Attribute-Based Routing in Sensor Networks

Dec 9, 2005

Thomas Little

In collaboration with Ashish Agarwal and Andrew Ke

This material is based on work supported by the national science foundation under grant No. CNS-0435353.

Overview of the Talk

- A case for attribute-based routing by showing how it **enables** the dominant communication patterns of a SNET
  - Rooted sink trees (e.g., directed diffusion)
  - Queries on a sensor field (e.g. TinyDB)
  - Aggregation of data
  - Localized, in-network processing (general case of aggregation)
  - Event triggering

- An **attribute labeling scheme** based on XML

- Preliminary ideas for a **tasking scheme** that instantiates the mechanisms above using the labeling scheme

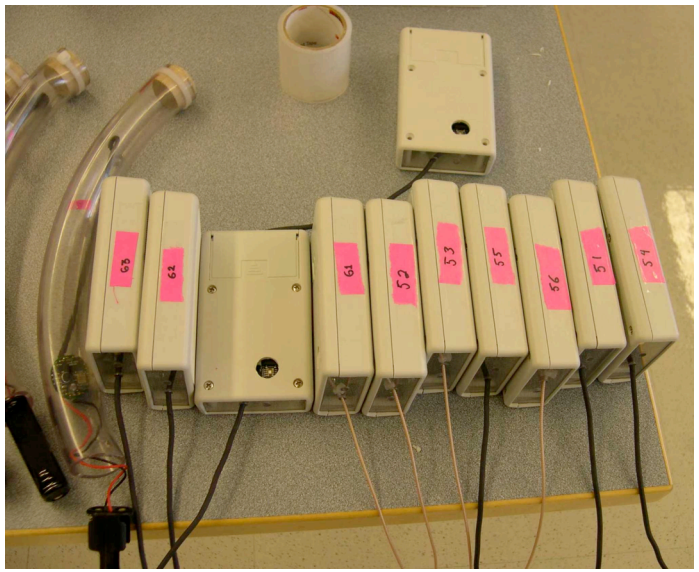
- And the details that connect the pieces above

Outline

- Characteristics of SNETs
- Routing
  - General
  - Data-centric routing
  - SNET
- Motivating examples
- Attribute labeling
- Tasking and task instantiation
- Research thrusts and active plans
- Relationship to Nets-Noss grant

Sensor Networks

- Many applications
- Characterized by
  - Many small nodes
  - Resource constrained
  - Wireless, battery powered
  - Sense some phenomenon
  - Report or act on sensed data
- Example: Bird and bat fatalities in wind farms
- HVAC control systems in buildings
- Animal tracking
- Security systems
- Agriculture -- irrigation control
- Vehicular warning functions
- Parking spot occupancy



Specific Challenges and Goals

Challenges

- Complexity and scale -- many nodes
- Batteries, node failure, replenishment, maintenance
- Limited resources
- Evolving missions

Goals

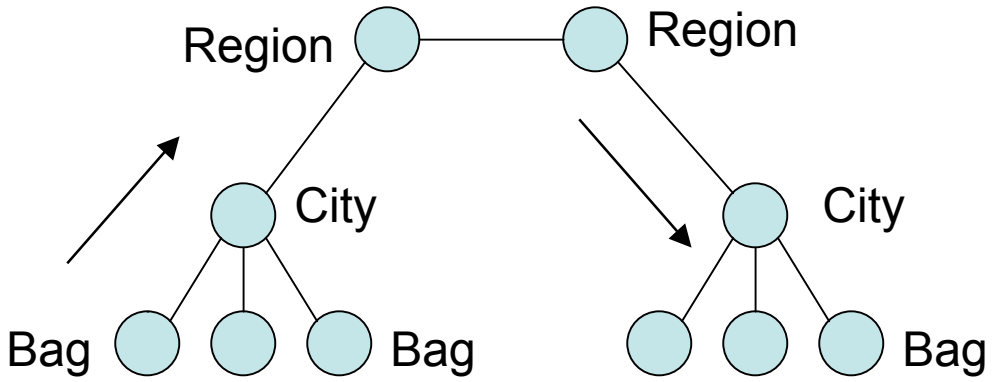
- Adaptability
- Reconfigurable -- retaskable
- Easy to program
- Efficiency but as a secondary goal
  - Must operate comparably to existing schemes
  - Great if meet or exceed current energy use profiles

Routing -- General

Routing is about getting a package (packet) from a source to a destination through multiple through intermediate stops (hops)

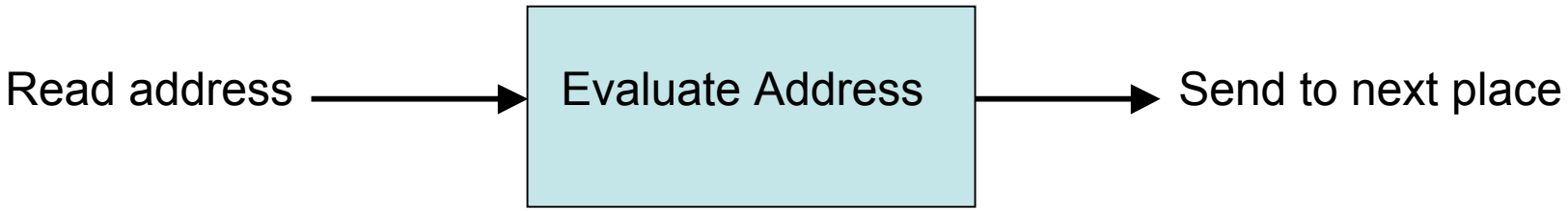
Examples

- Send a package by US Mail
  - Addressing: Name, street address, city, state
  - Pickup at house --> city post office --> regional mail center --> regional mail center --> city post office --> mail bag --> house



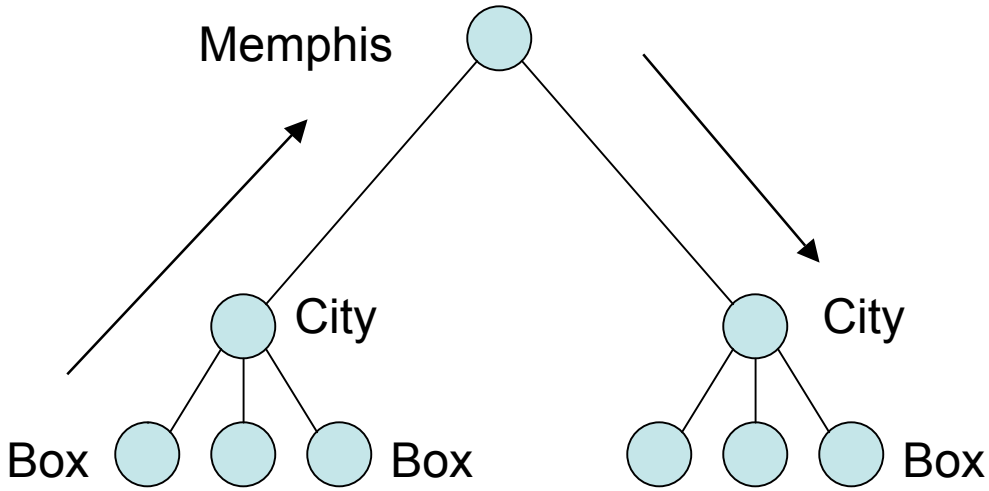
Routing cont.

- **It's hierarchical** and relies on an preexisting delivery routes
  - Outbound: up the hierarchy
  - Inbound: down the hierarchy
- Each step evaluates the address appropriate for that level
- Each level only requires info about it's peers -- routing information is distributed according to hierarchy



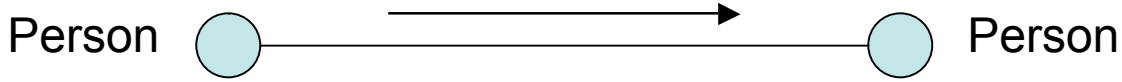
Routing cont.

