Introduction to Sensor Networks
Introduction

• A large number of low-cost, low-power, multifunctional, and small sensor nodes

• Sensor nodes consist of
  - sensing
  - data processing
  - communicating components

• The position of sensor nodes need not be engineered or pre-determined.
  - sensor network protocols and algorithms must possess self-organizing capabilities
Sensor Networks vs. Ad Hoc Networks

- **Sensor networks**
  - Larger # nodes
  - More prone to failures.
  - Topology changes very frequently.
  - Mainly use broadcast communication whereas most ad hoc networks use point-to-point communications.
  - Limited in power, computational capacities, and memory
  - no global ID
Sensor networks communication architecture

Each of these scattered sensor nodes has the capabilities to collect data and route data back to the sink.

The sink may communicate with the task manager node via Internet or Satellite.
Factors influencing sensor network design

- fault tolerance
- scalability
- production costs
- operating environment
- sensor network topology
- hardware constraints
- transmission media
- power consumption
Hardware constraints

• A sensor node is made up of four basic components
  - sensing unit
    • usually composed of two subunits: sensors and analog to digital converters (ADCs).
  - processing unit
    • manages the procedures that make the sensor node collaborate with the other nodes to carry out the assigned sensing tasks.
  - transceiver unit
    • connects the node to the network
  - power units (the most important unit)
Hardware constraints

Fig. 1. The components of a sensor node.
Hardware constraints

• **Location finding system.**
  - Most of the sensor network routing techniques and sensing tasks require the knowledge of *location* with high accuracy.

• **Mobilizer**
  - May be needed to move sensor nodes when it is required to carry out the assigned tasks.
Hardware constraints

- matchbox-sized module
- consume extremely low power
- operate in high volumetric densities
- have low production cost and be dispensable
- be autonomous and operate unattended
- be adaptive to the environment
Sensor network topology

- **Pre-deployment and deployment phase**
  - Sensor nodes can be either thrown in mass or placed one by one in the sensor field.

- **Post-deployment phase**
  - Sensor network topologies are prone to frequent changes after deployment.

- **Re-deployment of additional nodes phase**
  - Addition of new nodes poses a need to re-organize the network.
Deployment Environments for Sensor Networks

- Micro-sensors, onboard processing, wireless interfaces feasible at very small scale--can monitor phenomena “up close”
- Enables spatially and temporally dense environmental monitoring

*Embedded Networked Sensing will reveal previously unobservable phenomena*

Ecosystems, Biocomplexity

Marine Microorganisms

Contaminant Transport

Seismic Structure Response
Transmission media

• Industrial, scientific, and medical (ISM) bands
  - 915 MHz ISM band has been widely suggested for sensor networks
  - offer license-free communication in most countries.

• Infrared
  - license-free and robust to interference
  - requirement of a line of sight between sender and receiver

• UWB (ultra wide band)
Power consumption

• Only be equipped with limited power source (<0.5 Ah 1.2V)
• Node lifetime strongly dependent on battery lifetime
• Power consumption can be divided into three domains:
  - sensing, communication, and data processing
Sensor networks communication architecture

- Used by the sink and sensor nodes

---

Fig. 3. The sensor networks protocol stack.
Routing in Sensor Networks
Routing in Sensor Networks

• Large scale sensor networks will be deployed, and require richer inter-node communication
  - In-network storage (DCS, GHT, DIM, DIFS)
  - In-network processing
  - “Fireworks routing”

• Need point-to-point routing to scale
  - Many nodes
  - Many flows
  - Different densities
Design Goals

1. Simple - minimum required state, assumptions
2. Scalable - low control overhead, small routing tables
3. Efficient - low routing stretch
4. Robust - node failure
How to route in a large sensor network?
GPSR: Greedy Perimeter Stateless Routing for Wireless Networks

Brad Karp, H. T. Kung
Harvard University
GPSR: Motivation

• Ad-hoc routing algorithms (DSR, AODV)
  - Suffer from out of date state
  - Hard to scale

• Use geographic information for routing
  - Assume every node knows position (x,y)
  - Keep a lot less state in the network
  - Require fewer update messages
GPSR Algorithm: Greedy Forwarding

- Each node knows the geographic location of its neighbors and destination
- Select the neighbor that is geographically closest to the destination as the next hop
GPSR Algorithm: Greedy Forwarding (Cont.)

