Challenges in Programming Sensor Nets

Feng Zhao
Networked Embedded Computing Group
Microsoft Research

DCOSS05 keynote, July 1, 2005
Acknowledgment

- Joint work with Jie Liu, Alec Woo, Elaine Cheong, Kamin Whitehouse, Siddharth Seth, Prabal Dutta
- Discussions with Jeremy Elson and Leo Guibas

Networked Embedded Computing Group, http://research.microsoft.com/nec
From Mojave Desert to India Hills …
Lessons Learned

OS/Network-centric view

Designing component-level abstractions

Information-centric view

Designing application-level abstractions

Majority of the code is in glue logic!

Macroprogramming (TinyDB, Regions, Kairos, State-space, EnviroTrack …)
Programming Challenges

You’ve heard all of this:

• Limited node capability
• Limited, uncertain connectivity
• Complex collaborative processing tasks

More importantly:

• Sensors are not just a random collection of objects
• Rich structure in system (e.g., connectivity, link quality, node density)
• Rich structure in sensing (e.g., spatio-temporal coherence, data reliability)
• Programming models should provide abstractions for these structures
• Permit couplings between system and information processing design
Proliferation of networked devices

Characteristics:
- Heterogeneous devices
- Disparate capabilities
- Physically embedded (energy, size, noise, real-time events, …)
- Dynamic topology
- Large scale
- Inherent uncertainties (in systems and environment)
- Concurrent user queries

Desired properties:
- Easy to program, deploy, and manage
- Robust to failure
- Responsive
- Re-taskable
- Scalable
- Secure
A Scenario: What’s happening in the garage?

Multiple, concurrent queries:

- Traffic engineering: where can one find a parking spot?
- Security: what is going on down there?
- Corporate health services: when should the air exhaust fan run?
One may argue for central data collection and server programming.

Unlike traditional query processing, need:
- In-network processing
- Semantic interpretation of raw data streams
- Handling uncoordinated tasks injected any time/any place

But centralized data collection is not scalable with the number of nodes.
Parking garage testbed/scenarios

When should I arrive in order to park on P1?

Get magnetic signatures of moving objects.

Take pictures of speeding vehicles.
But the problem is made hard by …

• A great deal of uncertainty
  – System dynamism and unreliability
    • Nodes/links come and go
    • Possibly non-replenishable resources
    • Parts of system may be deployed over time by different vendors, using different technologies (e.g., network protocols)
  – Data uncertainty
    • Uncertain data due to sensor noise, packet loss
    • Incomplete information due to partial observability of the world

• User tasks often specified in high-level languages
  – E.g., “Tell me if you see a red car”, or, “Doing this with that data”.

• As a result, a number of systems have been built vertically
  – With own system abstractions and data models
  – Make sense from efficiency pov, but with relatively small code reuse
Where is the narrow waist?
Proposed Common Interfaces

- Task Pool
  - Declarative queries
  - Planner
  - Application Programming
    - GUI
    - Excel
    - Service lib
    - Config scripting

- Sensor Net Deployment
  - Tasking ML
  - Tasking ML
  - Concrete services

- System Run Time Support
  - Sensor tasking
    - Optimizer 1
    - Optimizer 2
  - Run time monitoring

- Sensor Net Deployment
  - Abstract services, dependencies, priorities, E-E constraints
  - Link reliability, net latency, data fidelity
Task Pool Abstraction

- Application programs inject abstract services and their dependencies into task pool, specifying what/how
  - Services have priorities
  - Applications have end-to-end constraints such as latency, data quality, energy usage

- Sensor Tasking embed services onto nodes, instantiate where/when
  - Need information about physical topology, link quality, latency, data fidelity from run time
  - Search for optimal assignment satisfying EE constraints
    - Using a variety of tools such as CLP(R), LP, or Monte Carlo
    - Tasking ML describes concrete services instances

- Task pool is agnostic to application programming environments and run-time system support, provided that
  - Services are described in a common intermediate language
  - Run-time provides system and data reliability info
Concurrent Uncoordinated Tasks

Tasks are sent to microservers at uncoordinated times, running for unpredictable duration. Tasks may partially overlap.
A Service Model for Sensor Net

Sensor net as provider of services
- Services encapsulating data and computation
- Service discovery, composition, execution
- Tasking description

![Diagram of A Service Model for Sensor Net]

- User
- Query
- Planner
- Service graph
- Service Embedding
- Service Discovery/ Self-monitoring
- Execution
- Service scheduler
- Service Scheduler
- Tasking Progress
- Tasking ML
- Run-time resource info
- Run time
- Schedule time
Example of services and their composition

**Counting vehicles** with a sensor array

- Extract edges from break beam detections
- Sort edges into consecutive detections
- Detect vehicles based on timing relations among detections
- Count vehicles
- Generate an arrival histogram report
SONGS: Service Oriented Networked ProGramming of Sensors

Service Abstraction and Interface

Service Planning

Service Embedding

Service Scheduling and Execution

histogram(S)
car(X)
speed(X, S)

{S>30}
Service Abstraction and Interfaces

• In embedded sensing systems, service abstractions are more than just invocation interfaces.