- Send a package via FedEx
- Same addressing scheme but
  - Send it all to Memphis tonight
  - Send it to regional distribution centers and destinations tomorrow
- Still hierarchical, but skips some intermediate steps

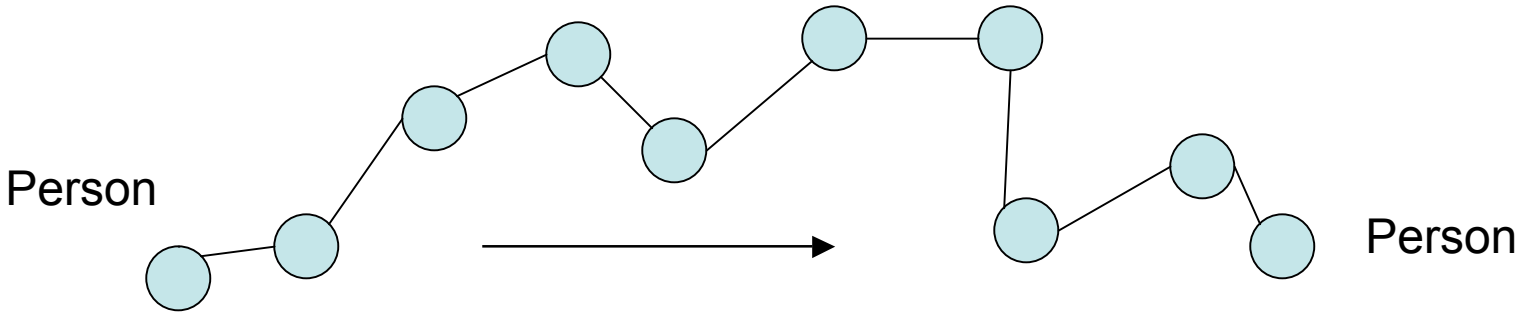


Routing cont.

- Send a package by courier
  - Point-to-point from origin to destination
  - Knowledge of courier skips all hierarchy
  - Courier has a complete picture of all destinations



- Pass a note to a friend in the audience
  - Destination is specified and follows the message
  - “Flat routed” and “single hop”
  - Everyone can easily find their neighbor



Routing cont.

- Each example has
  - Explicit address
  - Addressing scheme
  - Some rules for what to do at any given step (routing rules)
  - Packages! -- We need the people (applications) who what to send packages (messages)

- We also need ways to keep the routing information up to date
  - When a new airport opens up, or a new town is created

- None of these schemes care about package contents (bits)

Routes

- Routes can be represented by a sequence of hops.
  - Origin-->A-->B-->D-->E-->Destination
  - Sometimes used this way
  - Or can deposit this information at each node: next hop information for any specific destination

- Result: a routing table

- FedEx example -- routing table
  - If a package is received, use the destination state to select the next hop

Boston Table

Destination	Next Hop
San Fran	Memphis
Seattle	Memphis
Chicago	Memphis
Boston	Boston

Routes

- FedEx example -- routing table

Memphis Table

Dest	**Next Hop**
San Fran	San Fran
Seattle	Seattle
Chicago	Chicago
Boston	Boston

- Note:
  - The table is different at each location
  - The way the routing table is used is invariant -- just different entries
  - **It's simple**

Summary for Routing Design

- The building blocks
  - Addressing scheme and addresses
  - Routing rules
    - The table structure
    - How the table is applied
  - How the table is maintained
    - How and when to put new destinations in the table
  - How much information should be localized or distributed (flat vs hierarchical)

- Some other routing variants
  - Generate the route information on the fly -- by discovering them when you need them (**reactive** technique) -- “route discovery”
  - Generate the route information in advance (**proactive** technique) -- “bootstrapping” the routes
  - Dealing with added nodes or ones that move or fail (or are malicious) -- “**route maintenance**”

Should we Examine the Package?

- In some cases the **contents** of the package being routed are relevant
  - All heavy items go by truck (an attribute of the package)
  - Priority letters go by plane (type information)
  - Perishable items go in a refrigerated container -- all are put in same box
  - Wine can only go to licensed distribution centers
- In a SNET, this concept can be exploited
  - “Data-centric” -- use knowledge of the data to direct messages
  - Label the data in the messages so that they are not just a bitstream
  - Send types, values, or complex predicates on contained data to appropriate sinks
- Example:
  - Suppose you are concerned about freezing temperatures in a warehouse
  - “Send all temperature readings T < 2 degC to the base station”
  - Need type, value, predicate to evaluate a message and determine to forward -- these need to be discernable in the message body
- Attribute-Based Routing!

How to Use this New Information?

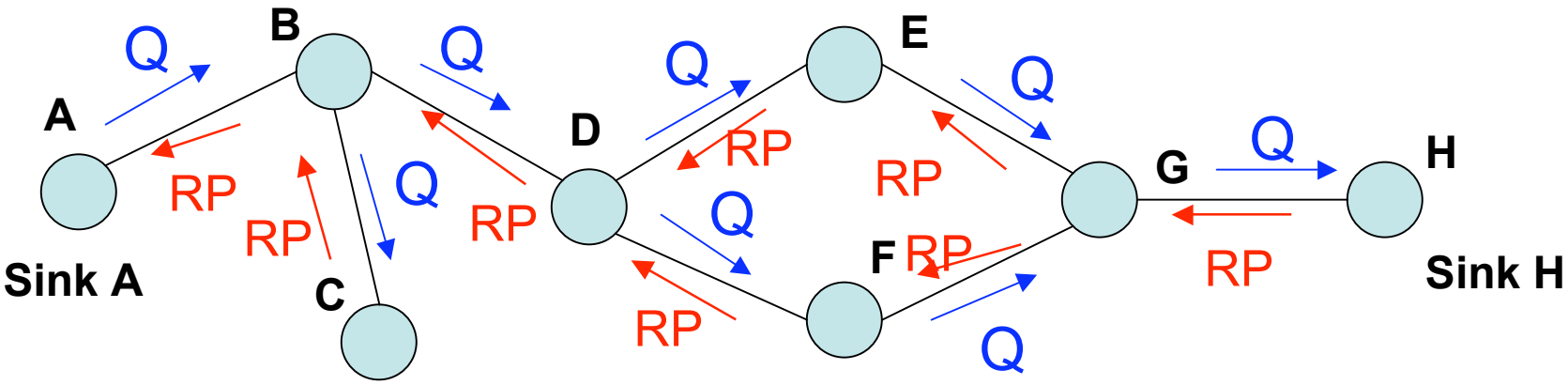
- Some key questions arise
  - How are the data labeled? Are the labels in a node's vocabulary?
  - How will a type, value, predicate be evaluated (how are the routing rules and routing table applied?)
  - How will an arbitrary node know where a destination is? (How to bootstrap the routing information?)
  - How do nodes comprising the SNET get "programmed" to use the labeled data?
  - How to mix (not to mix) application and routing?
- Is not: active networks
- Is: inspection of message contents by intermediate nodes for routing
- Next: some examples of how this is achieved using existing schemes:
  - Directed Diffusion
  - TinyDB
  - Aggregation
  - Abstract regions
- Approach: build up from existing SNET communication models: 4 cases

Case 1: Directed Diffusion

A routing scheme for SNETs

- The network starts with no routing information
- A sink generates a request (“interest”) for some data type (e.g., temperature)
- Unique “interest ID” Q
- The interest is flooded to the network

Directed Diffusion Illustrated



- As the query propagates, reverse paths to the sink are formed
  - Residue is left in the routing table by the propagating query
  - Table entries: [Q, next hop towards sink]
  - Duplicate messages from flooding are dropped
- When the query lands on a sensor that matches the interest, values are subsequently routed along the multihop path back to the sink
- Future queries create additional entries to the routing table
- Predicate can be embedded in the query and the results propagated on Q (but the predicate is evaluated at the endpoint)

Case 2: TinyDB

TinyDB treats the SNET as a distributed database

- A TinyDB-enabled SNET can be queried like a database
- The universe of sensed values form a set of columns in a table

  [nodeID | temp | pressure | light | lat | lon | time | moisture |…]

- Queries are for specific attributes on the table recorded at intervals in the future. Example:

