Sensor Networks

- Potential Applications
- Examples of Nodes
- Issues
- Protocols
- Early Ideas
- Summary
Potential Applications

- Micro-sensors, onboard processing, and wireless interfaces all feasible at very small scale
  - can monitor phenomena “up close”
- Will enable spatially and temporally dense environmental monitoring
- Embedded Networked Sensing will reveal previously unobservable phenomena
Potential Applications

Slide from D. Estrin

Transportation and Urban Monitoring

Smart Kindergarten

Disaster Response

High-speed Wireless LAN (WLAN)
Node Examples

Rockwell Hidra
3.5”x3.5”x3”
StrongARM 1100 processor @ 133 MHz
Connexant’s RDSSS9M Radio @ 100 kbps
1-100 mW, 40 channels
Various sensors
http://wins.rsc.rockwell.com/

Berkeley Motes
Atmel microcontroller
temperature, light, humidity,
pressure, 3 axis magnetometers,
3 axis accelerometers
10kbps, 20m
Node Examples

Slide from Mani Srivastava

**UCLA iBadge**
Wearable Sensor Badge
- acoustic in/out + DSP
- temperature, pressure, humidity,
- magnetometer, accelerometer
- ultrasound localization
- orientation via magnetometer and
  accelerometer
- bluetooth radio

Sylph Middleware

**UCLA Medusa Localizer Node**
- 40MHz ARM THUMB
  - 1MB FLASH, 136KB RAM
  - 0.9MIPS/MHz  480MIPS/W
- RS-485 bus
  - Out of band data collection,
    formation of arrays
- 540mAh Rechargeable Li-Ion battery
Node Examples

BWRC’s *PicoNode* TripWire Sensor Node

Slide from Jan Rabaey

**Version 1: Light Powered**

Size determined by power dissipation (1 mW avg)

**Version 2: Vibration Powered**
Node Examples

IPaq Sensor Node

WaveLan Card
- IEEE 802.11b Compliant
- 11 Mbit/s Data Rate

HM2300 Magnetic Sensor
- uC Based with RS232
- Range of +/- 2Gausus
- Adjustable Sampling Rate
- X, Y, Z output
- Device ID Management

Familiar v0.5
- Linux Based OS for iPAQ H3600s
- JFFS2, read/write iPAQ’s flush
- Tcl ported

iPAQ 3670
- Intel StrongARM
- Power Management (normal, idle & sleep mode)
- Programmable System Clock
- IR, USB, Serial (RS232)
- Transmission

Acoustic Sensor & Actuator
- Built-in microphone
- Built-in speaker

Slide from Mani Srivastava
Issues

- Low Power → special protocols
  - Ex.: Wake up; periodic sleep
- Small Memory → specialized code
  - Ex.: TinyOS, Sylph, …
- Localization
  - GPS, triangulation, …
- Synchronization
  - GPS, NTP, …
- Addressing & Naming
- Routing
  - Broadcast, multicast, anycast, mobility, …
- Transport
  - Intermittent connectivity; noisy links; error tolerance
- Observation
  - Protocols should be application specific, not generic
## Protocols

<table>
<thead>
<tr>
<th>Resource constraints call for more tightly integrated layers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Open Question:</strong></td>
</tr>
<tr>
<td>Can we define an Internet-like architecture for such application-specific systems??</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Queries, External Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>In-network: Application processing, Data aggregation, Query processing</td>
</tr>
<tr>
<td>Data dissemination, storage, caching</td>
</tr>
<tr>
<td>Adaptive topology, Geo-Routing</td>
</tr>
<tr>
<td>MAC, Time, Location</td>
</tr>
<tr>
<td>Phy: comm, sensing, actuation, SP</td>
</tr>
</tbody>
</table>

Slide from D. Estrin
### Systems Taxonomy

- **Spatial and Temporal Scale**
  - Extent
  - Spatial Density (of sensors relative to stimulus)
  - Data rate of stimuli
- **Variability**
  - Ad hoc vs. engineered system structure
  - System task variability
  - Mobility (variability in space)
- **Autonomy**
  - Multiple sensor modalities
  - Computational model complexity
- **Resource constraints**
  - Energy, BW
  - Storage, Computation

### Load/Event Models

- **Frequency**
  - Spatial and temporal density of events
- **Locality**
  - Spatial, temporal correlation
- **Mobility**
  - Rate and pattern