• A rich service interface may contain:
  – Interaction interface
    • data I/O
    • data semantics (e.g. sensing modality, sampling rate, uncertainty)
  – Location
  – Resource requirements (e.g. in power and bandwidth)
  – Fidelity specifications (e.g. uncertainties in processing)
Service Planning

- High-level user interface eases programming
  - Declarative or Excel-like
- Automatic service planning uses goal-directed inference
- Resource sharing and reuse supports system retaskability
- Constraint-based specification allows flexibility in design trade-off
Automatic Service Planning

- Declarative specification
- Automatic resource sharing and reuse
- Reasoning about space/time constraints
- Execution monitoring and re-planning
- Constraint-based trade-offs

**Service Description**

```prolog
sensor( magSensor, [[60,0,0],[70,10,10]])
sensor( camera, [[40,0,0],[55,15,15]]).
sensor( breakSensor, [[10,0,0],[12,10,2]])

service( breakService (Region),
    needs( sensor (breakSensor,Region) ),
    creates( break (X), detected (X,T,Region) )).
```

**User Query**

```
car(X), speed(X,S), histogram(S), {S>30}.
```
car(X), magStream(Y), triggered(Y,X), photo(Z), triggered(Z,Y), length2(X,L), (L>30), delay(T,T2,Delta), (Delta<30), intensity(X,Y), (l<100)

break(X)
broadcastGroup(Region, GroupName)
car(X)
confidence(C,C)
delay(T,T2,Delta)
detected(X,T,Region)
direction(X,D)
frequency(X,Freq)
human(X)
intensity(X,Y)
length2(X,L)
lengthHistogram(X)
magStream(X)
photo(X)
sensor(Name, Region)
speed(X,S)
speedHistogram(X)
supports(GroupName, X)
timeHistogram(X)
triggered(X,Y)
Service Embedding

- Embed service graph onto physical nodes
- Preserve proximities in data flows
- Optimize for resource usage, load balance, latency, and robustness
- Exploit physical abstractions to find good mapping
Proximity Based Greedy Search

- Assume nodes close in physical space are also close in the communication graph.
- Sort nodes in SCG into levels, where sensor nodes at bottom level are anchored.
- Starting from the bottom, for each floating service, place it at the weighted geographical mean of child nodes.
- Repeat until all services are placed.
- Map each service at a virtual location to nearest physical node.
An Example of TML

- Task graph for counting vehicle query
  - Ports
  - Services
  - Wiring services together
- Description in Tasking ML (micro-server tasking markup language)
Service Run Time (SERUN)

- On-demand instantiation
- Adaptive to resource availability
- Collaboration among peers and between tasks
- Dynamic service migration
A Service Sharing Runtime Environment

- Event-driven execution
- Publish/subscribe composition
Optimization in run-time service composition

• Syntactic analysis
  – Graph analysis to remove duplications

• Semantic analysis
  – Analyze information content to determine reuse or sharing (e.g., sampling)

Vehicle counting services

Optimize out common components

Large vehicle detection services
Video: car driving

• As the vehicle drives through the testbed, the infrared break beams can detect its speed, length and direction.

• If the vehicle is very large, a magnetic signature can be acquired. If the vehicle is speeding or moving in the wrong direction, an image can be acquired.

• Users can query the system independently, viewing output such as occurrences of vehicles.
Capture an image of a speeding vehicle
Juggling Multiple Tasks

A blue vehicle initially spotted
Juggling Multiple Tasks

A red vehicle enters the scene

Bottom camera relinquishes blue vehicle to top camera, and starts to track red vehicle

Cameras successfully track both vehicles
  • Without resource allocation and re-tasking, both might have looked at blue vehicle, leaving no camera to track red vehicle
Sensor Tasking

• Unique to sensor net
  – Concurrent events, uncoordinated tasks, limited resources

• Tasking and re-tasking based on
  – Conflict resolution
  – Load balancing
  – Failure recovery
  – Cascading tasks
  – ...

• Need to be aware of the physical phenomena being monitored as well as system configuration
Manage both system and event uncertainty

A tracking example
As the event moves, one needs to move the code and/or state.
What if nodes fail?

• Contingency plan?
• How many copies do I send around, to maximize odds of not getting lost?
… and world knowledge is incomplete?

