EnviroTrack: Towards Environmentally Immersive Programming

Tarek Abdelzaher
Department of Computer Science
University of Illinois at Urbana Champaign

Real-Time and Embedded Systems
The Next Frontier

Networking Infrastructure
Emergency Response
Autonomic Computing
Privacy Security

Clusters, Farms, Grids, WWW
Embedded Systems
Embedded Everywhere
- Transparent
- Context-aware
- Mobile
- Miniature
- Ubiquitous
(Smart attire, smart spaces, ...)

Today
Sensor Networks

Applications
- Precision Agriculture
- Habitat Monitoring
- Disaster Response
- Target Tracking
- Infrastructure Protection
- Border Control

Features
- Ad hoc deployment
- Massive distribution
- Interaction with a physical environment
- Unattended operation

A Fundamental Challenge: Sensor Network Application Development

- Cost of sensor network software will dominate total system cost
- Example: U. Virginia’s VigilNet
  - Cost of hardware: $10K
    (100 nodes * $100/node at scale)
  - Cost of software development/debugging/testing: $120K
    (5 graduate students * 20 weeks * 40 hrs * $30/hr)
  - Hardware cost is decreasing but programmers’ cost isn’t
- Reducing software cost
  - Development cost: reusable components, high-level abstractions
  - Debugging cost: automated checking and analysis tools
  - Testing cost: realistic simulation/emulation environments
Reducing Development Cost: Distributed Programming Abstractions

- Distributed programming paradigms
- Abstract distributed communication
- Provide location transparency
Reducing Development Cost: Distributed Programming Abstractions

- Sensor Network Programming Abstractions
  - Represent the physical world to the programmer
  - Abstract distributed interaction with the physical environment

- Distributed programming paradigms
  - Abstract distributed communication
  - Provide location transparency
"Macro-programming": Sensor Network Programming Languages

- Node-based
  - NesC, MANTIS, Snack, TinyScript, ...
- Event-based
  - DSWare, SensorWare, GalsC/GuysC
- Group-based
  - Abstract regions, State-centric programming, Hood, ...
- Query-based
  - Cougar, TinyDB/TAG, TinySQL, Tina, ...
- Virtual machines
  - Mate, ASVM, ...
- Biological
  - Amorphous computing, swarm computing
- EnviroSuite: An object based system where objects represent environmental entities

Environmentally Immersive Programming

- Exports a new address space in which the addressed entities (called contexts) are representations of physical entities in the external environment

  **Contexts:**
  - Logical representations of entities in the external world
  - Have unique names (context labels) – same as IP hosts
  - Instantiated when the corresponding external entities are observed in the environment – follow these entities around
  - Tracking objects (tasks) can be attached to contexts to execute in the vicinity of the corresponding real-world entity

- Tasks (attached to contexts) can communicate and invoke each other’s methods remotely
Programming Model

Context Type: Car
Context Label
Tracking Objects
Aggregate State Variables

Context Type: Person
Context Label
Tracking Objects
Aggregate State Variables

Object Representation Protocol (Tracking Object)

External Entity

Contexts and Objects

- Contexts: Encapsulate entity state and tracking objects
- Tracking objects: Perform entity-specific computation, communication and sensing

Programmer’s View

Entity

State

History

ID
Communication

- Objects may export methods for remote invocation

```
begin context tracker
  sense: magnetic() + motion();
  state: location = avg (position, 3, 2);
end
```

Context Example
Attaching Objects

**begin context** tracker
sense: magnetic() + motion();
state: location = avg (position, 3, 2);
**end**

**begin object** reporter
send (state, home);
**end**

State$_e$( ) = average position
Context ($e$)
Programmer’s View
Entity
Members
Leader
Sense$_e$( ) = TRUE
ID
History

Attaching Objects

**begin context** tracker
sense: magnetic() + motion();
state: location = avg (position, 3, 2);
**end**

**begin object** reporter
send (state, home);
**end**

State$_e$( ) = average position
Context ($e$)
Programmer’s View
Entity
Members
Leader
Sense$_e$( ) = TRUE
ID
History

