TAG: A Tiny Aggregation Service for Ad-Hoc Sensor Networks

Samuel Madden, Michael Franklin, Joseph Hellerstein, and Wei Hong
UC Berkeley
CS 102
Presented by: [Name]
Database Group. University of Pennsylvania
[Note from presenter: courtesy of Samuel Madden]

TAG Introduction

- What is a sensor network?
- Programming sensor nets is hard!
- Declarative queries are easy
- Tiny Aggregation (TAG) - in-network processing via declarative queries
- Connection to previous papers in class
  - Declarative query processing: few lines of SQL, many lines of C.
  - Do as much processing in the network as possible
- Example:
  - Vehicle tracking application: 2 weeks for 2 students.
  - Vehicle tracking query: took 2 minutes to write, worked just as well.

Sensor Networks (cont)

- Sensor networks vs. other networks
  - Messages get lost quite often (20-70%)
  - Payload is small (e.g., 30 bytes) - maximum is total throughput
  - Packet overheads, large proportion of payload
- Why?
  - Some signal decay: two A sensors 1 meter apart.
  - No forward error correction (e.g., Turbo Codes in cell phones): limited power and inventory.
  - Larger packet means larger probability of collision (and more lost messages)
- Power needs to be limited
  - Can do processing but not too sophisticated (e.g., Chaos and BackChaos)
- Bandwidth requirements - need to process and send while others are still listening.
  - Clock is slow - tens of MHz.
  - Also operate for months on a couple of batteries.

Overview

- Sensor Networks
- Queries in Sensor Nets
- Tiny Aggregation
  - Overview
  - Simulation & Results

Device Capabilities

- "Mica Motes" (2002): still widely used
  - 8-bit ARM processor
  - Roughly a PC AT
  - 40 Mbit COMA radio
  - 1 kilobyte RAM, 5 kilobyte ROM
  - TinyOS based
- Variety of other similar platforms exist
  - UC Irvine: Windows, Princeton ZebraNet, Spark
  - Others: Tidbits: 10Kb, research: Intel Mootez (520 MHz, 32Kb RAM).

Sensor Net Sample Apps

- Habitat Monitoring: Storm
  - Storm petrels grant; duck island, Alaska, National Science Foundation
- Earthquake Monitoring in shake test lab
- Vehicle detection: sensors along a road, collect data about passing vehicles.
- Final round open parking appelleation.
**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 communicate
  - 30ms to send message
- Our metric: communication

---

**Communication In Sensor Nets**

- Radio communication has high link-level losses
  - Typically about 20% @ 5m
- Ad-hoc neighbor discovery
- Tree-based routing
  - Aggregation tree, rooted at the base station
  - Choose parent at minimum distance to base station

---

**Overview**

- Sensor Networks
- Queries in Sensor Nets
- Tiny Aggregation
  - Overview
  - Optimizations & Results

---

**Declarative Queries for Sensor Networks**

**Examples:**

<table>
<thead>
<tr>
<th>Epoch</th>
<th>Sensor</th>
<th>Light</th>
<th>V usefulness</th>
<th>T usefulness</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>d</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>1</td>
<td>f</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>f</td>
<td>1</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

*“Where” predicate is local, can be pushed down the aggregation tree*

---

**Aggregation Queries**

2. SELECT AVG(sound) FROM sensors EPOCH DURATION 30s

<table>
<thead>
<tr>
<th>Epoch</th>
<th>AVG(sound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>440</td>
</tr>
<tr>
<td>1</td>
<td>445</td>
</tr>
</tbody>
</table>

3. SELECT roomNo, AVG(sound) FROM sensors GROUP BY roomNo HAVING AVG(sound) > 200 EPOCH DURATION 30s

<table>
<thead>
<tr>
<th>Epoch</th>
<th>AVG(sound)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>340</td>
</tr>
<tr>
<td>1</td>
<td>520</td>
</tr>
<tr>
<td>2</td>
<td>420</td>
</tr>
</tbody>
</table>

---

**Overview**

- Sensor Networks
- Queries in Sensor Nets
- Tiny Aggregation
  - Overview
  - Optimizations & Results
TAG

- In-network processing of aggregates
  - Common data analysis operation
    - Also gathering, filtering, or reduction in parallel programming
    - Communication reducing
      - Operator dependent benefit
      - Across nodes during same epoch
  - Exploit semantics improve efficiency!