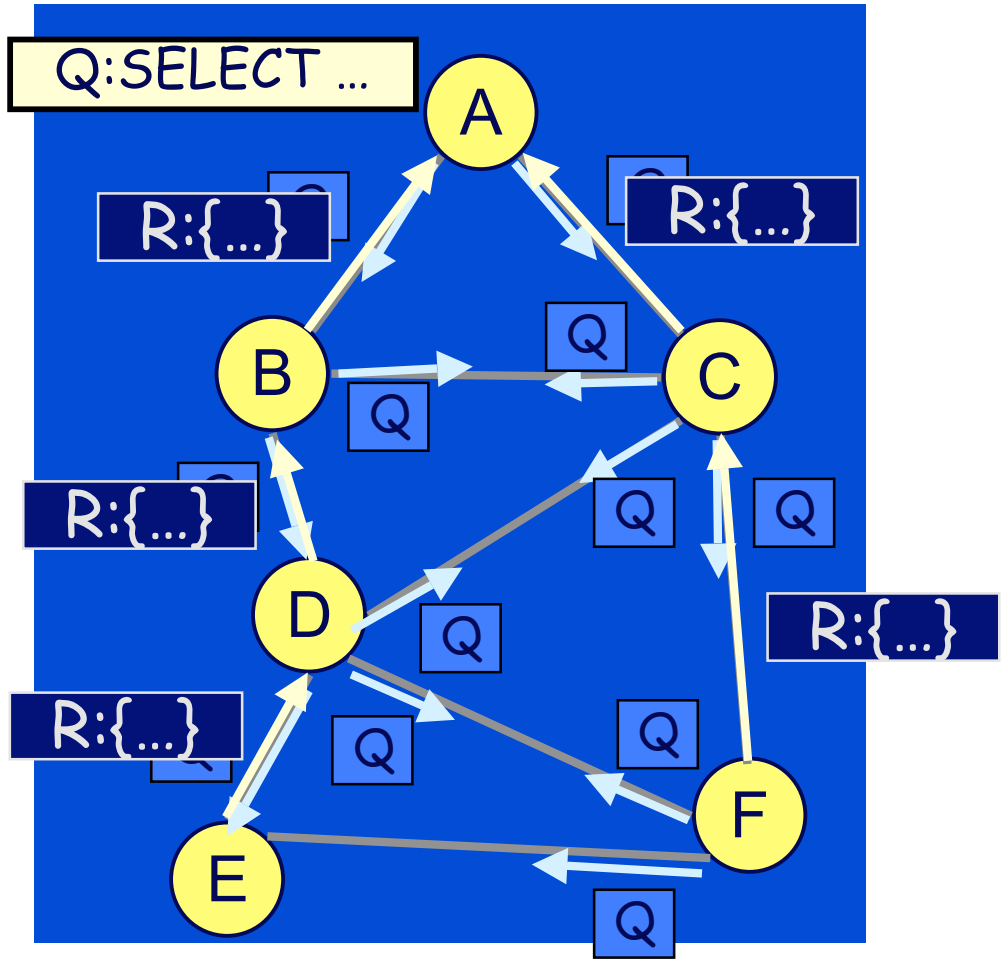
```
SELECT
Nodeid, light,temp FROM
Sensors WHERE
Temp < 2
SAMPLE PERIOD 10240
```

TinyDB cont.

- Very similar propagation scheme to DD
  - The network starts with no routing information
  - A sink generates a query for selected fields in the table (e.g., temp)
  - The query has a unique ID
- The query is flooded to the network
  - As the query propagates, reverse paths to the sink are formed
    - Residue is left in the routing table by the propagating query
    - Table entries: [QueryID, next hop towards sink]
    - Duplicate flood messages are dropped
- The query lands on all nodes
  - The future results are propagated back to the sink via the sink-rooted spanning tree
  - Future queries from the same sink can use the same rooted tree
  - Additional sinks require additional rooted trees
  - Possible to do aggregation and optimization of the DB functions by using the formed tree
- Note: DB predicates are instantiated in nodes for sourcing data but not for routing (e.g., T < 2)

TinyDB Example

SELECT
Nodeid, light,temp
FROM
Sensors WHERE
Temp < 2
SAMPLE PERIOD 10240



Animation from Hong and Madden

Limits of TinyDB

- Does:
  - Uses attributes
  - Aggregation (in-network)
  - Support multiple concurrent queries
  - Provides a data retrieval metaphor
  - Simplify programming substantially
  - Reduce energy consumption by limiting data transmission to data sought based on predicates
- Does not:
  - Eliminate duplicate propagation from multiple queries
  - In-network propagation (routing) of results for consensus or distributed estimation techniques
  - Support routing in general classes
  - Support dynamically changing routing except by killing query and restarting
  - Support tasking that deviates from a query metaphor
  - Support tasks that are not rooted in sink

Case 3: Aggregation and In-Network Processing

- Concept:
  - Employ nodes in the system to synthesize results from sensed data
  - Per-node or collaborative
  - Aggregation operations: Min, Max, Count, Avg, etc.
  - Target classification (e.g., identify bird sound)
- How to 'program' such a system?
  - Monolithic application -- application layer
  - As a built-in capability of the nodes -- predefined functions
  - Using TinyDB -- for database aggregate operations (predefined)
  - Heavyweight: use mobile agents…
- But, these potentially lack flexibility and reconfigurability

- How can attributes attribute-based routing help here?

- Can we ask each node to interpret the contents of messages?

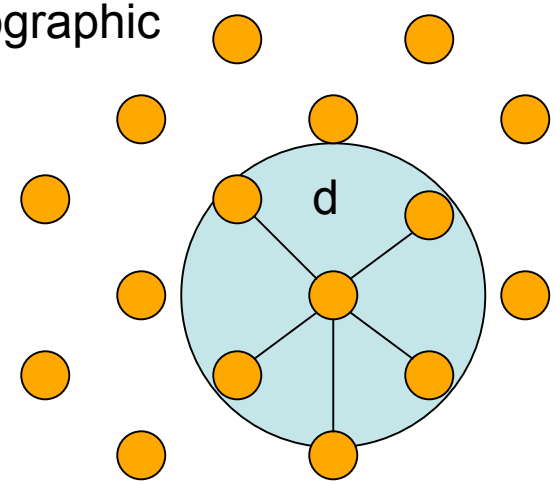
Case 4: Abstract Regions

- Concept:
  - Local communication structures are useful and efficient
    - Where spatial phenomena are sensed
  - Communication structures include:
    - Geographic range (circle)
    - Planar mesh
    - Spanning tree
- Embed functions to allow these structures to be created at any node for
  - Tracking
  - Contour mapping
  - Directed diffusion
  - GPSR functions
- But abstract regions imply neighbor or location-specific attributes

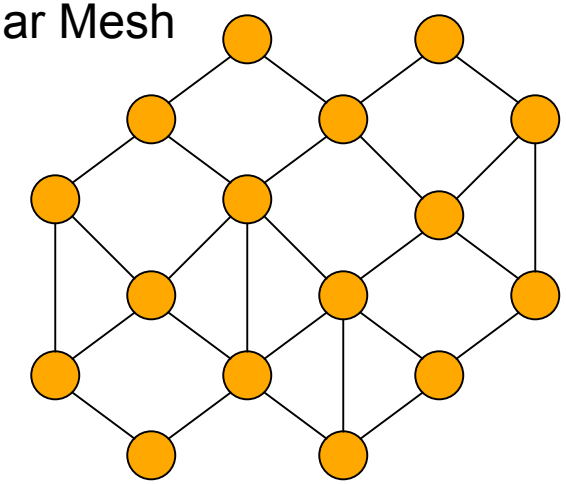
- Attribute-based routing should accommodate these scenarios plus more general cases -- attributes not tied to locality

