Aggregation in Sensor Networks

NEST Weekly Meeting
Sam Madden
Rob Szewczyk
10/4/01
**Why bother with aggregation**

- **Individual sensor readings are of limited use**
  - Interest in higher level properties, e.g. what vehicles drove through, what is the spread of temperatures in the building
  - We have a processor & network on board, let's use it

- **We cannot survive without aggregation**
  - Delivering a message to all nodes much easier than delivering a message from each node to a central point
  - Delivering a large amount of data from every node harder still, vide connectivity experiment
  - Forwarding raw information too expensive
    » Scarce energy
    » Scarce bandwidth
    » Multihop performance penalty
Aggregation challenges

- Inherently unreliable environment, certain information unavailable or expensive to obtain
  - how many nodes are present?
  - how many nodes are supposed to respond?
  - what is the error distribution (in particular, what about malicious nodes?)
  - Trying to build an infrastructure to remove all uncertainty from the application may not be feasible – do we want to build distributed transactions?

- Information trickles in one message at a time
  - Never have a complete and up-to-date information about the neighborhood

- What type of information should we expect from aggregation
  - Streams
  - Robust estimates
Scenario: Count
**Scenario:** Count

**Goal:** Count the number of nodes in the network.

Number of children is unknown.

**Goal:** Count the number of nodes in the network.

Number of children is unknown.
Scenario: Count

Goal: Count the number of nodes in the network.

Number of children is unknown.

Sensor #

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Scenario: Count
Scenario: Count

Goal: Count the number of nodes in the network.

Number of children is unknown.

Goal: Count the number of nodes in the network.

Number of children is unknown.
Goal: Count the number of nodes in the network.

Number of children is unknown.

Scenario: Count
Scenario: Count

Goal: Count the number of nodes in the network.

Number of children is unknown.
Scenario: Count

Goal: Count the number of nodes in the network.

Number of children is unknown.

Sensor #

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1+2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1+2</td>
<td>1+1/2</td>
<td>1+1/2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1+3</td>
<td>1+1/2</td>
<td>1+1/2</td>
<td>1+1</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Scenario: Count

Goal: Count the number of nodes in the network.

Number of children is unknown.

Sensor #

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1 + 2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>1 + 2</td>
<td>1 + ½</td>
<td>1 + ½</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 + 3</td>
<td>1 + ½</td>
<td>1 + ½</td>
<td>1 + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 + 3</td>
<td>1 + 2/2</td>
<td>1 + 2/2</td>
<td>1 + 1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 + 4</td>
<td>1 + 2/2</td>
<td>1 + 2/2</td>
<td>1 + 1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Sensor #

Number of children is unknown.

Scenario: Count
Counting Lessons

• Take advantage of redundancy to improve accuracy (reply to all parents, not just one)
• Use broadcast to reduce number of messages
• Result is a stream of values: much more robust to failures, movement, or collision than a single value.
Aggregation in network programming

• Network programming problem
  – Reliable delivery of a large number of messages to all nodes in range, while exploiting the broadcast nature of the medium

• Basic setup
  – Broadcast a known number of idempotent program fragments
  – Each node keeps a bitmap of fragments received (1=packet received)
  – Two stages of the problem: single hop, and multihop

• Solutions
  – Single hop, dense cell
    » Broadcasting the program – trivial, the central node broadcasts
    » Feedback from nodes – broadcast a request from the central node: Is anyone missing packets in this packet range?
    » Convergence: no replies to the request
Aggregation in multihop network programming

• Broadcasting the program – use flooding
  – Remember the last 8 packets forwarded, use that cache to decide whether to forward or not

• Feedback from nodes
  – Distribute requests for feedback using the flooding
  – After some delay, respond if any packets are missing locally
  – Responses from children: AND with the local bitmap, store the result locally, forward the request
    » Suboptimal because there is no local fixups

• Convergence
  – No replies to the request
Aggregation over streams

- **Inherent uncertainty of the system**
  - Can nodes communicate, do they have enough power, have they moved?
  - Computing a complete single answer can be very expensive, and may not be possible
  - Partial estimates have their own value

- **Aggregation over streams**
  - Values reflect the current best estimates
  - Self stabilizing: in the absence of changes converges to a desired value within $N$ steps
What does it mean to aggregate (The DB Perspective)

• General purpose solution: apply standard aggregation operators like COUNT, MIN, MAX, AVERAGE, and SUM to any set of sensors.
  – Previous example are application specific
  – In sensors, operators may be arbitrary signal processing functions

• Provide grouping semantics: e.g. ‘select avg(temp) group by trunc(light/10)’
  – In sensor networks, groups may be random samples
Identifying Groups

• Need a way to identify groups
  – Idea: set of membership criteria pushed down
    » Nodes determine their membership set based on those criteria
    » Nodes can be in multiple but not unlimited groups
    » E.g. “Group 1 : 0 <= t < 10, Group 2 : 10 <= t < 20, ...”

• Need a way to evaluate aggregation predicates by group

• May want to allow grouping and aggregation predicates to be expressed together to take advantage of broadcast effects
Local Query Rewrite

• Intermediate nodes may determine that it’s faster to evaluate an aggregate by asking children a different question.
  – Example 1: MAX(t). Once we have a guess T for MAX, ask children to report iff t > T, rather than asking all children to compute a local maximum.
  – Example 2: Network programming. Rather than asking nodes what packets they have, ask them to report iff packets missing.

• Is this a general technique? Maybe:
  – Inform child of guess at aggregate, ask it to refute.
    » Works for average (within error bound), not count.
Wins and pitfalls of aggregation

• Aggregation over natural network topology
  – Aggregation over an arbitrary subset of the network may be a loss

• Really dense cells
  – Aggregation does not help with the starvation problem
  – Use the message suppression via query rewrite technique
  – Still beneficial in a multihop scenario
Discussion

• More advanced aggregation: Least squares
  – Challenge of deciding what samples are related
  – Vehicle tracking examples in 29 Palms

• What type of aggregation is useful for this group?

• What kind of support should the OS offer to support aggregation?