Sensor Networks

- Directed Diffusion
- Aggregation
- Assigned reading
  - TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks
  - Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks
Outline

- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

Smart-Dust/Motes

- First introduced in late 90’s by groups at UCB/UCLA/USC
  - Published at Mobicom/SOSP conferences
- Small, resource limited devices
  - CPU, disk, power, bandwidth, etc.
- Simple scalar sensors – temperature, motion
- Single domain of deployment (e.g. farm, battlefield, etc.) for a targeted task (find the tanks)
- Ad-hoc wireless network
Smart-Dust/Motes

- Hardware
  - UCB motes
- Programming
  - TinyOS
- Query processing
  - TinyDB
  - Directed diffusion
  - Geographic hash tables
- Power management
  - MAC protocols
  - Adaptive topologies

Berkeley Motes

- Devices that incorporate communications, processing, sensors, and batteries into a small package
- Atmel microcontroller with sensors and a communication unit
  - RF transceiver, laser module, or a corner cube reflector
  - Temperature, light, humidity, pressure, 3 axis magnetometers, 3 axis accelerometers
### Berkeley Motes

*(Levis & Culler, ASPLOS 02)*

<table>
<thead>
<tr>
<th>Mote Type</th>
<th>beeC</th>
<th>reen2</th>
<th>reen2</th>
<th>dot</th>
<th>mica</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>9/99</td>
<td>10/00</td>
<td>6/04</td>
<td>8/01</td>
<td>2/02</td>
</tr>
</tbody>
</table>

**Microcontroller**

<table>
<thead>
<tr>
<th>Type</th>
<th>AT99L8535</th>
<th>ATMega163</th>
<th>ATMega163</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prog. mem. (KB)</td>
<td>8</td>
<td>16</td>
<td>128</td>
</tr>
<tr>
<td>RAM (KB)</td>
<td>0.5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

**Nonvolatile storage**

<table>
<thead>
<tr>
<th>Chip</th>
<th>24LC256</th>
<th>AT45DB041B</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection type</td>
<td>I2C</td>
<td>SPI</td>
</tr>
<tr>
<td>Size (KB)</td>
<td>32</td>
<td>512</td>
</tr>
</tbody>
</table>

**Default Power source**

<table>
<thead>
<tr>
<th>Type</th>
<th>Li</th>
<th>Alk</th>
<th>Li</th>
<th>Alk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>CR2450</td>
<td>2xAA</td>
<td>CR2032</td>
<td>2xAA</td>
</tr>
<tr>
<td>Capacity (mAh)</td>
<td>575</td>
<td>2850</td>
<td>225</td>
<td>2850</td>
</tr>
</tbody>
</table>

**Communication**

<table>
<thead>
<tr>
<th>Radio</th>
<th>RFM TR1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate (Kbps)</td>
<td>10</td>
</tr>
<tr>
<td>Modulation type</td>
<td>OOK</td>
</tr>
</tbody>
</table>

### Sensor Net Sample Apps

**Habitat Monitoring:** Storm petrels on Great Duck Island, microclimates on James Reserve.

**Earthquake monitoring** in shake-test sites.

**Vehicle detection:** sensors along a road, collect data about passing vehicles.

**Traditional monitoring apparatus.**
Metric: Communication

- Lifetime from one pair of AA batteries
  - 2-3 days at full power
  - 6 months at 2% duty cycle
- Communication dominates cost
  - < few mS to compute
  - 30mS to send message

![Time v. Current Draw During Query Processing](image)

Communication In Sensor Nets

- Radio communication has high link-level losses
  - typically about 20% @ 5m
- Ad-hoc neighbor discovery
- Tree-based routing
The long term goal

Embed numerous distributed devices to monitor and interact with physical world: in workspaces, hospitals, homes, vehicles, and “the environment” (water, soil, air…)

Network these devices so that they can coordinate to perform higher-level tasks.