Abstract Regions

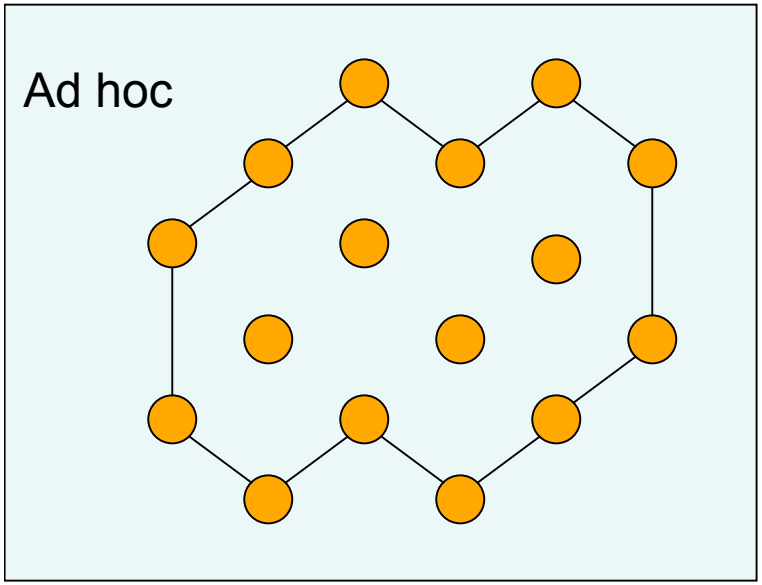
Geographic



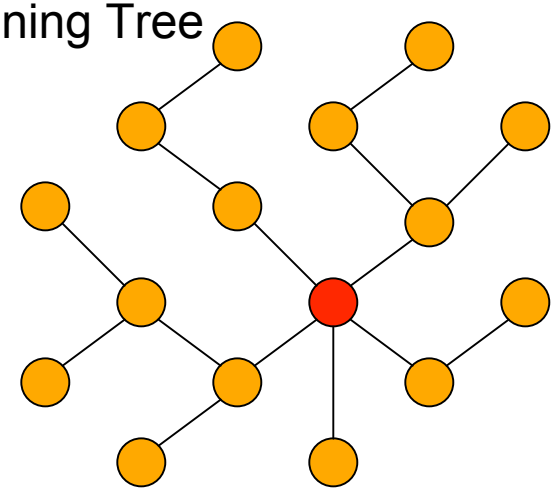
Planar Mesh



Ad hoc



Spanning Tree



Some Motivating Examples

Scenario 1: Optimized Irrigation

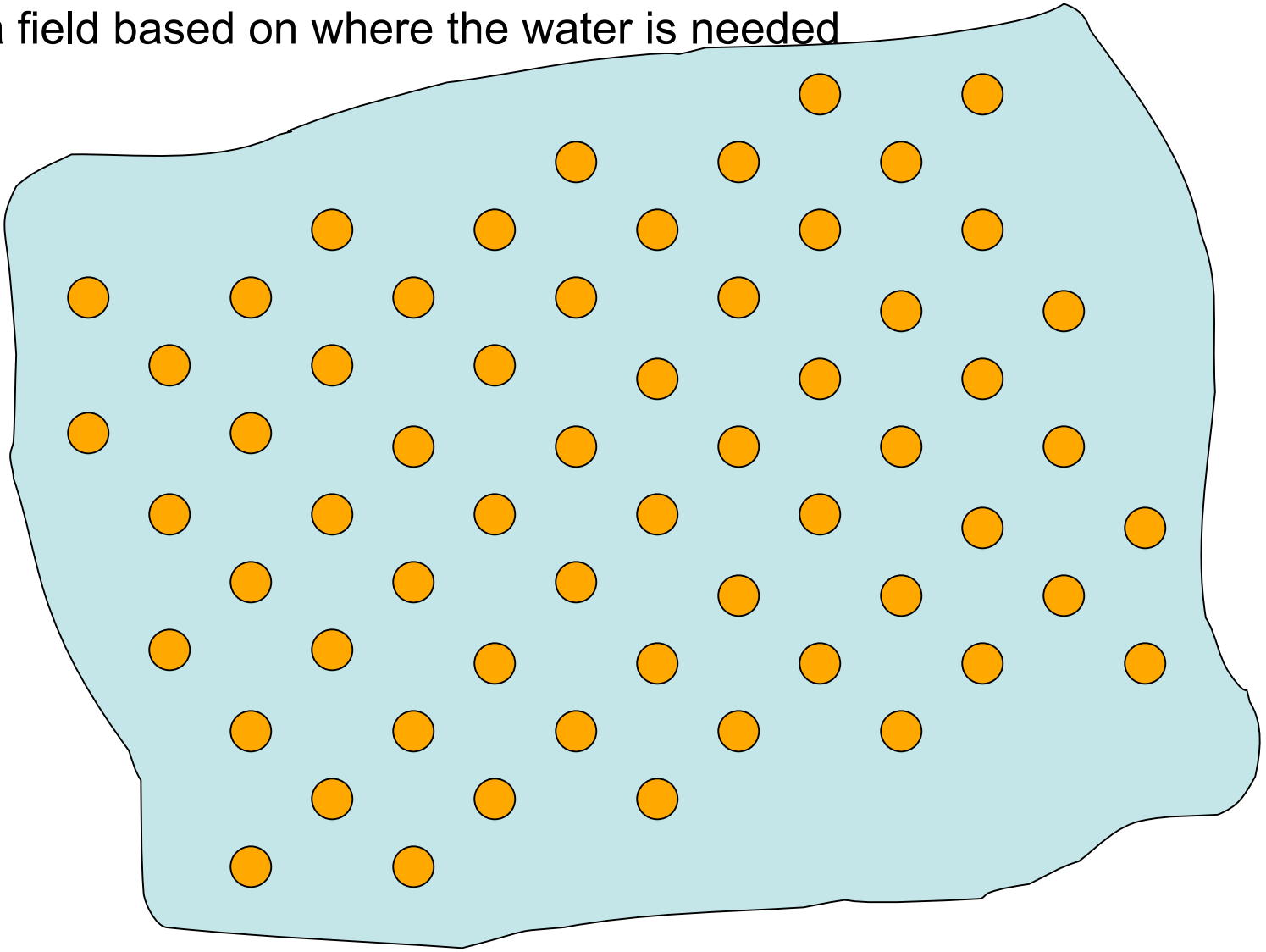
- Strategy: measure soil moisture at a fine spatial scale to better target irrigation
- Motes should exchange measurements to compute local mean Mi.
- When Mi < threshold, turn the water on in region i.
- Report back to base the expenditure of water

- Characteristics:
  - Local control loop
  - Little or no need to report to gateway
  - Local routing

- Possible to formulate a variant in which heterogeneous nodes (sprinklers and sensors) form different subsets and AR does not appear to apply