Basic Aggregation

- In each epoch:
  - Each node samples local sensors once
  - Generates partial state record (PSR)
    - Local readings
    - Results from children
  - Outputs PSR during its comm. slot.
- At the end of epoch, PSR for whole network output at root
- (In paper: pipelining, grouping)

Communication Staggering

Illustration: Aggregation

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

<table>
<thead>
<tr>
<th>Comm. Slot #</th>
<th># of child sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>2</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>3</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>4</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>

Illustration: Aggregation

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

<table>
<thead>
<tr>
<th>Comm. Slot #</th>
<th># of child sensor</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>2</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>3</td>
<td>1 2 3 4 5</td>
</tr>
<tr>
<td>4</td>
<td>1 2 3 4 5</td>
</tr>
</tbody>
</table>
Illustration: Aggregation

Illustration: Aggregation

Illustration: Aggregation

Illustration: Aggregation

Aggregation Framework

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

$$
\text{Aggregation Function} (f) = \text{Function Output} \rightarrow \text{Aggregate Value}
$$

Example: Average

- AVG: \( \rightarrow \left\langle \frac{1}{n} \sum_{i=1}^{n} x_i \right\rangle \)

- AVG$_{max}$: \( \rightarrow \left\langle \max \{x_1, x_2, \ldots, x_n \} \right\rangle \)

- AVG$_{min}$: \( \rightarrow \left\langle \min \{x_1, x_2, \ldots, x_n \} \right\rangle \)

- AVG$_{mean}$: \( \rightarrow \left\langle \frac{1}{n} \sum_{i=1}^{n} x_i \right\rangle \)

Types of Aggregates

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

- Any of these functions can be computed via TAG

- In-network benefit for many operations:
  - E.g., standard deviation, top/bottom N, spatial union/intersection, histogram, etc.
  - Compactness of PSR necessary for efficient operation

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>Example</th>
<th>Affects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partial State</td>
<td>MEDIAN, COUNT, MAX, MIN</td>
<td>Effectiveness of TAG</td>
</tr>
<tr>
<td>Skew-Character</td>
<td>MIN, MAX, mean</td>
<td>Reading latency</td>
</tr>
<tr>
<td>Dependency vs. Summation</td>
<td>MAX, COUNT, SUM</td>
<td>Applicability of sampling, effect of LRU</td>
</tr>
<tr>
<td>Semantic</td>
<td>MAX, MIN, mean</td>
<td>Hypothetical editing, shaping</td>
</tr>
<tr>
<td>Metric</td>
<td>MAX, MIN, mean</td>
<td>Hypothetical editing, shaping</td>
</tr>
</tbody>
</table>
**TAG Advantages**

- Communication Reduction
  - Important for power and contention
- Continuous stream of results
  - Smooth transient faults across epochs
    - Interpolate FRs between epochs
- Lots of optimizations
  - Via operator semantics

**Simulation Environment**

- Evaluated via simulation
- Coarse grained event-based simulator
  - Sensors arranged on a grid
  - Two communication models
    - Lossless: All neighbors hear all messages
    - Lossy: Messages lost with probability that increases with distance

**Benefit of In-Network Processing**

<table>
<thead>
<tr>
<th>Simulation Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>2500 Nodes</td>
</tr>
<tr>
<td>50x50 grid</td>
</tr>
<tr>
<td>Depth = ~10</td>
</tr>
<tr>
<td>Neighbors = ~20</td>
</tr>
</tbody>
</table>

- Some aggregates require dramatically more state!

**Optimization: Channel Sharing ("Snooping")**

- Insight: Shared channel enables optimizations
- Suppress messages that won’t affect parent aggregate - "horizontal aggregation"
  - 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
    - A posteriori, if impression allowed with some error bound
- How is hypothesis computed?
  - Blind or statistically informed guess
  - Observation over network subset

**Experiment: Hypothesis Testing**

- Uniform Value Distribution
- Dense Packing
- Ideal Communication

![Graph showing messages per epoch vs. network diameter]
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
  - Injust one message, we broadcast
  - Decreases variance

Multiple Parents Results

- Benefits: pre-exist multiple parents
- Load balancing with multiple parents
- Independently decouple link

Summary

- TAG enables in-network declarative query processing
- State dependent communication benefit
- Transparent optimization via taxonomy
- Hyperbolic Testing, Parent Sharing
- Declarative queries are the right interface for data collection in sensor nets!
- Easier to program and more efficient for vast majority of users

Some Later Research

- Load distribution
- Join optimization