- Each node only needs to keep state for its neighbors

- **Beaconing mechanism**
  - Provides all nodes with neighbors’ positions
  - Beacon contains broadcast MAC and position
  - To minimize costs: piggybacking
GPSR Algorithm: Greedy Forwarding (Cont.)

- Greedy forwarding does not always work!
Getting Around Void

• The right hand rule
  - When arriving at node $x$ from node $y$, the next edge traversed is the next one sequentially counterclockwise about $x$ from edge $(x,y)$
  - Traverse the exterior region in counter-clockwise edge order
Planarized Graphs

- A graph in which no two edges cross is known as **planar**.
  - Relative Neighborhood Graph (RNG)
  - Gabriel Graph (GG)
Relative Neighborhood Graph

An edge \( (u, v) \) exists between vertices \( u \) and \( v \) if the distance between them, \( d(u, v) \), is less than or equal to the distance between every other vertex \( w \), and whichever of \( u \) and \( v \) is farther from \( w \). In equational form:

\[
\forall w \neq u, v : d(u, v) \leq \max[d(u, w), d(v, w)]
\]
An edge \((u, v)\) exists between vertices \(u\) and \(v\) if no other vertex \(w\) is present within the circle whose diameter is \(\overline{uv}\). In equational form:

\[
\forall w \neq u, v : d^2(u, v) < [d^2(u, w) + d^2(v, w)]
\]
Final Algorithm

• Combine greedy forwarding + perimeter routing
  - Use greedy forwarding whenever possible
  - Resort to perimeter routing when greedy forwarding fails and record current location $L_c$
  - Resume greedy forwarding when we are closer to destination than $L_c$
Protocol Implementation

• Support for MAC-layer feedback
• Interface queue traversal
• Promiscuous use of the network interface
• Planarization of the graph
Simulation and Evaluation

• 50, 112, and 200 nodes with 802.11 WaveLAN radios.
• Maximum velocity of 20 m/s
• 30 CBR traffic flows, originated by 22 sending nodes
• Each CBR flows at 2Kbps, and uses 64-byte packets
Simulation and Evaluation

- Packet Delivery Success Rate
Simulation and Evaluation

- Routing Protocol Overhead
Simulation and Evaluation

- Path Length
Simulation and Evaluation

- Effect of Network Diameter
Simulation and Evaluation

• State per Router for 200-node
  - GPSR node stores state for 26 nodes on average in pause time-0
  - DSR nodes store state for 266 nodes on average in pause time-0
Pros and Cons?
GPSR

• *Combine greedy forwarding + perimeter routing*
  - Use greedy forwarding whenever possible
  - Resort to perimeter routing when greedy forwarding fails and record current location $L_c$
  - Resume greedy forwarding when we are closer to destination than $L_c$
Pros and Cons

• Pros:
  - Low routing state and control traffic ➔ scalable
  - Handles mobility

• Cons:
  - GPS location system might not be available everywhere.
  - Geographic distance does not correlate well with network proximity.
  - Overhead in location registration and lookup
  - Limitations of planarization algorithm
    • works under unit disk model, which doesn’t hold in practical network
    • hard to handle mobility
    • planarization reduces network connectivity
Beacon Vector Routing
Scalable Point-to-point Routing in Wireless Sensor Networks

R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, I. Stoica

UC Berkeley
Beacon Vector Routing

• Solution: fake geography
  - Create a routing gradient from connectivity information rather than geography
    • Nodes assigned positions based on connectivity
    • Greedy forwarding on this space
Beacon-Vector: Algorithm

• 3 pieces
  - Deriving positions
  - Forwarding rules
  - Lookup: mapping node IDs \(\rightarrow\) positions
Beacon-Vector: deriving positions

