Brief History

- The early days (’99 - ’03)
  - Smartdust, Berkeley motes
  - TinyOS, Directed Diffusion
  - Themes: thousands of *homogeneous* devices operating *autonomously* in *remote* environments

- The near future (’08 - ?):
  - New hardware: motes, iPhones, laptops
  - New environments: urban, personal
  - New users: “citizen scientists”
  - Themes: *heterogeneous*, mobile devices operating *interactively* in *everyday* environments
Brief History

- The early days ('99 - '03)
  - Smartdust, Berkeley motes
  - TinyOS, Directed Diffusion
  - Themes: thousands of homogeneous devices operating autonomously in remote environments

- The near future ('08 - ?):
  - New hardware: motes, iPhones, laptops
  - New environments: urban, personal
  - New users: “citizen scientists”
  - Themes: heterogeneous, mobile devices operating interactively in everyday environments
Applications and Challenges

- Applications
  - Environmental monitoring
Applications and Challenges

• Applications
  – Environmental monitoring
  – Disaster management
Applications and Challenges

• Applications
  – Environmental monitoring
  – Disaster management
  – Health care
Applications and Challenges

- Applications
  - Environmental monitoring
  - Disaster management
  - Health care

- What are the research challenges?
  - Privacy
  - Middleware
  - End user interfaces
  - System integration
Applications and Challenges

• Applications
  – Environmental monitoring
  – Disaster management
  – Health care

• What are the research challenges?
  – Privacy
  – Middleware
  – End user interfaces
    • How will users interact with these devices?
  – System integration
    • How will sensor networks integrate with existing software infrastructures?
End-User Interfaces

- Application developers face enormous challenges
- Most sensor network users are *not* developers
  - Domain specialists, ordinary users
- How are typical users expected to use sensor networks effectively?
  - Trained specialists (expensive)
End-User Interfaces

- Application developers face enormous challenges
- Most sensor network users are not developers
  - Domain specialists, ordinary users
- How are typical users expected to use sensor networks effectively?
  - Trained specialists (expensive)
  - Pre-built applications (inflexible)
End-User Interfaces

• Application developers face enormous challenges
• Most sensor network users are *not* developers
  - Domain specialists, ordinary users
• How are typical users expected to use sensor networks effectively?
  - Trained specialists (expensive)
  - Pre-built applications (inflexible)
  - *Co-opting familiar interfaces*
    - Allow domain specialists to *create* their own applications in a friendly development environment
    - Flexible and inexpensive
Tables

- Table-based language environment for sensor networks
- Use familiar spreadsheet metaphor to *program* the sensor network
  - Spreadsheets are familiar
  - Provides a data-organized view
  - Encourages iterative design
- Data comes from sensor networks
- Functions execute on the sensor nodes
Tables Workflow

- Sensors are continually collecting and storing data
- Users specify what data to view
  - Using a pivot table
- Spreadsheet is populated with data from the sensor network
- User optionally specifies functions to operate on that data
  - This function is executed on the sensor network
- User requests updated data
Pivot Tables

- Miniature representation of the spreadsheet
  - Method to query the sensor network and organize the data
  - Users click-and-drag items to the data and metadata panes
- Data: data the user wants to view
- Row, Column: spatially organize the data
- Sheet: organize the data into logical sensor groups
- Sensor list is updated automatically as new types of data are generated
Pivot Tables

- Pivot table is compiled and propagated to the sensor network
- Each sensor node responds to pivot table requests
  - Transmits relevant data queue along with requested metadata
- Tables combine data from the sensor network with the pivot table to construct a final view
Local Functions

- Users need a way to do more than simply *viewing* the data
  - Example: filtering noise
- Users type in spreadsheet-like functions in empty cells
  - Functions execute whenever there is new data
Local Functions

- Users need a way to do more than simply *viewing* the data
  - Example: filtering noise
- Users type in spreadsheet-like functions in empty cells
  - Functions execute whenever there is new data
  - Arithmetic, sum, average, conditional, assignment

Local Functions:

(1)
```plaintext
-PSLOPE := 0
-TSLOPE := 0
-if( |TSLOPE - slope(2, Thermistor)| > 2 ) TSLOPE := slope(2, Thermistor)
-if( |PSLOPE - slope(2, Photometer)| > 2 ) PSLOPE := slope(2, Photometer)
```

Collective Functions: (None)

Pivot Tables:

(2) Data = PSLOPE, TSLOPE : metadata = Node ID, Time, Sensor Type
Local Functions

- Users need a way to do more than simply viewing the data
  - Example: filtering noise
- Users type in spreadsheet-like functions in empty cells
  - Functions execute whenever there is new data
    - Arithmetic, sum, average, conditional, assignment
- Functions are propagated and executed in sensor network
  - Nodes are defined by the sheet
Collective Functions