Attach
**Attaching Objects**

```
begin context tracker
  sense: magnetic() + motion();
  state: location = avg (position, 3, 2);
end

begin object reporter
  send (state, home);
end

begin object mic
  turn-on microphone
  send (sound, home);
end
```

State,() = average position

**Programmer’s View**

**Entity**

**Members**

**History**

**Leader**

**ID**

**Sense,() = TRUE**

**Aggregation and State Management**

**Application Objects**

- Timer
  - Send/receive
- Remote Invocation
- Leader
  - Aggregation Function
    - State()
  - Trigger Function
    - Sense()
- Member
  - Periodic Sensor Reports
    - Start/stop
  - Trigger Function
    - Sense()

**Group Management and Leader Election**
Summary:
1. Define objects and contexts

- Object Statistics
  - Count()
  - HandleAlarms()
  - GetHistory()

- Object Tracker
  - GetPosition()
  - MicOn()
  - MicOff()

- Object Alarm
  - DefineTrigger()
  - EnableAlarms()
  - DisableAlarms()

Context Type: Intruder

Context Type: Metal

Summary
2. Attach objects to contexts

- Object Statistics
  - Count()
  - HandleAlarms()
  - GetHistory()

- Object Tracker
  - GetPosition()
  - MicOn()
  - MicOff()

- Object Alarm
  - DefineTrigger()
  - EnableAlarms()
  - DisableAlarms()

Context Type: Intruder

Context Type: Metal

Tracker

Alarm

Stats
Programmer’s View

Context Type: Intruder

Object Statistics
Count()
HandleAlarms()
GetHistory()

Object Tracker
GetPosition()
MicOn()
MicOff()

Object Alarm
DefineTrigger()
EnableAlarms()
DisableAlarms()

Context Type: Metal

Tracker

Method Invocation

Stats

Example: Intrusion Detection Scenario

Deployed Sensor Network
Example: Intrusion Detection Scenario

Goal: monitor physical entities in the environment

Context 1. Type: Metal  
Context 2. Type: Intruder

Abstract representation: physical entities as logical contexts

Sensor field
Example: Intrusion Detection Scenario

Logical contexts can communicate, host computation, and have unique identifiers.

Context 1. Type: Metal
Context 2. Type: Intruder
Entity-to-entity Communication

Sensor field

EnviroSuite Walkthrough
II. Object Representation

Declarations

Context vehicle {
  sense = magnetic()
  state = AVG(position())
  method sendHome {
    send (base, state)
  }
}

Simulations/ Emulation

Specifications:
• communication delays
• data throughput
• lifetime
• …

Analysis Engine

Constraints

Feasibility Check

Configuration

Feedback

Application code
Comm. Subsystem
Storage Subsystem
Protocol Stack
Run-time Code
Object representation protocols
Tracking Objects
Region Objects
Self-Named Objects

Sensor data processing library

Object Representation Protocols

- Implement distributed algorithms for exporting the abstraction of physical objects
  - Unique representation
  - Identity management
  - Need well defined semantics
    - Crossing targets?
    - False positives/false negatives?
    - Spatial object boundaries?

- Object types
  - Tracking objects
  - Region objects
  - Self-named objects
Tracking Objects: Group Management Protocols

- Nodes sensing a given target form a single context
- Context has leader, members, and followers

Nodes that cannot sense the target but hear the leader (followers)

Nodes that can sense the target and hear the leader (members)

Node State Transition Diagram

New Leader Election

Leader

Member

Default

Follower

win

lose

handoff

sense stop

sense stop

receive leader heartbeat

timeout
Condition for Unique Representation

- Target can’t move fast enough to be sensed by a node that is outside leader heartbeat range:
- Communication range > 2 Sensing Range + Target speed * group migration time

Group management optimization 1: Semi-dynamic leader election

- Idea: allows only a subset of nodes to compete for leadership
  - Faster leader election
  - Fewer message collisions

- Algorithm:
  - Pre-elect potential leaders
  - Only potential leaders compete for leadership
Group management optimization 1: Semi-dynamic leader election

- Leader desert problem

Group management optimization 1: Piggy-backed heartbeat

- Observation:
  - Members periodically send data messages
  - Leaders periodically send heartbeats
- Idea: piggy-back leader heartbeats onto member messages
  - Less communication overhead
  - Better dissemination of object info.
Group management optimization 1: Piggy-backed heartbeat

