NEST Wireless OEP Application Decomposition Exercise

University of California, Berkeley
University of California, Los Angeles
Kestrel Institute, Palo Alto
University of California, Irvine

Outline

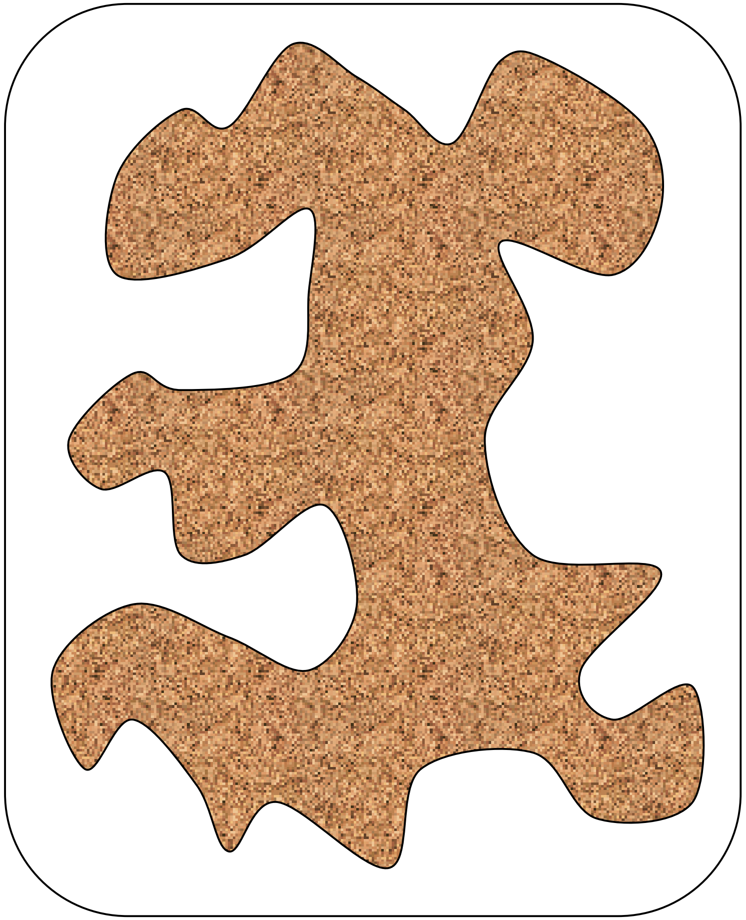
- Minitask Approach
- Application Scenario
- Platform
- Interactions Among Components
- Time-Bounded Synthesis
- Composition
- Coordination Service Approaches
- Real Time and Fault Tolerance
- Conclusions

Minitask Approach

- Use the minitask to exercise NEST software technology concepts
  - identify NEST components in the context of a specific application
  - relationship among components
  - key challenges
  - candidate solutions
- rather than to test particular controller designs in the small