Requires robust distributed systems of tens of thousands of devices.
Motivation

- Properties of Sensor Networks
  - Data centric, but not node centric
  - Have no notion of central authority
  - Are often resource constrained
- Nodes are tied to physical locations, but:
  - They may not know the topology
  - They may fail or move arbitrarily
- Problem: How can we get data from the sensors?

Directed Diffusion

- Data centric – nodes are unimportant
- Request driven:
  - Sinks place requests as interests
  - Sources are eventually found and satisfy interests
  - Intermediate nodes route data toward sinks
- Localized repair and reinforcement
- Multi-path delivery for multiple sources, sinks, and queries
Motivating Example

• Sensor nodes are monitoring a flat space for animals
• We are interested in receiving data for all 4-legged creatures seen in a rectangle
• We want to specify the data rate

Interest and Event Naming

• Query/interest:
  • Type=four-legged animal
  • Interval=20ms (event data rate)
  • Duration=10 seconds (time to cache)
  • Rect=[-100, 100, 200, 400]

• Reply:
  • Type=four-legged animal
  • Instance = elephant
  • Location = [125, 220]
  • Intensity = 0.6
  • Confidence = 0.85
  • Timestamp = 01:20:40

• Attribute-Value pairs, no advanced naming scheme
Diffusion (High Level)

- Sinks broadcast interest to neighbors
- Interests are cached by neighbors
- Gradients are set up pointing back to where interests came from at low data rate
- Once a sensor receives an interest, it routes measurements along gradients

Illustrating Directed Diffusion
Summary

• Data Centric
  • Sensors net is queried for specific data
  • Source of data is irrelevant
  • No sensor-specific query

• Application Specific
  • In-sensor processing to reduce data transmitted
  • In-sensor caching

• Localized Algorithms
  • Maintain minimum local connectivity – save energy
  • Achieve global objective through local coordination

• Its gains due to aggregation and duplicate suppression may make it more viable than ad-hoc routing in sensor networks

Outline

• Sensor Networks

• Directed Diffusion

• TAG

• Synopsis Diffusion
TAG Introduction

- Programming sensor nets is hard!
- Declarative queries are easy
  - Tiny Aggregation (TAG): In-network processing via declarative queries
- In-network processing of aggregates
  - Common data analysis operation
  - Communication reducing
    - Operator dependent benefit
    - Across nodes during same epoch
- Exploit semantics improve efficiency!

Example:
- Vehicle tracking application: 2 weeks for 2 students
- Vehicle tracking query: took 2 minutes to write, worked just as well!

SELECT MAX(mag)
FROM sensors
WHERE mag > thresh
EPOCH DURATION 64ms

Basic Aggregation

- In each epoch:
  - Each node samples local sensors once
  - Generates partial state record (PSR)
    - local readings
    - readings from children
  - Outputs PSR during its comm. slot.

- At end of epoch, PSR for whole network output at root
- (In paper: pipelining, grouping)
Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```

Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```

Illustration: Aggregation

```
SELECT COUNT(*)
FROM sensors
```
### Illustration: Aggregation

```sql
SELECT COUNT(*) FROM sensors
```

<table>
<thead>
<tr>
<th>Sensor #</th>
<th>Slot #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Slot 3

### Illustration: Aggregation

```sql
SELECT COUNT(*) FROM sensors
```

<table>
<thead>
<tr>
<th>Sensor #</th>
<th>Slot #</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
</tbody>
</table>

Slot 4
Types of Aggregates

- SQL supports MIN, MAX, SUM, COUNT, AVERAGE

- Any function can be computed via TAG

- In network benefit for many operations
  - E.g. Standard deviation, top/bottom N, spatial union/intersection, histograms, etc.
  - Compactness of PSR
Taxonomy of Aggregates

- TAG insight: classify aggregates according to various functional properties
  - Yields a general set of optimizations that can automatically be applied