- Local functions aren't enough
- Many interesting computation requires collecting data from many nodes
  - Regional averages
  - Centroid calculations
- How to do this without explicit communication?
  - Use the sheet dimension to organize the network into logical groups
Collective Functions

- **Sheet**: Sensor Type
  - Produces two sheets: *thermistor* and *photometer*
  - All nodes with a *thermistor* form a *logical group*
  - Each group assigns a leader
- The group leader sends a publication request
  - Nodes publish new data
  - Collective functions are executed with new data
- Nodes can join and leave the group
Collective Functions

- Example: centroid calculation
- Define set of local functions
  - Checks proximity value of magnetometer
  - Publishes weighted sensor value

Local Functions:
(1)
=DETECTION := 0
=DISTANCE := 0
=if( Magnetometer > 0 ) DETECTION := 1 else DETECTION := 0
=if( DETECTION == 1 && Magnetometer > 7 ) PUBLISH := 1 else PUBLISH := 0
=if( PUBLISH == 1 ) WY := Magnetometer * YLoc
WXY := Magnetometer * XLoc
SENSOR := Magnetometer

Collective Functions:
(3)
CentroidX := sum(ALL, WX) / sum(ALL, SENSOR)
CentroidY := sum(ALL, WY) / sum(ALL, SENSOR)

Pivot Tables:
(2) Data = Node ID : Metadata = DETECTION
Collective Functions

- Example: centroid calculation
- Define set of local functions
  - Checks proximity value of magnetometer
  - Publishes weighted sensor value
- Create pivot table to view data from proximate sensors

Local Functions:
(1)
=DETECTION := 0
=DISTANCE := 0
=if( Magnetometer > 0 ) DETECTION := 1 else DETECTION := 0
=if( DETECTION == 1 &&
    Magnetometer > 7 ) PUBLISH := 1 else PUBLISH := 0
=if( PUBLISH == 1 ) WY := Magnetometer * YLoc
   WX := Magnetometer * XLoc
   SENSOR := Magnetometer

Collective Functions:
(3)
CentroidX := sum(ALL, WX) / sum(ALL, SENSOR)
CentroidY := sum(ALL, WY) / sum(ALL, SENSOR)

Pivot Tables:
(2) Data = Node ID : Metadata = DETECTION
Collective Functions

- Example: centroid calculation
- Define set of local functions
  - Checks proximity value of magnetometer
  - Publishes weighted sensor value
- Create pivot table to view data from proximate sensors
- Define functions to average weighted sensor values from proximate sensors

Local Functions:
(1)
=DETECTION := 0
=DISTANCE := 0
=if( Magnetometer > 0 ) DETECTION := 1 else DETECTION := 0
=if( DETECTION == 1 &
   Magnetometer > 7 ) PUBLISH := 1 else PUBLISH := 0
=if( PUBLISH == 1 ) WX := Magnetometer * YLoc
   WY := Magnetometer * XLoc
   SENSOR := Magnetometer

Collective Functions:
(3)
CentroidX := sum(ALL, WX) / sum(ALL, SENSOR)
CentroidY := sum(ALL, WY) / sum(ALL, SENSOR)

Pivot Tables:
(2) Data = Node ID : Metadata = DETECTION
Collective Functions

- Example: centroid calculation
- Define set of local functions
  - Checks proximity value of magnetometer
  - Publishes weighted sensor value
- Create pivot table to view data from proximate sensors
- Define functions to average weighted sensor values from proximate sensors
- Construct a pivot table view the averaged locations

Local Functions:
(1)
=DETECTION := 0
=DISTANCE := 0
=if( Magnetometer > 0 ) DETECTION := 1 else DETECTION := 0
=if( DETECTION == 1 && Magnetometer > 7 ) PUBLISH := 1 else PUBLISH := 0
=if( PUBLISH == 1 ) WY := Magnetometer * YLoc
   WX := Magnetometer * XLoc
   SENSOR := Magnetometer

Collective Functions:
(3)
CentroidX := sum(ALL, WX) / sum(ALL, SENSOR)
CentroidY := sum(ALL, WY) / sum(ALL, SENSOR)

Pivot Tables:
(2) Data = Node ID : Metadata = DETECTION
Current Status

- Tables written in Java
- Sensor nodes equipped with Mantis OS
- Collection Tree Protocol for routing
- Kensho group management middleware
- Exploring collective group implementations
  - Simple strategy: elect the basestation as the leader
  - Better strategy: elect a local leader
Users want to integrate with existing programming and management tools.

Lots of existing software that could do interesting things with sensor data (databases, grid and internet services, HPC, games).

How do we integrate?