- Overcoming the leader desert problem

\[ \text{Number of Objects} \\
\text{Target Velocity (grid/s)} \]

won't create spurious object

---

Group management optimization 1: Piggy-backed heartbeat

Maximum trackable speed

- EnviroSuite \( \leq 1 \text{ grid/s (22 mph)} \)
- Piggy-backed heartbeat \( \geq 8 \text{ grid/s (179 mph)} \)

\[ \text{Number of Objects} \]
\[ \text{Target Velocity (grid/s)} \]
Group management optimization 2:
Implicit leader election

- Observation:
  - Leaders periodically send data messages (target positions) to the base station
- Idea - reuse leader data messages as leader competition messages
  - For each period, the potential leader that first sends out a leader message becomes the leader
  - Others become silent in this period

Group management optimization 3:
Implicit leader election

<table>
<thead>
<tr>
<th></th>
<th>Maximum trackable speed</th>
<th>Number of control messages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Piggy-backed heartbeat</td>
<td>&gt;=8 grid/s (179 mph)</td>
<td>50 ~ 70</td>
</tr>
<tr>
<td>Implicit leader election</td>
<td>&gt;=8 grid/s (179 mph)</td>
<td>0</td>
</tr>
</tbody>
</table>
EnviroSuite Walkthrough
III. Analysis Tools

Declarations
Context vehicle {
  sense = magnetic()
  state = AVG(position())
  method sendHome {
    send (base, state)
  }
}

Simulation/Emulation

Analysis Engine

Specifications:
• communication delays
• data throughput
• lifetime
• ...

Analysis

Configuration

Feasibility Check

No?

Application code

Comm. Subsystem

Storage Subsystem

Protocol Stack

Sensor data processing library

Run-time Code

Object representation protocols
Tracking Objects
Region Objects
Self-Named Objects

Constraints

Sensor data processing library

Application code

Protocol Stack

Comm. Subsystem

Storage Subsystem

Run-time Code

Analysis

No?

Feedback
Aggregate Performance Tradeoffs in Sensor Networks

Analysis of Latency/Capacity Trade-offs Sensor Networks

- What is the relation between radio range, network density, total number of nodes, number of sinks, packet length, packet scheduling policy, packet throughput, MAC-layer protocol, and end-to-end packet ability to meet latency constraints?
Network Real-time Capacity

- Network bandwidth is the bottleneck
- Intuitively, network ability to meet latency requirements (schedulability) *decreases* with:
  - Increased packet size (task processing time), $C$
  - Increased distance between source and destination, $L$
  - Decreased end-to-end latency constraint, $D$
- Schedulability decreases with $CL/D$
- Is there a bound $Capacity_{RT}$, such that all packets, $i$, reach their destinations by their deadlines if:
  \[
  \sum_i \frac{C_i L_i}{D_i} \leq Capacity_{RT}
  \]

Multi-stage Resource Pipelines

- **The Stage Delay Theorem:** If the synthetic utilization of resource $j$, does not exceed $U_j$, then no task is queued on resource $j$ for more than a fraction $\beta_j$ of its end-to-end deadline\(^2\), where:
  \[
  \beta_j = \frac{U_j (1 - U_j/2)}{(1 - U_j)}
  \]

\(^2\)This is under deadline monotonic scheduling. Similar results derived for other policies.
The Single Hop Problem

- Only one node in the vicinity of the receiver can send a packet.

Equivalent Virtual Queue

Receiver’s Radio Range

Throughput Optimization: Maximizing Real-Time Capacity

- Consider a localized communication pattern where each node communicates with nodes at most N hops away.

