Information management (1)

Feng Zhao
Papers

- Querying and in-network aggregation
  - Madden et al., “TAG: a Tiny AGgregation Service for Ad Hoc Sensor Networks.”
  - Hellerstein, “Beyond Averages: Towards Sophisticated Sensing with Queries.”

- Data-centric storage and access
  - Ratnasamy et al., “GHT: A Geographic Hash Table for Data-Centric Storage.”
### Sensor Network Abstraction

<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>

- **Characteristics:** distributed, resource-constrained, failure prone
  - How does an application user specify a problem?
  - How does a system developer model the capabilities of the system?
  - What features should a lower layer expose to higher layers?

- **From data storage point of view:** think of a sensor net as a distributed database
  - How is data stored after sensing?
  - What is the interface to the network?
  - How does an external query find the data in an efficient manner?

Figure source: Estrin et al.
TinyDB meets TinyOS

TinyDB

TinyDB meets TinyOS

TARGET_TRACKING

BLESS

CLOCK

PHOTO

LEDS

GENERIC_COMM

MAIN

PC

Mote

Query

FIELDS
nodeId
light
OPS
NULL

Result

1 28
2 55
3 46

Result

2 55

Result

3 48

SELECT nodeId, light FROM SENSORS
Querying sensor networks: TAG and Tiny DB

- SQL-like declarative interface
  - `SELECT max(temp), room FROM sensors WHERE floor = 6 GROUP BY room HAVING max(temp) > threshold EPOCH DURATION 30sec`

- In-network aggregation to reduce message count
  - See example on the right

- Manage transmission and aggregation:
  - A hierarchical routing tree
  - Epoch-based time division
    » Time synch is important
  - Repair routing

Left: centralized, requiring 16 message to be sent. Right: in-network aggregation, requiring 6 messages to be sent
### Aggregates and their efficiency in TAG

<table>
<thead>
<tr>
<th></th>
<th>MAX, MIN</th>
<th>COUNT, SUM</th>
<th>AVERAGE</th>
<th>MEDIAN</th>
<th>COUNT DISTINCT</th>
<th>HISTOGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicate Sensitive</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Exemplary (E), Summary (S)</td>
<td>E</td>
<td>S</td>
<td>S</td>
<td>E</td>
<td>S</td>
<td>No</td>
</tr>
<tr>
<td>Monotonic</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>S</td>
</tr>
<tr>
<td>Partial State</td>
<td>Distributive</td>
<td>Distributive</td>
<td>Algebraic</td>
<td>Holistic</td>
<td>Unique</td>
<td>Content-Sensitive</td>
</tr>
</tbody>
</table>

Some can be decentralized better than others!
Extend TAG beyond averages to spatio-temporal aggregates

- Geometry
  - Example: isobar mapping

- Resolution
  - Example: wavelet compression

- Time (Example: tracking)
  - Who triggers the aggregation?
  - Interface between querying and tracking?
Data Centric Storage

- TinyDB routing tree or fixed storage point supports only limited access patterns
- Need more general indexing scheme
  - Scalable
  - Load balance
  - Tolerant to failures and changes
    » Persistent
    » Consistent
Geographic Hash Table

- Data is indexed by geographic coordinates
  - Data centric: data is named by physical attributes external to the nodes or network topology
  - Geographic hashing
    » Key of a key-data pair hashed to geo location (for both PUT and GET)
    » Load balanced
  - Storage localization:
    » GPSR geographic routing (discussed earlier) to find storage node, defined as the node nearest to the geo location
Locate storage node using GPSR

- Home node: closest to the geo location
- Replica nodes: those along the perimeter enclosing the geo location
- Perimeter Refresh Protocol to ensure persistency and consistency
  - Node may fail or move
  - Home node periodically sends out refresh packet to the geo location. This updates the home node when necessary
  - Time-out mechanism to deal with home node failure