A Sensor Network Architecture

Philip Levis
Stanford University
The Internet and sensornets have different end goals, but many of the same architectural principles apply.

### Network Design Goals

<table>
<thead>
<tr>
<th>Internet</th>
<th>Sensornet</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interconnect separate networks</td>
<td>Dense real world monitoring</td>
</tr>
<tr>
<td>Resilience to loss and failure</td>
<td>Resilience to loss, failure and noise</td>
</tr>
<tr>
<td>Support many protocols</td>
<td>Support many applications</td>
</tr>
<tr>
<td>Accommodate variety</td>
<td>Scale to large, small, and long</td>
</tr>
<tr>
<td>Distributed management</td>
<td>Self-organizing</td>
</tr>
<tr>
<td>Cost effective</td>
<td>Cost effective</td>
</tr>
<tr>
<td>Low effort attachment</td>
<td>Easily composition</td>
</tr>
<tr>
<td>Resource accountability</td>
<td>Evolvable in resources</td>
</tr>
</tbody>
</table>

... but many of the same architectural principles apply
Dense Real World Monitoring

- An embedded sensornet is not a transit network
  - Pure connectivity is not the end goal
- Transit networks benefit from hiding proximity
  - Can talk to something close just like something far away
  - Far and close depend on logical, not physical topology
- Data networks benefit from revealing proximity
  - Proximity reveals cost, proximity reveals relevance
- Intermingling of data collection and processing does not preclude layers: it just calls for new ones
The Four Node Primitives

- Sensing
- Computation
- Communication
- Storage
The Four Network Primitives

- Sensing: collecting data when needed
- Filtering: reducing data volume (inter-, intra-node)
- Communication: getting data to its destination
- Storage: buffering vs. indexing

The decisions of when, where, and how to use these four primitives are all driven by a need to optimize the tradeoff between energy cost and data fidelity.
Example Network
Communication

- Two topologies: DAGs and fields (dissemination)
  - Latency: high or low
  - Rate: high or low (bulk or small)
- Take advantage of SP architecture underneath
  - Buffering lots of data can be a good strategy
  - Large data items gives flexibility
    - Per-packet reliability
    - Bulk reliability
Three Routing Primitives

- **Collection**
  - Datagram-based protocol
  - Immediate forwarding
  - Congestion control

- **Channel**
  - Stream-based protocol (for efficiency)
  - Dedicated explicit path

- **Buffered path**
  - Stream-based protocol
  - Non-volatile in-network storage along path
# Routing Table

<table>
<thead>
<tr>
<th>Latency</th>
<th>Rate</th>
<th>System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>Low</td>
<td>Collection</td>
</tr>
<tr>
<td>High</td>
<td>Low</td>
<td>Buffer</td>
</tr>
<tr>
<td>Low</td>
<td>High</td>
<td>Channel</td>
</tr>
<tr>
<td>High</td>
<td>High</td>
<td>Buffer? STRAW?</td>
</tr>
</tbody>
</table>
Collection Example
Channel Example
Buffered Paths Example
Buffer Example
Buffer Example
Buffer Example
Buffer Example
Tree Formation

- Every node has a tree formation service
  - Can support up to $N$ trees concurrently
  - Each tree has a unique identifier
- Tree formation/destruction
  - Consistent algorithm based on ID
Tree Formation

<table>
<thead>
<tr>
<th>Tree</th>
<th>Node</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
<td>33</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>21</td>
</tr>
<tr>
<td>9</td>
<td>17</td>
<td>14</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tree</th>
<th>Node</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
<td>33</td>
</tr>
<tr>
<td>1</td>
<td>38</td>
<td>35</td>
</tr>
<tr>
<td>8</td>
<td>22</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>48</td>
</tr>
<tr>
<td>6</td>
<td>17</td>
<td>33</td>
</tr>
</tbody>
</table>
Data Model

- All data is typed with an opaque identifier
Filters

- Filters apply to a single data item
  - Input: data item
  - Output: data item or null
  - Can keep state, transform types
- Predicates, EWMA, n-to-1 TAG operators
- Input filters vs. output filters
  - Pushing to outputs is generally preferred
  - Tradeoff in transmitting filter state vs. data
- Source management
Storage

- Buffered storage
  - Typed, queue policy (FIFO, priority, etc.)
  - Buffered and channel communication
- Indexed storage ("file system storage")
  - Persistent (allocation/deallocation)
  - Caching, indexing, organization are all open questions
Storage Example
Programming

- Programming is defining the placement of a library of primitives within the network
  - Declarative vs. Dynamic vs. Imperative
  - Aggregation, file system, query parsing
- Chaining primitives into larger services

Collect events, filter as an aggregate, store in indexed storage, query for a large piece of data: GHT
A Layered Approach

- Application
- Storage
- Filtering
- Multihop
- Filtering
- Storage
- Single Hop
- Neighborhood
- Data Link
Open Questions

- Data forking
  - E.g., send to two places, store and send, etc.
  - Flexibility vs. expressiveness

- Dissemination
  - Used to create network structure (channels, etc.)
  - Dissemination classes

- Data model
  - Typing, type registries
  - Raw vs. processed values
Questions