Feedback control for tuning resource usage for desired loss level or energy budget
- Programming language design based on region primitives

Spatial operations over “virtual” coordinate spaces

- Define overlay set in space, e.g., grid, disc-neighborhood, Voronoi diagrams
- Allow query to arbitrary point in that space
- E.g., “sensor reading at $(30.5, 42.6)$”
- Translates into interpolation across nearby sensor values

Temporal operations and aggregation

- Triggers and event-driven operations
- Sample and aggregate over time
- Specify timeouts, periodic execution, etc.

Conclusion

How do you program a entire network of distributed, volatile, resource-limited sensors?

- Program “the network” rather than individual nodes
- Requires appropriate programming models and communication primitives

Spatial programming and communication using abstract regions

- Communication and aggregation within local regions
- **Region formation** maintains neighborhood set
- **Shared variables** provide simple data sharing
- **Reductions** provide data aggregation

Exposing the resource-accuracy tradeoff to applications is crucial

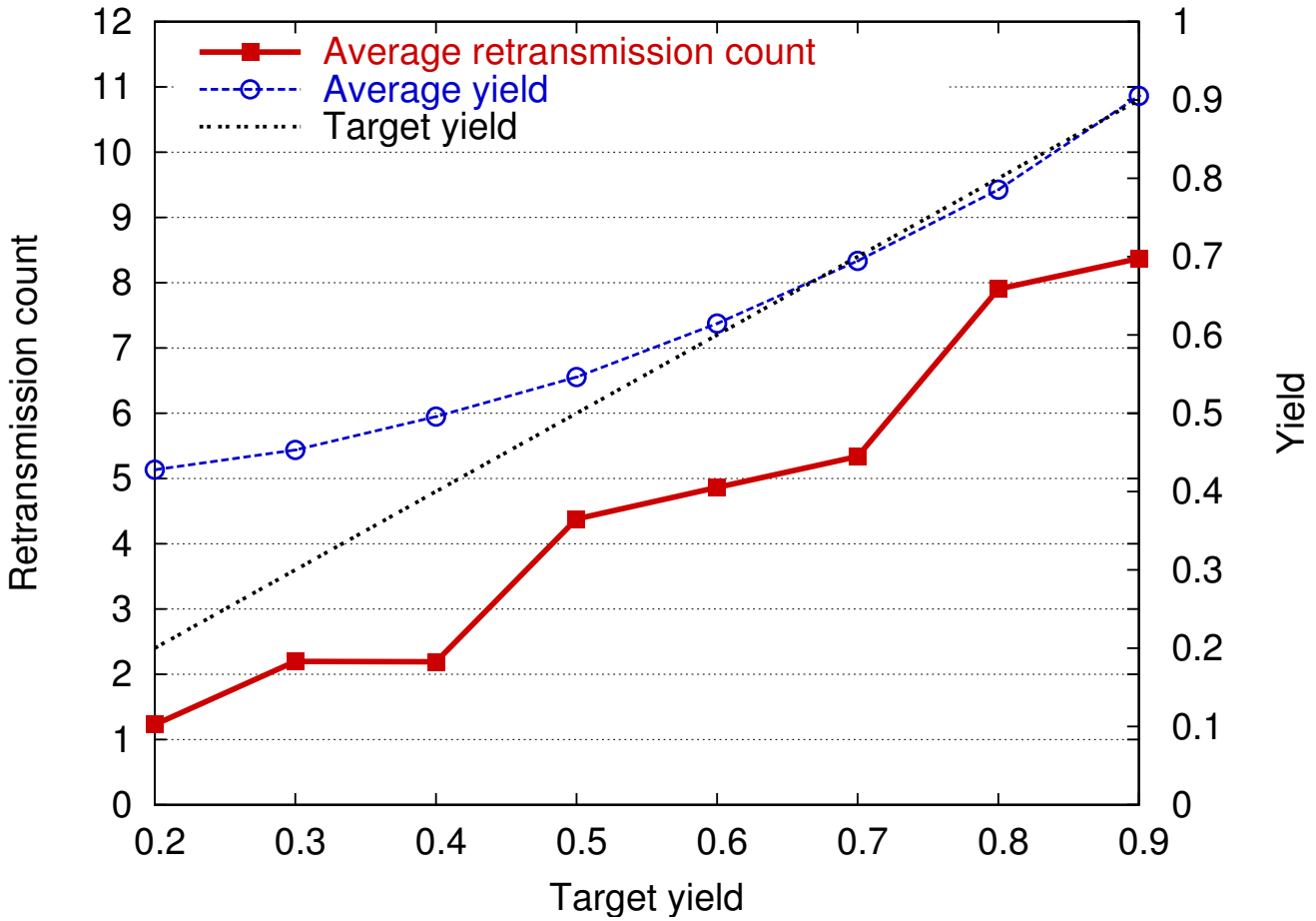
- Sensor network communication is inherently statistical
- Applications must adapt to changing network conditions
- Abstract region operations provide accuracy feedback and tuning knobs

For more information:

`http://www.eecs.harvard.edu/~mdw/proj/mp`

Backup Slides Follow

Adaptive Reduction



Tune message retransmission count to meet yield target

- Take EWMA of yield, adjust retransmission count using simple AIAD controller
- Very accurate for yield targets above 0.5
- For low targets, low retransmission count often achieves better than desired results

Macroprogramming

How do you program a system composed of a large number of distributed, volatile, error-prone systems?

- Initial focus is on sensor networks
- Approach applies to many other domains:
- Distributed systems, protocol design, and P2P to name a few

Developing a high-level language to express **aggregate programs** across an entire field of motes

- Examples: contour finding, object tracking, distributed control
- TinyDB [Madden et al.] is one step in this direction

Current programing models are **node centric**

- NesC focuses entirely on individual nodes, rather than the aggregate
- Want to program the “whole system”

Current programing models are **too low-level**

- Scientists don’t want to think about gronky details of radios, timers, battery life, etc.
- Like writing Linux by toggling switches on a PDP-11
- Evidence: Huge engineering effort for each demo

Macroprogramming goals

Develop a set of communication and coordination primitives

- Goals somewhat akin to MPI
- Abstract the details of underlying communication
- Provide enough structure to permit optimizations

Expose these primitives in a global, high-level language

- Simple syntactic structures for performing spatiotemporal aggregation
- Automatically compile down to low-level behavior of individual nodes
- Compiler-directed optimization for energy and bandwidth usage

Expose the tradeoff between resource usage and accuracy

- Support “lossy programming”
- Programmer can tune resource usage of the communication layer
- Each collective operation reports its **yield**