1. $r$ beacon nodes $(B_0, B_1, ..., B_r)$ flood the network; a node $q$'s position, $P(q)$, is its distance in hops to each beacon

$$P(q) = \langle B_1(q), B_2(q), ..., B_r(q) \rangle$$

2. Node $p$ advertises its coordinates using the $k$ closest beacons (we call this set of beacons $C(k, p)$)

3. Nodes know their own neighbors' positions

4. Nodes also know how to get to each beacon
Beacon-Vector: forwarding

1. Define the distance between two nodes \( P \) and \( Q \) as
   \[
   \dist_k(p, q) = \sum_{i \in C(k, q)} \omega_i \left| B_i(p) - B_i(q) \right|
   \]

2. To reach destination \( Q \), choose neighbor to reduce \( \dist_k(*, Q) \)

3. If no neighbor improves, enter Fallback mode: route towards the beacon which is closer to the destination

4. If Fallback fails and you reach the beacon, do a scoped flood

Does the forwarding scheme guarantee reachability?
Beacon maintenance

• Route based on the beacons the source and destination have in common
  - Does not require perfect beacon info.
• Each entry in the beacon vector has a sequence number
  - Periodically updated by the corresponding beacon
  - Timeout
• If the \#beacons < r, non-beacon nodes nominate themselves as beacons
Location directory

• Who maintain location directory?

• How to achieve scalability and resilience?

• How to find the nodes who maintain the directory?
Location directory

• Store location mapping at beacon nodes
  - Hashing \( H: \text{nodeid} \rightarrow \text{beaconid} \) \[14\]

• Each node \( k \) that wants to be a destination periodically publishes its coordinates to its corresponding beacon \( b_k = H(k) \)

• When a node wants to route to node \( k \), it sends a lookup request to \( b_k \)

• Cache the coordinates
Simple example
Simple example

Route from 3,2,1 to 1,2,3

Fallback towards $B_1$
Evaluation - Simulation

• Packet level simulator in C++
• Simple radio model
  - Circular radius, “boolean connectivity”
  - No loss, no contention
• Larger scale, isolate algorithmic issues
Evaluation - Implementation

• Real implementation and testing in TinyOS on mica2dot Berkeley motes

• 4KB of RAM!
  - Judicious use of memory for neighbor tables, network buffers, etc

• Low power radios
  - Changing and imperfect connectivity
  - Asymmetric links
  - Low correlation with distance

• Two testbeds
  - Intel Research Berkeley, 23 motes
  - Soda Hall, UCB, 42 motes
Simulation Results
Effect of the number of beacons

Can achieve performance comparable to that using true positions
Scaling the number of nodes

Number of beacons needed to sustain 95% performance

Beaconing overhead grows slowly with network size (less than 2% of nodes for larger networks)
Great benefit for deriving coordinates from connectivity, rather than positions.
Scope of floods

![Graph showing the average scope of flood vs. number of beacons for 3200 nodes and 10 routing beacons.]
Other results from simulation

- **Average stretch is consistently low**
  - Less than 1.1 in all tests

- **Performance with obstacles**
  - Modeled as walls in the network ‘arena’
  - Robust to obstacles, differently from geographic forwarding
Simulation Results

• Performance similar to that of Geographic Routing (small fraction of floods)
• Small number of beacons needed (<2% of nodes for over 95% of success rate w/o flooding)
• Scope of floods is costly
• Resilient to low density and obstacles
• Low stretch
Implementation Results
Routing performance

• Soda Testbed, 3100+ random pairs

- 88.4% success w/o flood
- 4.57% flood (avg. dist 2.6)
- 0.5% stuck (no good neighbor to forward)
- 6.5% drops (contention and radio drops)
Coordinate stability

• Coordinates are quite stable
  - E.g., almost 80% of the nodes had 2 or fewer changes, and over 90% of the changes were smaller than 3 hops
Implementation Results

• Success rates and flood scopes agree with simulation

• Sustained high throughput (in comparison to the network capacity)

• Coordinates were found to be stable
  - Few changes observed, small changes
Conclusions

• BVR is simple, robust to node failures, scalable, and presents efficient routes

• Using connectivity for deriving routes is good for low density/obstacles

• The implementation results indicate that it can work in real settings
Pros and Cons?
Comments

• Simple and easy to implement

• Work in real networks

• Routing and transmission stretches can be high

• No delivery guarantee even with scoped flooding. Why?