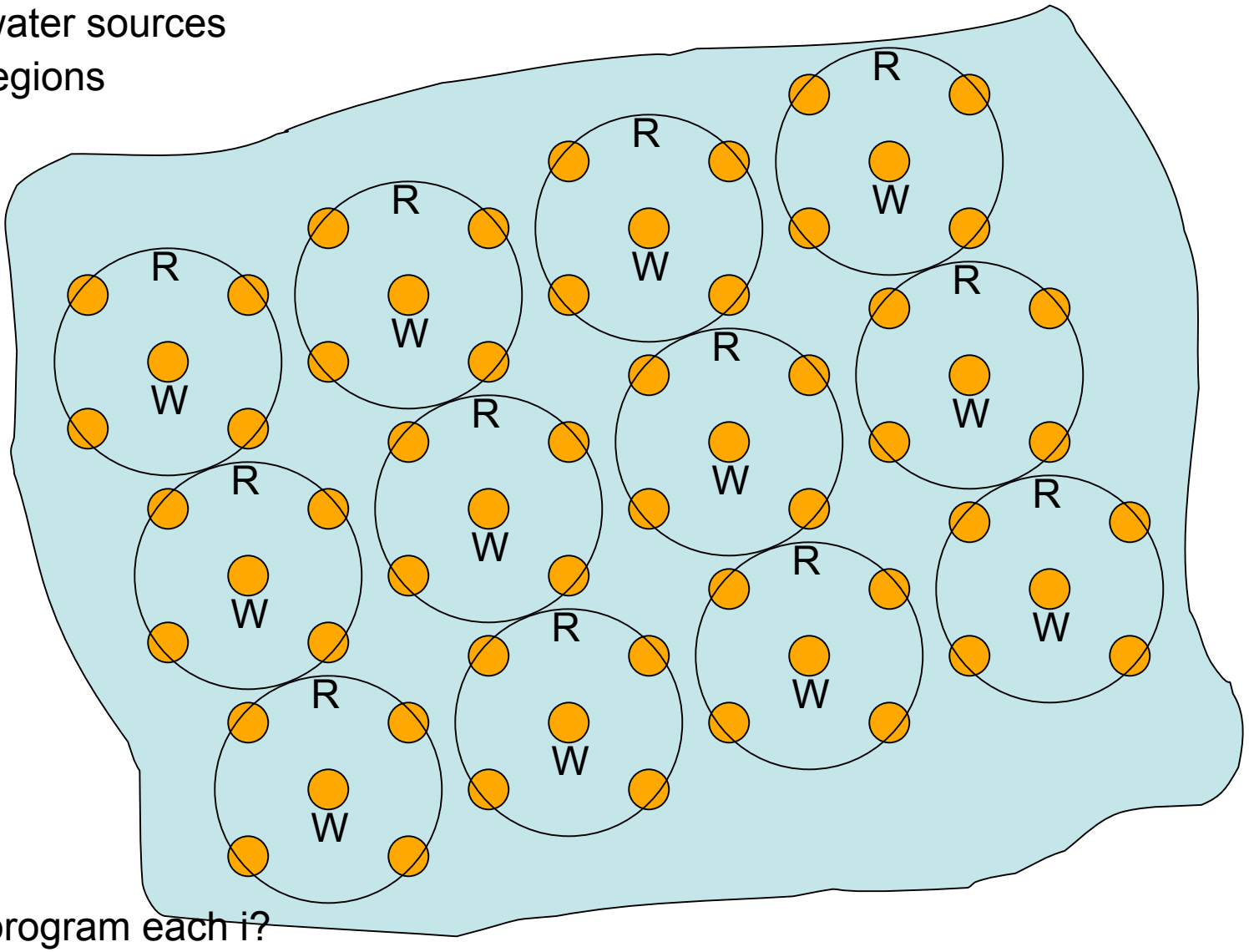
Scenario 1: Irrigation

Irrigate a field based on where the water is needed



Define Neighbor Regions

- W -- water sources
- R_i -- regions



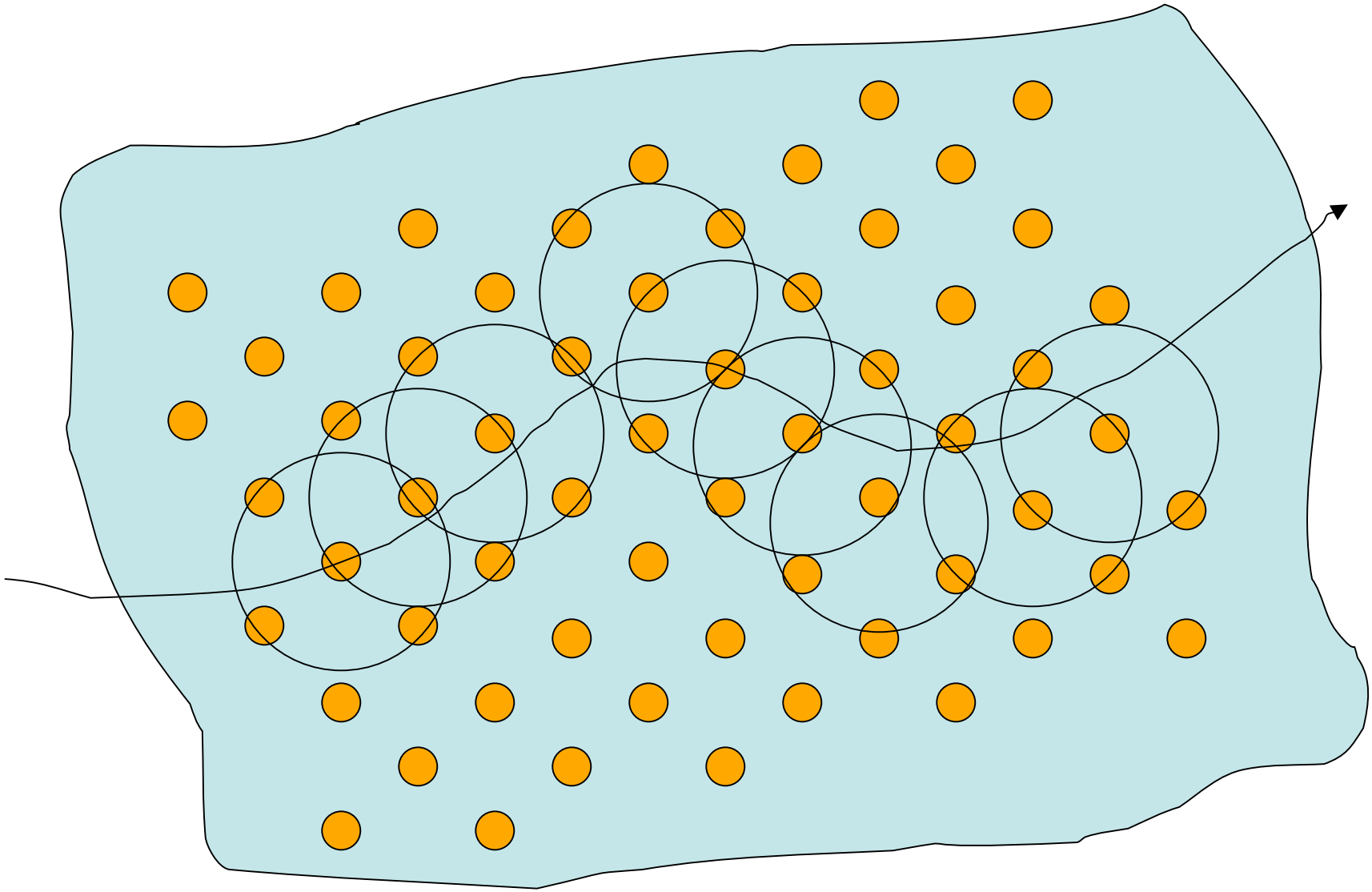
- How program each i?

Scenario 2: Animal Tracking

- Strategy: establish animal location and energize motes on animal trajectory
- Nodes should collaborate to locate target and predict movement
- Identify node with strongest (sensed) signal and pass control (elect leader)
- Turn off motes out of range

- Characteristics:
  - Local control, in network processing
  - Little or no need to report to gateway
  - Local routing