- On any path, maximize $\sum_j U_j$ subject to
  $$\sum_{j=1}^N \frac{U_j(1-U_j/2)}{1-U_j} \leq 1$$

- From symmetry $U_j = U$
  $$\frac{U(1-U/2)}{1-U} = \frac{1}{N}$$

- Hence,
  $$U = \left(1 + \frac{1}{N} - \sqrt{1 + \left(\frac{1}{N}\right)^2}\right)$$
Real-Time Capacity

- **The total capacity theorem:** n - load-balanced
  net or of n nodes e-ch ith - r-dio of tr-nss ission
  speed W - nd m neigh ors on - er-ge if
  co nic-tion is qe-ized ithin t ost N hops

  \[
  \text{Capacity}_{opt} = \frac{nW}{m} \left(1 + \frac{1}{N} - \sqrt{1 + \left(\frac{1}{N}\right)^2}\right)
  \]

- For rge N

  \[
  \text{Capacity}_{opt} \approx \frac{nW}{mN}
  \]

Real-time Capacity of Multi-hop Data-Collection in Sensor Networks

- In a data collection sensor network with K collection
  points, maximum path length N, and radio transmission
  speed W, what is a sufficient bound on real-time
  capacity?
Real-time Capacity of Multi-hop Data-Collection in Sensor Networks

In a data collection sensor network with $K$ collection points, maximum path length $N$, and radio transmission speed $W$, what is a sufficient bound on real-time capacity?

$$\sum_{j \in \text{path}} \frac{U_j(1-U_j/2)}{1-U_j} \leq 1, \quad U_j \propto 1/j$$

Real-time Capacity of Multi-hop Data-Collection in Sensor Networks

In a data collection sensor network with $K$ collection points, maximum path length $N$, and radio transmission speed $W$, a sufficient bound on real-time capacity is:

$$\text{Capacity}_{DC} \approx \frac{KNW}{1+0.5\ln N}$$
The Capacity/Latency Trade-off

- The following sufficient condition quantifies a (conservative) view of the capacity/latency trade-off in a data collection network:

\[
\sum_i \frac{C_i L_i}{D_i} \leq \frac{KNW}{1 + 0.5 \ln N}
\]

System evaluation – empirical results

System introduction

- Application Layer:
  - Classification
  - Tracking
  - False Alarm Filtering Engine
  - Velocity Regression

- Middleware Layer:
  - Time Sync
  - Localization
  - Report Engine
  - Group Mgmt
  - Sentry Service
  - Tripwire Mgmt
  - Power Mgmt

- Network Layer:
  - Robust Diffusion Tree
  - Asymmetric Detection
  - Radio-Base Wakeup
  - Frequency Filter
  - Continuous Calibrator

- Data Link Layer:
  - Interference Avoidance
  - Sensor Drivers

- eXtreme Scale Motes (XSM)
System evaluation – empirical results

Deployment

- Deployment – 200 XSMs (20% node failures)
  - potential leaders
  - base station
  - other nodes

![Image of deployment with Base Station labeled]

System evaluation – empirical results

Tracking performance

<table>
<thead>
<tr>
<th>Target type</th>
<th>Avg. tracking error</th>
<th>Calculated velocity</th>
</tr>
</thead>
<tbody>
<tr>
<td>walking person (3±1 mph)</td>
<td>6.19 meter</td>
<td>2.9 mph</td>
</tr>
<tr>
<td>running person (7±1 mph)</td>
<td>6.67 meter</td>
<td>6.9 mph</td>
</tr>
<tr>
<td>Vehicle (10±1 mph)</td>
<td>7.06 meter</td>
<td>10.5 mph</td>
</tr>
<tr>
<td>Vehicle (20±1 mph)</td>
<td>5.91 meter</td>
<td>23.5 mph</td>
</tr>
</tbody>
</table>
Summary and Conclusions

- High-level programming abstractions are needed for sensor network applications.
- Underlying foundations must be found for analysis of spatio-temporal behavior.
- EnviroSuite attempts to address the above challenges:
  - Ensuring correct representation of the environment (e.g., correct association of measurements and targets).
  - Providing analysis tools.
- General question: what services and analysis tools are most important for reducing the cost of development, validation, testing, deployment, and robust execution of sensor networks software?

Selected Publications

- Tian He, Sudha Krishnamurthy, Liqian Luo, Ting Yan, Lin Gu, Radu Stoleru, Gang Zhou, Qing Cao, Pascal Vicaire, John A. Stankovic, Tarek F. Abdelzaher, Jonathan Hui, Bruce Krogh, "VigilNet: An Integrated Sensor Network System for Energy-Efficient Surveillance," accepted to *ACM Transactions on Sensor Networks (ACM ToSN '05)*