- Rewrite software for sensor networks (expensive, difficult)
- Reuse data exchange formats (may still require rewriting software)
- Adapt filesystem metaphor
  - Most users and software understand this metaphor
  - Nodes appear as directories
  - Data appear as files
Potential Applications

- Use filesystem commands to manage applications
  - View data using 'ls' and 'cat'
  - Task program images and data using 'cp'
Potential Applications

- Use filesystem commands to *manage* applications
  - View data using 'ls' and 'cat'
  - Task program images and data using 'cp'
- Prototype applications
  - Use Matlab to quickly script applications

```matlab
function readsensors(path)
    while(true)
        sensor_light = dlmread(strcat(path, '/s0/PHOTO'))
        plot(sensor_light, 'mo')
        pause(2)
    end
end
```
Potential Applications

- Use filesystem commands to *manage* applications
  - View data using *'ls'* and *'cat'*
  - Task program images and data using *'cp'*

- Prototype applications
  - Use Matlab to quickly script applications

- Co-opt existing programs for sensor networks
  - File servers
A single filesystem interface is not enough

- Different applications require different file semantics

Need a framework to create different filesystem interfaces for sensor networks

- Should handle communication between the sensor network and filesystem server
- API should allow developers to optimize the filesystem for latency and memory
- Should be simple to integrate into existing applications
FUSN Architecture

• Tiered architecture:
• Each sensor node runs local FUSN runtime
  – Minimal processing
• Filesystem server
  – Presents filesystem interface using FUSE
  – Export additional services (ssh, NFS)
• End-user client
  – Mounts multiple filesystem server over internet
FUSN Architecture

- Tiered API

- Processing functions
  - Pre-processors transform data before transmission
    - sanitizing input
  - Post-processors transform data before presentation
    - create a text file

- Data functions
  - Supplies data and responds to new data
  - Not restricted to raw sensor values

Processing Functions
- run_length_encode
- run_length_decode
- comma_delimited_decode

Data Functions
- list_data
- read_data
- write_data

FUSE

USB

CTP
Example Filesystems

• **Basic Filesystem**
  - Each sensor device exported as a file
    • Uses post-processor to transform data into comma-delimited text file
  - Each LED device exported and controlled by a file
    • on, off, and toggle commands

• **Tasking Filesystem**
  - Provides a simple, tasking library
    • Uses pre- and post-processors to parse the text
  - Free debugging

<table>
<thead>
<tr>
<th>Task Command</th>
<th>Text Size</th>
<th>Binary Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>sample(TIME, A, 1, 2048)</td>
<td>22 bytes</td>
<td>14 bytes</td>
</tr>
<tr>
<td>classify(A, B, ABOVE, 32)</td>
<td>25 bytes</td>
<td>15 bytes</td>
</tr>
<tr>
<td>stat(B, C, AVG, 3)</td>
<td>19 bytes</td>
<td>13 bytes</td>
</tr>
<tr>
<td>stamp(MEM, C, D)</td>
<td>16 bytes</td>
<td>11 bytes</td>
</tr>
</tbody>
</table>
Implemented using Mantis OS

Use runlength processors to reduce memory consumption and message transmissions

File Latency
  - Less than 1 second for most common commands
    - Over USB, CTP has ~ 25% increase

Number of messages
  - Pack as many data values into a packet as possible (60 bytes)

Memory consumption
  - A few hundred bytes for 250 values with rl encoding
Summary and Open Questions

• Future class of sensor networks
  – Privacy, programming, and integration challenges

• Tables
  – View data easily and program complex applications

• FUSN
  – Treat the sensor network as a filesystem

• How do we model heterogeneous platforms?
• How do we integrate this model into a suitable middleware?
• How do we integrate user policies (privacy and application specific behavior) into middleware?
Acknowledgments

- Collaborators: Eric Nelson, Michael Groat
- Post-docs: Jean-Charles Tournier, Patrick Widener
- Committee: Arthur B. Maccabe, Stephanie Forrest, Patrick Bridges
- Funding: NNSA, Los Alamos National Lab
- Scalable Systems Lab
- Adaptive Computation Lab
- http://cs.unm.edu/~jhorey
Privacy and Security

- Deployments involving humans will have varying degrees of confidentiality restrictions
  - Must provide new methods for privacy and security
  - Must be *efficient* and *effective*

- Example: Traffic monitoring - Want to know the distribution of speeds
  - Each car has a speed sensor
  - Drivers not willing to send incriminating data

[Horey et al, Mobiquitous '07]
Privacy and Security

- Assume that all sensors have the same set of possible values.
- Sensors randomly choose a value not sensed.
- Basestation reconstructs the histogram of original values.
- Currently extending to include location anonymity.

\[ A_i = n - (t - 1) R_i \]
Privacy and Security

- What if the histogram is insufficient?
  - Store a “negative” database
- Basestation can ask certain questions efficiently
  - Is this data item in the database?
- Basestation can *not* ask for a complete list of data
  - NP-Complete