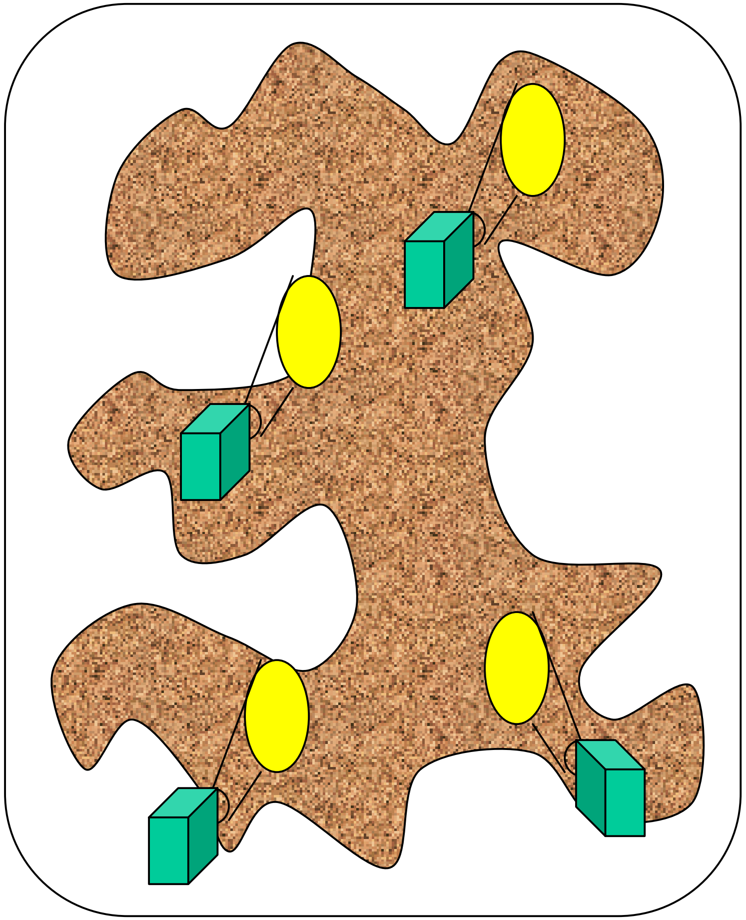
Application Scenario

- Dense field of small local sensor nodes over a portion of a large space
  - limited power & bandwidth



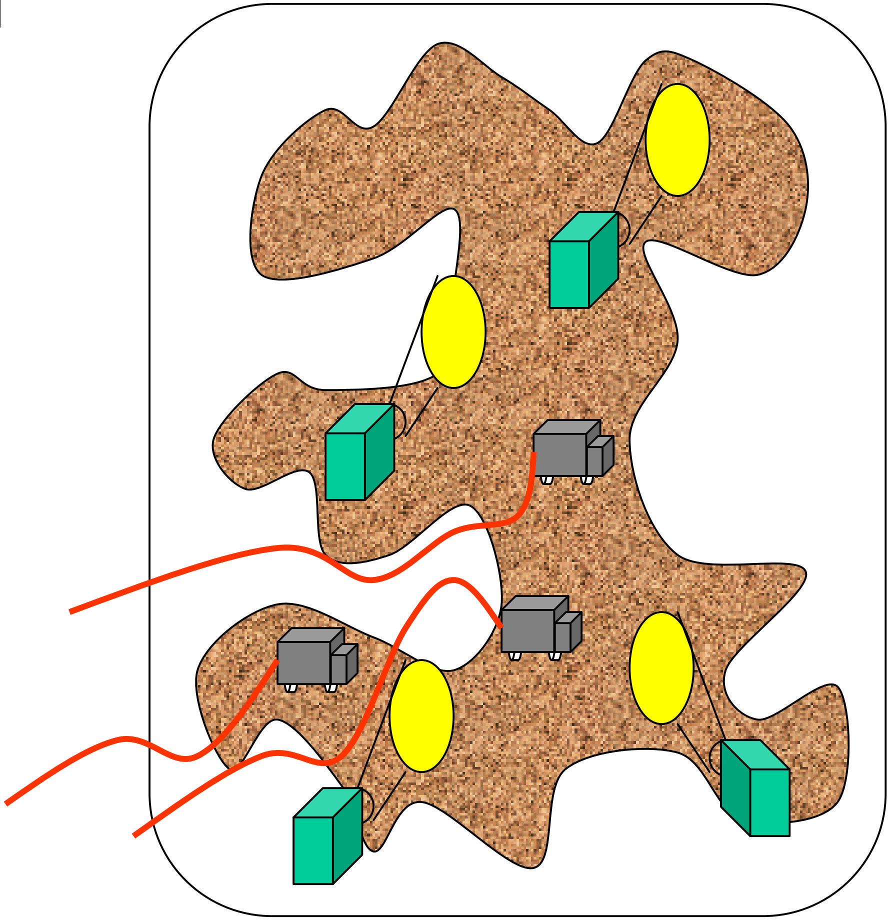
Application Scenario

- Dense field of small local sensor nodes over a portion of a large space
  - limited power & bandwidth
- Sparse higher powered resources with longer range cameras
  - limited field of view