Scenario 2: Animal Tracking

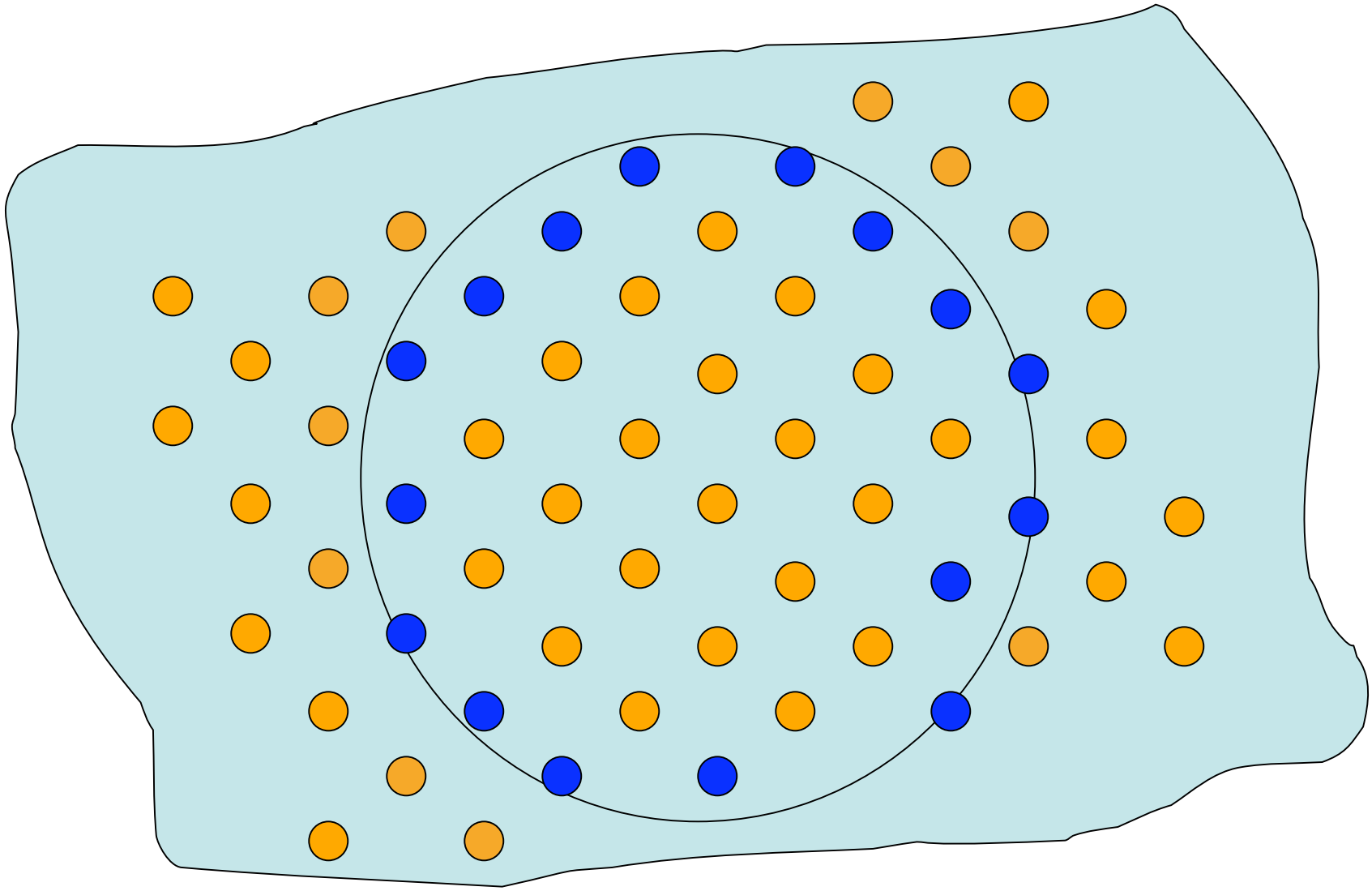


Scenario 3: Boundary Detection

- Strategy: collaborate with neighbors to establish contour on measured signal
- Explore neighbors in region of boundary
- De-energize motes far from boundary
- Generate trigger event for initiating a response or warning

- Characteristics:
  - Local control, in-network processing
  - Little or no need to report to gateway
  - Local routing

Boundary Detection



Open Questions

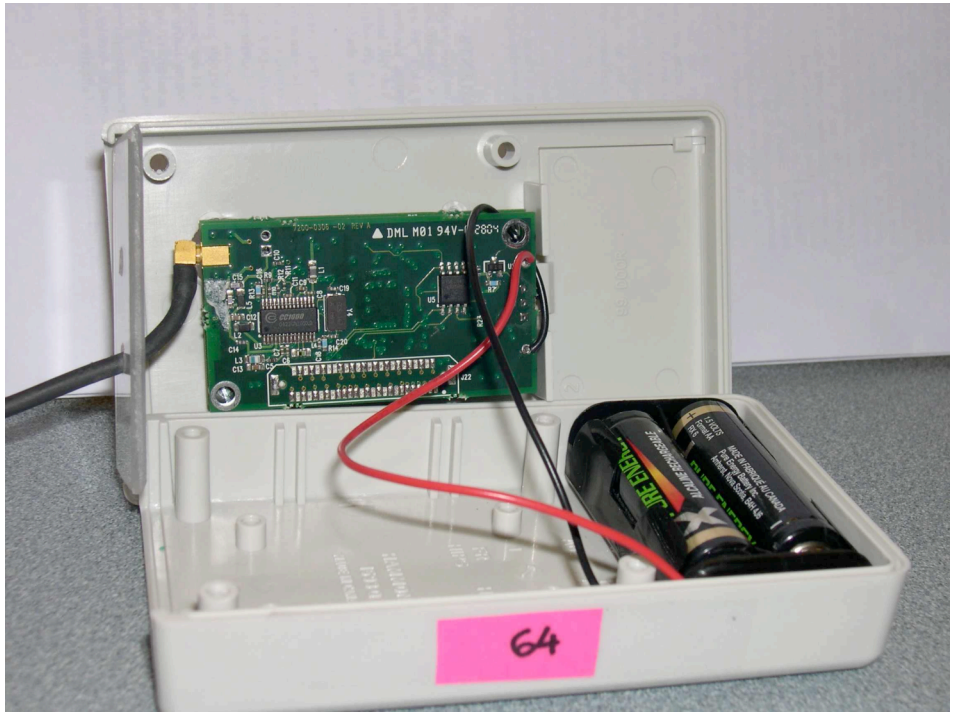
- How do attributes help here?
- How to instantiate the coordination structures and corresponding communication patterns required for these scenarios?
- Abstract regions apply well to predefined, localized communication structures -- how can we generalize?
- General case: sets formed based on attributes and predicates on the attributes -- ad hoc structures

Moving to General Attributes

- Key feature is creation and use of **subsets (**classes, groups, overlays)
- Examples:
  - Locality -- items located near one another, one hop distance, link signal strength -- a “flat routed” perspective
  - Containment -- items located in a room, a building, a town, etc. Or on a route, a path (as in VANET)
  - Modality -- video images, temperatures, wind speed
  - An event: a class defined by a trigger, a threshold value, a predicate
  - Synthesized: an average, a wind chill, a consensus value
- Then traffic can be routed based on subset
- Note: subset could be virtual or physical with respect to connectivity
  - Physical: locality, containment (e.g., nodes near each other) -- AR
  - Virtual: motes that have detected a bird -- not AR
  - Implication is in whether intermediate hops or hierarchy is required vs. flat routed

Defining General Attributes

- Attributes exist are are used in applications
- Need to be more explicit for general use; need to expose consistently
- Currently: encode in bit stream on channel (message type, corresponding TinyOS message structure for that type)
- Use **labeled attributes using XML syntax, common XML data catalog**
- Examples:
  - Weather data (MTS400)
  - Predicates



XML Examples

Weather Data from MTS400

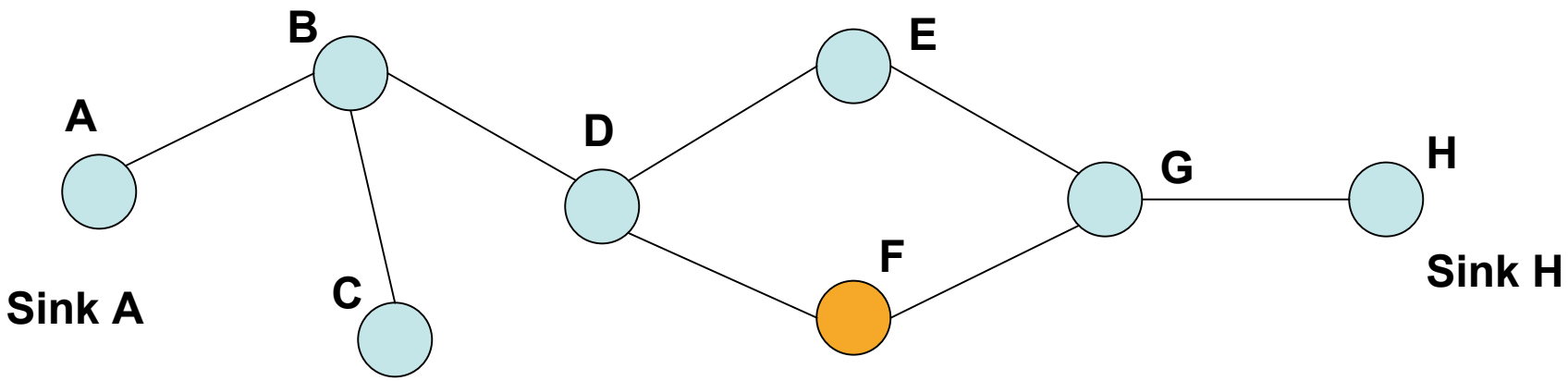
```
<attribute>
<name> light </name>
<type> uint8 </type>
</attribute>

<attribute>
<name> ozone </name>
<type> uint16 </type>
</attribute>
```

Predicate

```
<predicate>
<attribute> temp </attribute>
<operator> less than </operator>
<constant> 2 </constant>
</predicate>
```

Implications

- Nodes need to know how to interpret XML fields (e.g., parser, DTD)
- Precedent: SensorML
- Overhead
  - Estimated 20-30x in ascii text
  - Ways to reduce and manage (e.g., binary XML)
- Next: if data are labeled with attributes, how can we use this new information in routing?