### Metrics

- **Efficiency**
  - System lifetime/System resources
- **Resolution/Fidelity**
  - Detection, Identification
- **Latency**
  - Response time
- **Robustness**
  - Vulnerability to node failure and environmental dynamics
- **Scalability**
  - Over space and time
Early Ideas

- **Routing**
- **Finding**
- **Energy**
- **Location**

Most slides from D. Estrin
Routing

Special Features in Sensor Networks

- Event and Query
- Find data somewhere in network
- Broadcast a request
- Collect data from all nodes

Examples of routing algorithms

- Grab, Gossip, Ant, Directed Diffusion, Data Centric, Rumor
GRAdient Broadcast (GRAB)

- Builds a cost field toward a particular node, then reliably routing queries across a limited size mesh toward that node
- Overhead of network flood
- Queries route along short paths
- Delivered cheaply and reliably
- Not designed specifically to support in-network processing
Gossip Routing

- Nodes flood by sending message to some of neighbors
- By the redundancy in the link, most nodes receive the flooded packed
- Used to deliver query or flood events
- Less overhead than conventional flooding
- Not be designed specifically for energy constrained contexts
Ant Algorithm

- Agent traverses the network encoding the quality of the path they have traveled, and leave the encoded path as state in the nodes
- At every node, an agent picks its next hop probabilistically, biased toward already know good paths
- Very effective in dealing with failure, because always some amount of exploration
Directed Diffusion and Geo-Routing

- Provides a mechanism for doing a limited flooding of a query toward the event
- Set reverse gradients to send data back along the best route
- Results in high quality paths
- But requires an initial flooding of the query for exploration
Data-Centric Storage in Sensornets

- Allows access to named data by hashing the name to a geographic region in the network
- Used to efficiently deliver queries to named events by storing the location of the events
- Relies on a global coordinate system
Rumor Routing

- Create paths leading to each event
- Event flooding creates a network-wide gradient field
- Query sent random walk until find the event path
- No flooding event across the network
- Query discovers event path, then route directly to the event
- If path cannot be found, application re-submitting the query, flooding it
Finding

- Find all sensors where \textit{temperature} > 82 degrees
- Find \textit{average temperature} on first floor of Cory
- Find \textit{maximum temperature} in Cory
- Find all sensors where \textit{temperature} > \textit{mean} + 2\textit{STDV}

Requires

- Query language
- In-network processing
- Routing algorithm
- Selection of sampling rate and granularity
  (may not need info from all nodes, depending on required accuracy)
Energy Efficiency MAC

- Energy Consumers
- Waste of Energy
- Listen and Sleep
- Collision Avoidance
- Adaptive Topology
Identifying the Energy Consumers

Need to shutdown the radio

From Tsiatis et al. 2002

\[ E_{TX} \approx E_{RX} \approx E_{IDLE} \gg E_{SLEEP} \]
Waste of Energy

• Major sources of energy waste
  – Idle listening
    • Long idle time when no sensing event happens
    • Collisions
    • Control overhead
    • Overhearing

• Try to reduce energy consumption from all above sources

• TDMA requires slot allocation and time synchronization

• Combine benefits of TDMA + contention protocols

Common to all wireless networks
Periodic Listen and Sleep

- **Problem:** Idle listening consumes significant energy
  - Nodes do not sleep in IEEE 802.11 ad hoc mode

- **Solution:** Periodic listen and sleep
  - Turn off radio when sleeping
  - Reduce duty cycle to ~10% (200 ms on/2s off)
  - Increased latency for reduced energy
Periodic Listen and Sleep

- Schedule maintenance
  - Remember neighbors’ schedules — to know when to send to them
  - Each node broadcasts its schedule every few periods
  - Refresh on neighbor’s schedule when receiving an update
  - Schedule packets also serve as beacons for new nodes to join a neighborhood
Collision Avoidance

- **Problem:** Multiple senders want to talk
- **Options:** Contention vs. TDMA
- **Solution:** Similar to IEEE 802.11 ad hoc mode (DCF)
  - Physical and virtual carrier sense
  - Randomized backoff time
  - RTS/CTS for hidden terminal problem
  - RTS/CTS/DATA/ACK sequence
Overhearing Avoidance

- Problem: Receive packets destined to others
- Solution: Sleep when neighbors talk
  - Basic idea from PAMAS (Singh 1998)
  - But we only use in-channel signaling
- Who should sleep?
  - All immediate neighbors of sender and receiver
  - How long to sleep?
    - The *duration* field in each packet informs other nodes the sleep interval
Adaptive Topology

- Can we do more than shut down radio in between transmissions/receptions?
- Can we put nodes to sleep for longer periods of time?