<table>
<thead>
<tr>
<th>Property</th>
<th>Examples</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial State</td>
<td>MEDIAN: unbounded, MAX: 1 record</td>
<td>Effectiveness of TAG</td>
</tr>
<tr>
<td>Duplicate Sensitivity</td>
<td>MIN: dup. insensitive, AVG: dup. sensitive</td>
<td>Routing Redundancy</td>
</tr>
<tr>
<td>Exemplary vs. Summary</td>
<td>MAX: exemplary, COUNT: summary</td>
<td>Applicability of Sampling, Effect of Loss</td>
</tr>
<tr>
<td>Monotonic</td>
<td>COUNT: monotonic, AVG: non-monotonic</td>
<td>Hypothesis Testing, Snooping</td>
</tr>
</tbody>
</table>

Benefit of In-Network Processing

Simulation Results
2500 Nodes
50x50 Grid
Depth = ~10
Neighbors = ~20

Some aggregates require dramatically more state!
Optimization: Channel Sharing (“Snooping”)

• Insight: Shared channel enables optimizations

• Suppress messages that won’t affect aggregate
  • E.g., MAX
  • Applies to all exemplary, monotonic aggregates

Optimization: Hypothesis Testing

• Insight: Guess from root can be used for suppression
  • E.g. ‘MIN < 50’
  • Works for monotonic & exemplary aggregates
    • Also summary, if imprecision allowed

• How is hypothesis computed?
  • Blind or statistically informed guess
  • Observation over network subset
Optimization: Use Multiple Parents

- For duplicate insensitive aggregates
- Or aggregates that can be expressed as a linear combination of parts
  - Send (part of) aggregate to all parents
    - In just one message, via broadcast
    - Decreases variance

Multiple Parents Results

- Better than previous analysis expected!
- Losses aren't independent!
- Insight: spreads data over many links

![Diagram showing the benefit of result splitting](image-url)
Outline

- Sensor Networks
- Directed Diffusion
- TAG
- Synopsis Diffusion

Aggregation in Wireless Sensors

Aggregate data is often more important

In-network aggregation over tree with unreliable communication

Count

Used by current systems,
TinyDB [Madden et al. OSDI’02]
Cougar [Bonnet et al. MDM’01]

Not robust against node- or link-failures
Traditional Approach

- Reliable communication
  - E.g., RMST over Directed Diffusion [Stann’03]
- High resource overhead
  - 3x more energy consumption
  - 3x more latency
  - 25% less channel capacity
- Not suitable for resource constrained sensors

Exploiting Broadcast Medium

- Robust multi-path
- Energy-efficient
- Double-counting
- Different ordering

➤ Challenge: order and duplicate insensitivity (ODI)
A Naïve ODI Algorithm

• Goal: count the live sensors in the network

Synopsis Diffusion (SenSys’04)

• Goal: count the live sensors in the network

Approximate COUNT algorithm: logarithmic size bit vector
Synopsis Diffusion over Rings

- A node is in ring $i$ if it is $i$ hops away from the base-station
- Broadcasts by nodes in ring $i$ are received by neighbors in ring $i-1$
- Each node transmits once = optimal energy cost (same as Tree)

Evaluation

Approximate COUNT with Synopsis Diffusion

- More robust than Tree
- Almost as energy efficient as Tree

Per node energy:

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Energy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>41.8 mJ</td>
</tr>
<tr>
<td>Syn. Diff.</td>
<td>42.1 mJ</td>
</tr>
</tbody>
</table>
Design Considerations