Routing with Attributes

- Answer: use the attributes to make routing decisions
- Example: Suppose we **task a system to sense temperature** at nodes and return to a base station
- Messages are comprised of:

  [NodeID, Time, Location (x, y), Temp, Period]

- Assuming the routing function uses a table that was populated on initial tasking, then
- The router detects a message with temp and forwards the packet, unchanged, to the next hop in the spanning tree to the base station.
- Table at node F:

Attribute	Next Hop
temp	D

Routing with Attributes cont.

- Now suppose there are **two base stations with the same request**
- Messages are comprised of:

  [NodeID, Time, Location (x, y), Temp, Period]
- The table is populated on initial tasking by spanning trees from each base station
- The router detects temp and **forwards the packet, unchanged, to BOTH next hops** in the spanning trees of each base station.
- There is room for the router to optimize and send a single message if the two next hop nodes are the same -- like a single multicast message that splits when the paths diverge [can't do this without looking at the content]
- Table at node F

Attribute	Next Hop
temp	D
temp	G

Great! Now let's look at the tasking problem…

A Tasking Scheme

- Want to inject tasks into the system, like inject queries in TinyDB
- But more general
  - Could ask for interests (like DD)
  - Could send queries to the system (like TinyDB)
  - Could exploit locality (like ARs)
  - Could do ad hoc in-network processing
  - Could task parts of the SNET to be autonomous until events are detected
    - Example: sleep mostly, but if you find an intruder, track her, and send an alert to me if they attempt to enter a critical zone.
  - Could implement an arbitrary coordination structure (e.g., tree, ring, chain, mesh) corresponding to a desired algorithm and in-network processing

- And it's efficient to boot
  - Nodes that are not involved can sleep

Tasking with Attributes

- Example: Suppose we want to **task a system** to sense temperature at nodes and return to a base station
- Populate routing table and function at intermediate nodes
- Minimally, task comprised of

  [BaseID, TaskID, Time, Temp, Period]
- Task is flooded (like a tinyDB query)
- Each node adds a table entry

  TaskID | Temp | Period | Next Hop

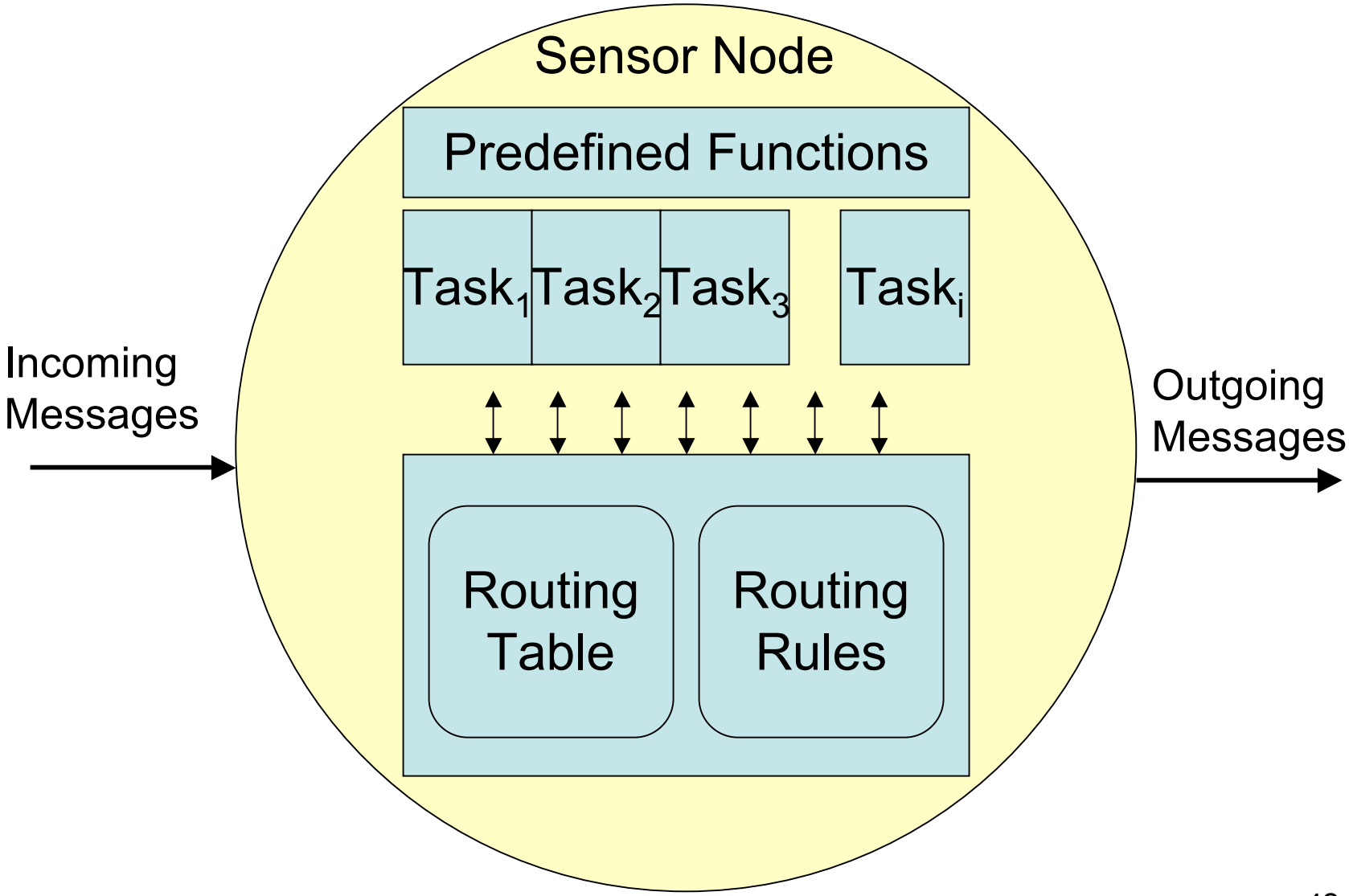
- This is sufficient to populate the routing tables for the aforementioned attribute-based routing example

TaskID	Attribute	Period	Next Hop
01	temp	10	D
02	temp	20	G

Tasking -- How we want to it

- Basic case is reasonable, more generally, we want to
  - Scientist envisions some task for the SNET
  - Formulate how the task should be applied to the SNET
  - Inject the task into the SNET
  - Wait for the task to produce a result and get the results
- Next:
  - How to represent complex tasks (in-network processing?
  - How to represent and evaluate complex predicates (subsets)?
- Need to
  - Either predefine computation (e.g., AVG, contour, etc.) or represent as parameters or mobile code
  - Encode predicates (and attributes)
- Open question:
  - **What aspects of a task are considered as part of the routing function what are not?**
  - Impacts simplicity of routing function

A Model for Routing at Each Node



Tasking a Tracking Scenario with Attributes

- Example: Task a system to detect an track a woodpecker using sound
- Populate table for routing function at relevant nodes
- Task comprised of at least

  [BaseID, TaskID, Time, detection-algorithm, comm-structure, scope-predicate, creation-threshold, deletion-threshold, ...]