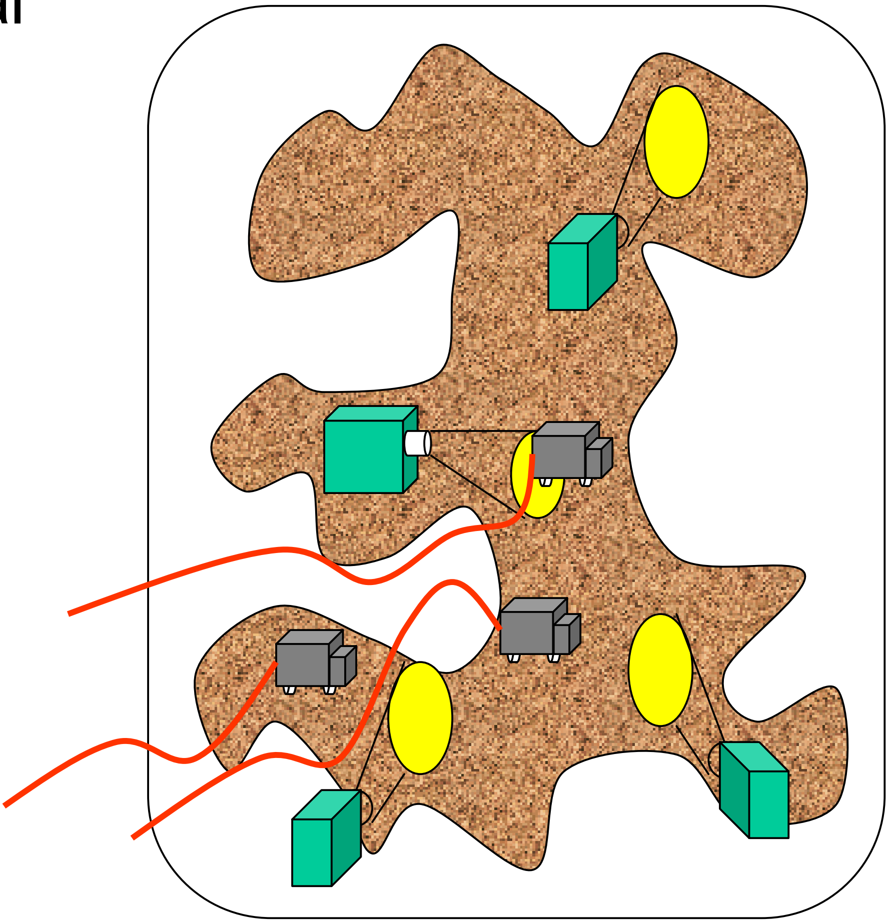
Application Scenario

- Dense field of small local sensor nodes over a portion of a large space
  - limited power & bandwidth
- Sparse higher powered resources with capable directional modes
  - cameras
  - limited field of view
- Multiple objects moving through



Application Scenario

- Dense Field of small local sensor nodes over a portion of a large space
  - limited power, BW
- Sparse higher powered resources with capable directional modes
  - cameras
  - limited field of view
- Multiple objects moving through
- Track and image subset with particular feature



Binding the Basic Scale

- 10^4 nodes, 10m ave. spacing,
  - 30m range (~20 neighbors)
  - $1km^2$ of patches out of ~$20km^2$ of space
- 1% higher powered nodes (100)
  - roughly 300m x 300m patches

Feature: fastest moving objects

Goal → Metrics

- Keep the fastest moving objects in field of view within available energy budget
- Metric:
  - maximize at each time t, $\Sigma_{\text{target } i}\, w(i) \times viz(i)$
  - where weight $w(i) \sim \text{speed}(i)^2$,
  - visibility $viz(i) \sim$ quality of imaging (1/distance)
- subject to
  - camera: limited view, limit number,
  - communication: limited bandwidth and range
  - energy: limited # messages per unit time
  - fault rate in nodes & links > 0