Goal:
- Exploit high density (over) deployment to extend system lifetime
- Provide topology that adapts to the application needs
- Self-configuring system that adapts to environment without manual configuration
Adaptive Topology: Tradeoff

- How many nodes to activate?
  - few active nodes:
    - distance between neighboring nodes high -> increase packet loss and higher transmit power and reduced spatial reuse;
    - need to maintain sensing coverage (see earlier session on coverage/exposure)
  - too many active nodes:
    - at best, expending unnecessary energy;
    - at worst nodes may interfere with one another by congesting the channel.
Adaptive Topology: Example – ASCENT

- The nodes can be in *active* or *passive* state.
  - Active nodes forward data packets (using routing mechanism that runs over topology).
  - Passive nodes *do not* forward any packets but may sleep or collect network measurements.
- Each node *joins* network topology or *sleeps* according to measured number of neighbors and packet loss, as *measured locally*. 

(a) Communication Hole  
(b) Self-configuration transition  
(c) Final State
Location

- Introduction
- Centralized LP
- Rectangular Intersection
- Network Coordinate
- DV-Hop
- Startup and Refinement
- Comparison

Slides from Jana van Greunen (UCB)
Introduction

Some applications require that the nodes in the network are aware of their geographic location.

- Too expensive to use GPS on every node
- Thus need algorithms to compute each node’s position using node-to-node range measurements and information from a few reference nodes
- Range measurements are made between each node and its neighbors within a circular area
Introduction cont’

- Measurements are made using TOA or RSSI – both methods are error prone

- A good localization algorithm should:
  - Be tolerant to range errors
  - Scale with network
  - Minimize communication & computation energy spent
  - Converge rapidly and accurately
  - Perform well across network topologies
  - Provide a measure of error
Centralized LP

- Key assumption: if two nodes can communicate with each other, they must lie within the communication radius $R$ of each other.
- Mathematically = 2-norm constraint on the node positions
- Find the positions that achieve the observed connectivity with the minimum total power
Rectangular intersection (Simić)

- Partition space into cubes/squares (cells)
- Communication area is a square (#cells)

At every unknown node:

**Step A:** Gather positions of one-hop neighbors with known positions

**Step B:** Compute estimated position via minimum rectangular intersection
Network Coordinate (Capkun)

- Each node $j$ measures distances to its one-hop neighbors and their distances from each other
- Place one-hop neighbors in local coordinate system
- Align in global coordinates

![Diagram](image-url)
DV-Hop (Niculescu et al.)

- Use ‘average’ distance to prevent error propagation
- Known nodes flood ‘hops’ and position through network
- Each unknown node stores the position and hop-distance (# hops*average distance) from all the known nodes
- When an unknown node has its hop-distance from more than three non-collinear known nodes it can compute its position via triangulation (solve $Ax=b$)
- This algorithm works well when topology is regular
Start-up & refinement (Savarese)

- **Startup: Initial position estimate**
  - DV-Hop

- **Refinement**
  - Nodes try to improve their position estimates by iteratively measuring the distances to one-hop neighbors and then performing weighted maximum likelihood triangulation.
  - All unknown nodes start with a weight of 0.1
  - Known nodes have weight 1.0
  - After each position update the weight of the node is set to the average weight of all its neighbors
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Centralized LP</th>
<th>Rectangular intersection</th>
<th>Network Coordinate</th>
<th>DV-Hop</th>
<th>Start-up &amp; refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Scalable</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Energy efficient</strong></td>
<td>No</td>
<td>Yes</td>
<td>Moderately</td>
<td>Yes</td>
<td>Moderately</td>
</tr>
<tr>
<td><strong>Accuracy</strong></td>
<td>Good</td>
<td>Spotty</td>
<td>Poor</td>
<td>Medium</td>
<td>Fair</td>
</tr>
<tr>
<td><strong>Speed of convergence</strong></td>
<td>Depends on network size</td>
<td>Fast</td>
<td>Depends on network size</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td><strong>Tolerant to range error</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Moderately</td>
</tr>
</tbody>
</table>

Note: No algorithm provides a measure of final position error
Summary

- New potential applications
- Many cute problems
- Next big thing … or niche technology?