- Detection-algorithm must be pre-existing or mobile

- Scope-predicate constrains the scope of the collaborating nodes

  [nearest k neighbors with audio sensor]

- Comm-structure defines the mode of interaction created in local routing tables

  [fully connected | ring | spanning tree | planar graph | …]

- Creation and deletion thresholds provide guidance for change of leader and task migration or replication

Tracking Scenario cont.

- Task is flooded (like a tinyDB query) from base
- Each node adds a table entry for communication back to base

  TaskID | … | Next Hop

- Each node in system instantiates routing function with neighbors corresponding to comm-structure

- Detection algorithm guides leader migration

- Examples of routing table entries at a local node

  TaskID | Signal | Location | Scope-predicate | Next hop 1

  TaskID | Signal | Location | Scope-predicate | Next hop 2

- Similar challenges for contour identification

- Work to be done to pass sufficiently rich description of task to SNET

Summary

- Use attributes to label data
- Use predicates to define arbitrary membership in sets beyond pure locality
- Use task instantiation to establish routes to a base station, and within communication structure and scope rules
- Most open challenge: clearly defining what is task, what is routing

- Next: relation to environmental applications

NeTS-NOSS Motivation

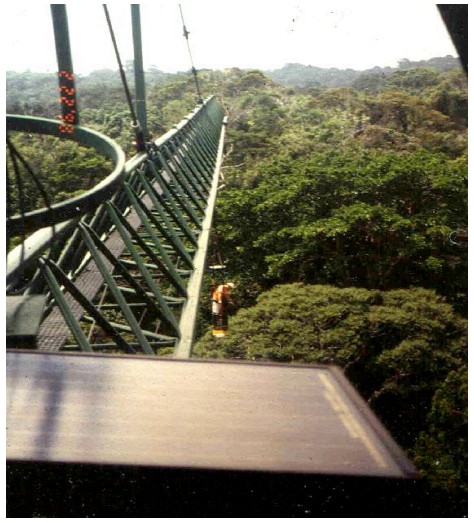
- Attribute-based routing being designed ecology research context
- Environmental applications are fundamentally limited by energy
- Require long-term deployment
- **Characterized by stasis, punctuated by extreme events on short time scales**
- Broad frontier of scientific inquiry devoid of viable instrumentation

Application: Study of forest eco-hydrology and forest-atmosphere mass and energy exchange

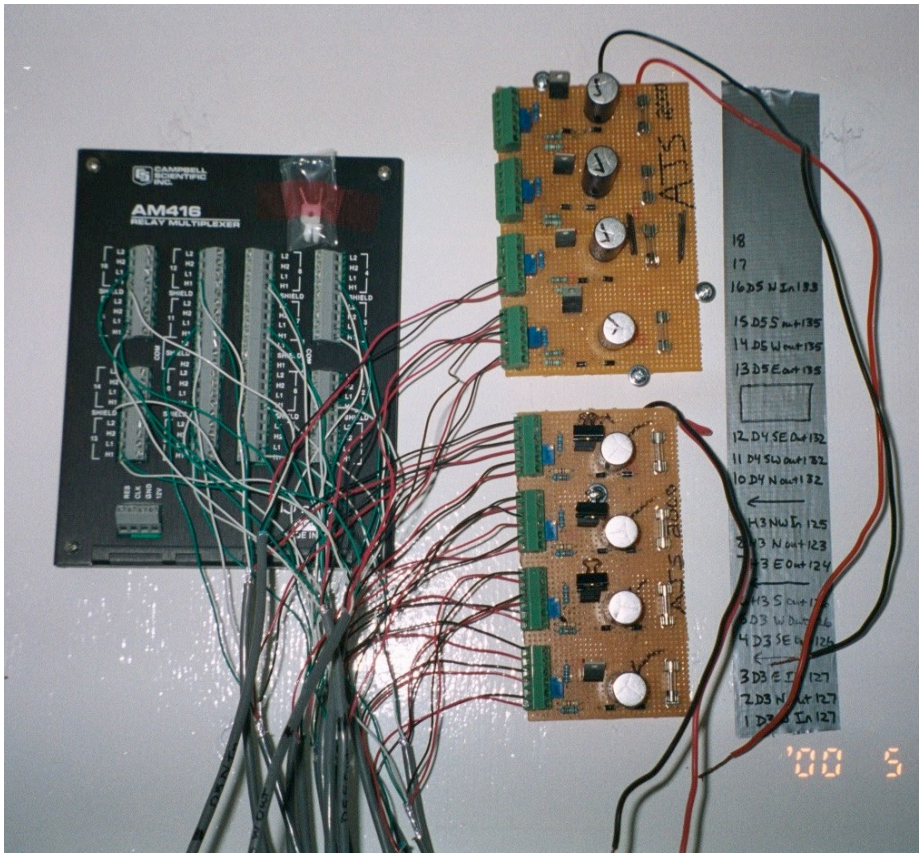
Temperate conifer

Dry angiosperm

Tropical angiosperm



A key limitation is having a wired and unintelligent data acquisition system



Sensor wires go to the Campbell Multiplexer and power boxes, which connect all sensors to the logger.

Application: Study the Ecosystem of Brazilian Free-tailed Bats

- Millions of bats
- Foraging area in 1000's of sq Km
- How impact ecosystem?
- How instrument with sensors?
- Measure what we can…in a SNET

Nightly Dispersal of Brazilian Free-Tailed Bats

Bat Fatalities in Wind Farms

High-Level Approach

- Deployment of sensor arrays
- Soil moisture --> emergence of insect prey --> increased likelihood of foraging bats --> tracking and identification of bats --> shutdown of selected wind turbines
- In-network distributed information processing

- Correlation of data from remote sensing (NEXRAD radar)

Thrusts

The following areas are open for investigation

- Conceptual/theoretical
  - Behavioral transition models for adapting to DTN or fully connected regimes
  - Tasking schemes: representations that are flexible and compact
  - Task instantiation mechanisms
  - Custody transfer in DTN and VANET
  - Performance characterization

- Platform and testbed
  - Implement wrapping of observations from motes in XML
  - Creation of web services interface to SNET via gateway
  - Implementation of routing function, tables, update mechanism using nesC, TinyOS, and Mica2 mote platform
  - Implement in-network processing algorithms (consensus) for event detection using nesC
  - Task instantiation
  - Implement A-B for different scenarios: parking, microclimates, bats…

End

- Dissertation defense of Andrew Ke
- Performance aspects of attribute-based routing and the containment attribute
- 3 PM Thursday, 12-15-2005

Extra Slides

Two Gateways Revisited

- Now suppose there are two base stations with **different** requests
- TaskA comprised of
  - [BaseID_A, TaskID_A, Time, Temp, [Temp < 2], Period = 10]
- TaskH comprised of
  - [BaseID_H, TaskID_H, Time, Temp, [Temp < 2], Period = 20]

- Tasks are flooded, spanning trees created
- Each node adds a table entry
  - TaskID_A | Temp | Period_A | Next Hop_A |
  - TaskID_H | Temp | Period_H | Next Hop_H |

- Messages are comprised of:
  - [TaskID, NodeID, Time, Location (x, y), Temp, Period]

- Data are sourced based on the Temp < 2 predicate -- do not hit the router unless this is satisfied
- The router detects Temp and forwards the packet, unchanged, to BOTH next hops in the spanning trees of each base station when the sampling duty cycles coincide (a strong argument for harmonics)
- It forwards to A for other cases

Other Considerations for a SNET

- Conventional (network) routing has name resolution (DNS)
  - Translate name to number -- useful for efficiency
- Can’t fit all name resolution at one node

- SNETs lack regular topology
- Can be dynamic topology -- mobile nodes

- In some cases there is NO regular infrastructure (e.g., VANET)

- Routing can be specific to the task at hand to be efficient

Extensive Use of Transducers and Data Acq

Water flux sensors implanted into the boles of 60 m tall Douglas-fir trees in Washington.