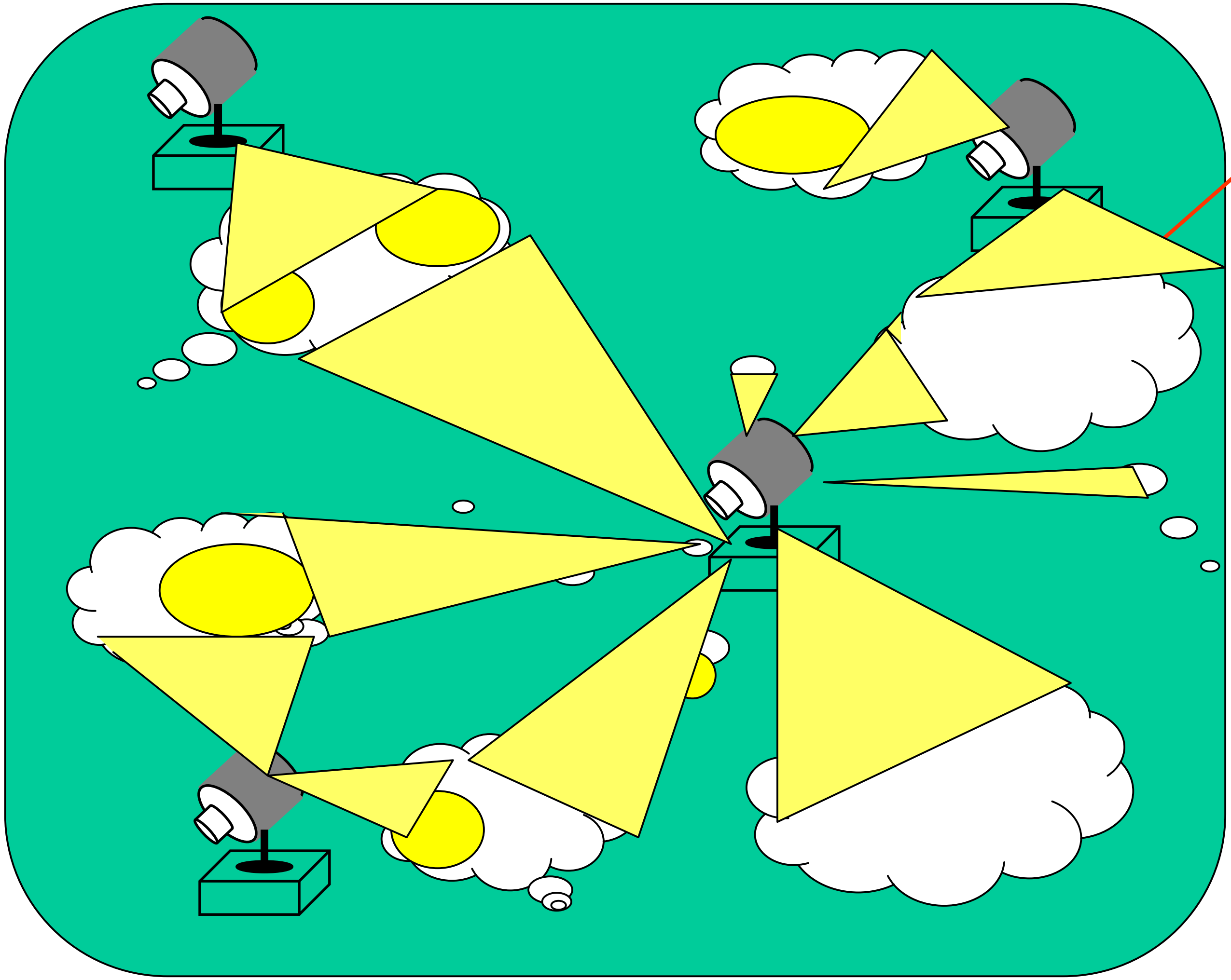
Basic Capabilities

- Local sensor observations at defined rate
- Messaging
- Energy monitoring (& harvesting?)
- Camera control and video processing

Parameterized Services

- Time synchronization (*fidelity*)
- Local coordinates of the nodes (*fidelity*)
- Estimated target position and velocity (*fidelity*)
- Routing (*redundancy*)