- Follow the clouds
- Probabilistic tasking?
- Proactive or reactive?
Centrally engineered, monitored, controlled

Decentralized Anarchy
(Implies that management decisions must be automatic in software, not a human’s intelligent judgment based on domain knowledge)

Digital, clean

Networked embedded systems

Physical, messy

Energy, bandwidth, attention

More resource constrained

Information uncertainty

System uncertainty

Server farms

Industrial control systems

Grid computing

Ubiquitous computing

Mesh networks

P2P networks

Mesh networks

Ubiquitous computing

Grid computing

Server farms

Industrial control systems
Visibility to systems

A spectrum allows high-visibility debugging before jumping into low-visibility deployment.

- Model-Based Simulation
- Trace-Driven Simulation
- Lab (“Indoor”) Testbeds
- In-Situ Testbeds
- Deployment

Scale

Reality

Easier to debug

Harder to debug

Source: Jeremy Elson
Data Collection

User Interface / Data Processing
(MS Excel)

SQL Query / Report
Raw Data + Processed Data

Archiving Events

DataBase
(MS Access / SQL Server 2005)

Gateway (MicroServer)

Sensor Net (Tmote Sky)

Visualize Events/ Process Data

Task Graph Configuration
Raw Data Streaming

Status / Sensor Readings
(TinyOS Packets)

XML packets

Raw Data Streaming

Archiving Events
<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td><img src="image" alt="Cold air from AC" /></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Mote (Id = 30)</td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Hot air from Laptop heat sink</td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td><img src="image" alt="Hot air from Laptop heat sink" /></td>
</tr>
<tr>
<td>9</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Mote (Id = 40)</td>
</tr>
<tr>
<td>11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
</tr>
</tbody>
</table>

- Cold air from AC
- Mote (Id = 30)
- Hot air from Laptop heat sink
- Mote (Id = 40)
\[ \text{Time} = -39.60 + 0.01 \times \text{Raw Data} \]

```
<oscopeMsg xmlns:xsd=http://www.w3.org/2001/XMLSchema
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <sourceMoteID>30</sourceMoteID>
  <lastSampleNumber>40550</lastSampleNumber>
  <channel>1</channel>
  <data1>6091</data1>
  <data2>6090</data2>
  <data3>6090</data3>
  <data4>6088</data4>
  <data5>6086</data5>
  <data6>6086</data6>
  <data7>6084</data7>
  <data8>6084</data8>
  <data9>6081</data9>
  <data10>6081</data10>
</oscopeMsg>
```
<table>
<thead>
<tr>
<th>Sample # (Mote 30)</th>
<th>Raw Data</th>
<th>°C</th>
<th>Sample # (Mote 40)</th>
<th>Raw Data</th>
<th>°C</th>
</tr>
</thead>
<tbody>
<tr>
<td>40550</td>
<td>6091</td>
<td>21.31</td>
<td>38550</td>
<td>8850</td>
<td>48.9</td>
</tr>
<tr>
<td>40551</td>
<td>6090</td>
<td>21.3</td>
<td>38551</td>
<td>8850</td>
<td>48.9</td>
</tr>
<tr>
<td>40552</td>
<td>6090</td>
<td>21.3</td>
<td>38552</td>
<td>8849</td>
<td>48.9</td>
</tr>
<tr>
<td>40553</td>
<td>6088</td>
<td>21.28</td>
<td>38553</td>
<td>8850</td>
<td>48.9</td>
</tr>
<tr>
<td>40554</td>
<td>6086</td>
<td>21.26</td>
<td>38554</td>
<td>8851</td>
<td>48.9</td>
</tr>
<tr>
<td>40555</td>
<td>6086</td>
<td>21.26</td>
<td>38555</td>
<td>8850</td>
<td>48.9</td>
</tr>
<tr>
<td>40556</td>
<td>6084</td>
<td>21.24</td>
<td>38556</td>
<td>8850</td>
<td>48.9</td>
</tr>
<tr>
<td>40557</td>
<td>6084</td>
<td>21.24</td>
<td>38557</td>
<td>8851</td>
<td>48.9</td>
</tr>
<tr>
<td>40558</td>
<td>6081</td>
<td>21.21</td>
<td>38558</td>
<td>8851</td>
<td>48.9</td>
</tr>
<tr>
<td>40559</td>
<td>6081</td>
<td>21.21</td>
<td>38559</td>
<td>8850</td>
<td>48.9</td>
</tr>
</tbody>
</table>
Narrow waist should allow applications to be specified independent of system configurations.

- Application Programming: Spreadsheet, logic/functional, visual, MATLAB, SQL
- System Run-Time Support: Mote programs, microserver programs, network protocols, …
- Sensor Net Deployment