<table>
<thead>
<tr>
<th>Diffusion element</th>
<th>Design Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interest Propagation</td>
<td>• Flooding</td>
</tr>
<tr>
<td></td>
<td>• Constrained or directional flooding based on location</td>
</tr>
<tr>
<td></td>
<td>• Directional propagation based on previously cached data</td>
</tr>
<tr>
<td>Data Propagation</td>
<td>• Reinforcement to single path delivery</td>
</tr>
<tr>
<td></td>
<td>• Multipath delivery with selective quality along different paths</td>
</tr>
<tr>
<td></td>
<td>• Multipath delivery with probabilistic forwarding</td>
</tr>
<tr>
<td>Data caching and aggregation</td>
<td>• For robust data delivery in the face of node failure</td>
</tr>
<tr>
<td></td>
<td>• For coordinated sensing and data reduction</td>
</tr>
<tr>
<td></td>
<td>• For directing interests</td>
</tr>
<tr>
<td>Reinforcement</td>
<td>• Rules for deciding when to reinforce</td>
</tr>
<tr>
<td></td>
<td>• Rules for how many neighbors to reinforce</td>
</tr>
<tr>
<td></td>
<td>• Negative reinforcement mechanisms and rules</td>
</tr>
</tbody>
</table>

**Figure 3: Design Space for Diffusion**
Directed Diffusion (Data)

- Sensors match signature waveforms from codebook against observations
- Sensors match data against interest cache, compute highest event rate request from all gradients, and (re)sample events at this rate
- Receiving node:
  - Finds matching entry in interest cache, no match – silent drop
  - Checks and updates data cache (loop prevention, aggregation)
  - Retrieve all gradients, and resend message, doing frequency conversion if necessary

Directed Diffusion (Reinforcement)

- Reinforcement:
  - Data-driven rules unseen msg. from neighbor → resend original with smaller interval
  - This neighbor, in turn, reinforces upstream nodes
  - Passive reinforcement handling (timeout) or active (weights)
Approach

• Energy is the bottleneck resource
  • And communication is a major consumer—avoid communication over long distances
• Pre-configuration and global knowledge are not applicable
  • Achieve desired global behavior through localized interactions
  • Empirically adapt to observed environment
• Leverage points
  • Small-form-factor nodes, densely distributed to achieve Physical locality to sensed phenomena
  • Application-specific, data-centric networks
  • Data processing/aggregation inside the network

Directed Diffusion Concepts

• Application-aware communication primitives
  • expressed in terms of named data (not in terms of the nodes generating or requesting data)
• Consumer of data initiates interest in data with certain attributes
• Nodes diffuse the interest towards producers via a sequence of local interactions
• This process sets up gradients in the network which channel the delivery of data
• Reinforcement and negative reinforcement used to converge to efficient distribution
• Intermediate nodes opportunistically fuse interests, aggregate, correlate or cache data
Query Propagation

Synopsis Diffusion (SenSys’04)

• Synopsis Diffusion: a general framework

<table>
<thead>
<tr>
<th>Method</th>
<th>Count</th>
<th>Uniform Sample</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum (Average)</td>
<td>Iceberg queries</td>
<td></td>
</tr>
<tr>
<td>Distinct count</td>
<td>Top-k items</td>
<td></td>
</tr>
</tbody>
</table>
Aggregation Framework

- As in extensible databases, we support any aggregation function conforming to:

\[
\text{\textit{Agg}}_n = \{ f_{\text{init}}, f_{\text{merge}}, f_{\text{evaluate}} \}
\]

\[
f_{\text{init}}(a_0) \rightarrow \langle a_0 \rangle
\]

\[
f_{\text{merge}}(\langle a_1 \rangle, \langle a_2 \rangle) \rightarrow \langle a_{12} \rangle
\]

\[
f_{\text{evaluate}}(\langle a_1 \rangle) \rightarrow \text{aggregate value}
\]

(Merge associative, commutative!)

**Example: Average**

AVG_{\text{init}} \{v\} \rightarrow \langle v, 1 \rangle

AVG_{\text{merge}} \{\langle S_1, C_1 \rangle, \langle S_2, C_2 \rangle\} \rightarrow \langle S_1 + S_2, C_1 + C_2 \rangle

AVG_{\text{evaluate}}(\langle S, C \rangle) \rightarrow \frac{S}{C}