Classifying Activity over Space

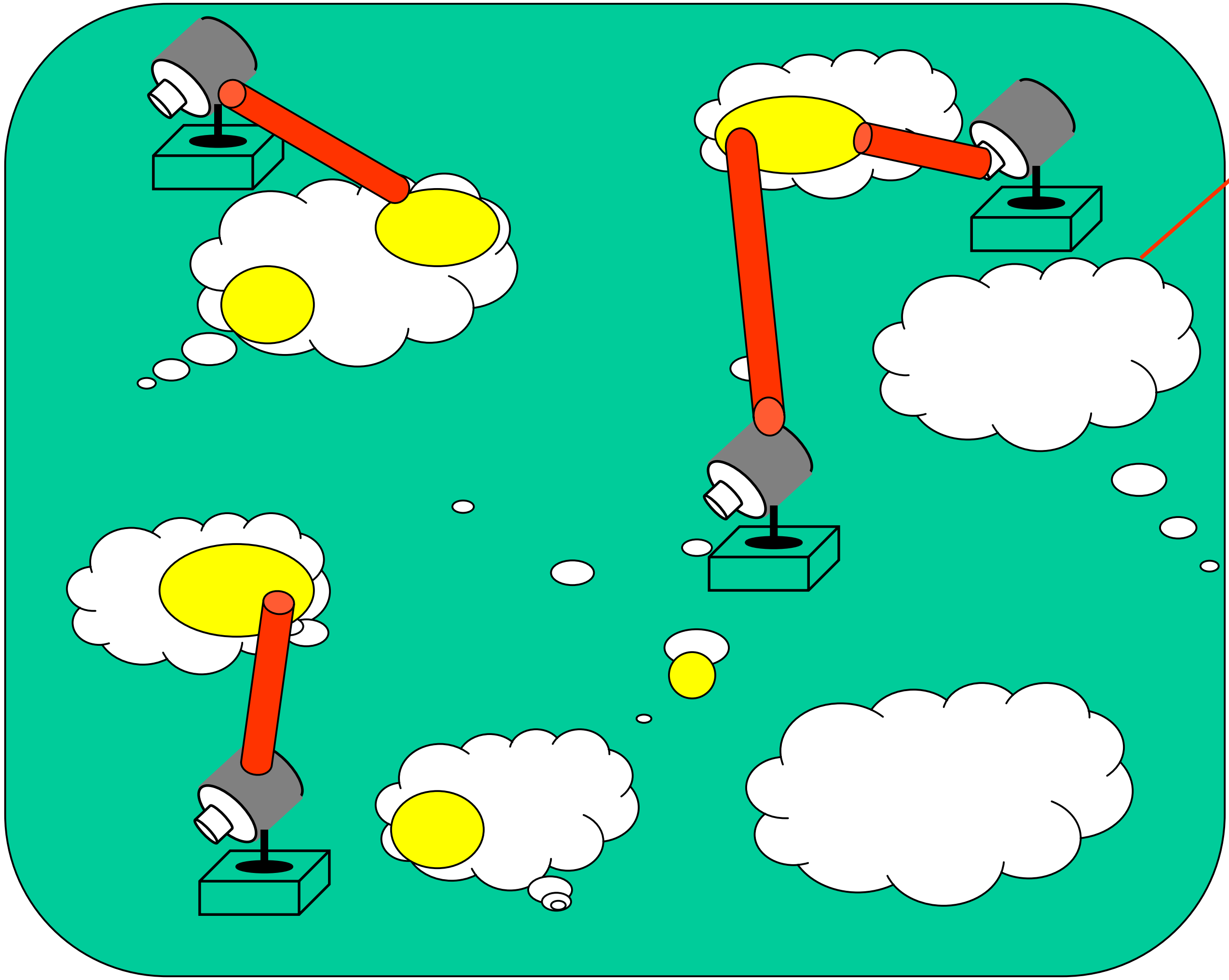


Numerous areas of activity

- detected locally

Subset determined to be worth monitoring

Classifying Activity over Space



Numerous areas of activity
- detected locally

Subset determined to be worth monitoring

Few individuals targeted

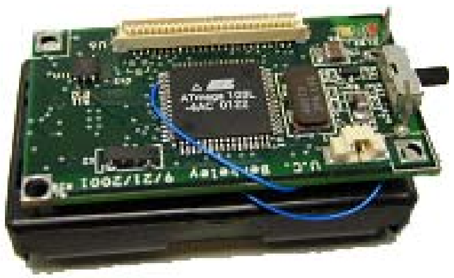
State → **`active`**, **`monitored`**, **`targeted`**

Platform

Hardware units

- Large number of constrained wireless nodes
  - two modes of sensing (acoustic and magnetic or vibration)
  - limited radio range
  - event-driven OS structure
  - limited energy reserves
- Small number of more powerful nodes
  - bridge short-range RF to long range communication
  - processing and storage capabilities
- Specialized “power assets”
  - computation and storage resources
  - cameras – pan, tilt and zoom but not covering entire space
  - panels with microphone arrays

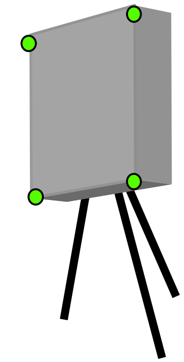
Field Nodes ("motes")



- Atmel ATMEGA103
  - 4 Mhz 8-bit CPU
  - 128KB Instruction Memory
  - 4KB RAM
- 4 Mbit flash (`AT45DB041B`)
  - SPI interface, 1-4 μj/bit r/w
- RFM TR1000 radio
  - 50 kb/s
  - Sense and control of signal strength
- Network programmable in place
- Multihop routing, multicast
- Sub-microsecond RF node-to-node synchronization
- Provides unique serial ID's

Power Nodes and Assets

- Bridge low-power network to 802.11
- Full Linux environment
- Cameras with pan and tilt
- Panels with microphone array
- Potentially: additional computational support such as DSP and FPGA for high-end acoustic, vision processing



Key Components

- Basic Capabilities
  - messaging, sensing, pointing, processing
- Parameterized Coordination Services
  - time synchronization
  - local coordinates of all the nodes
  - target position and velocity estimation
  - routing
- Synthesis and Composition
  - key requirements clear from service interaction (below)

Interactions Among Components

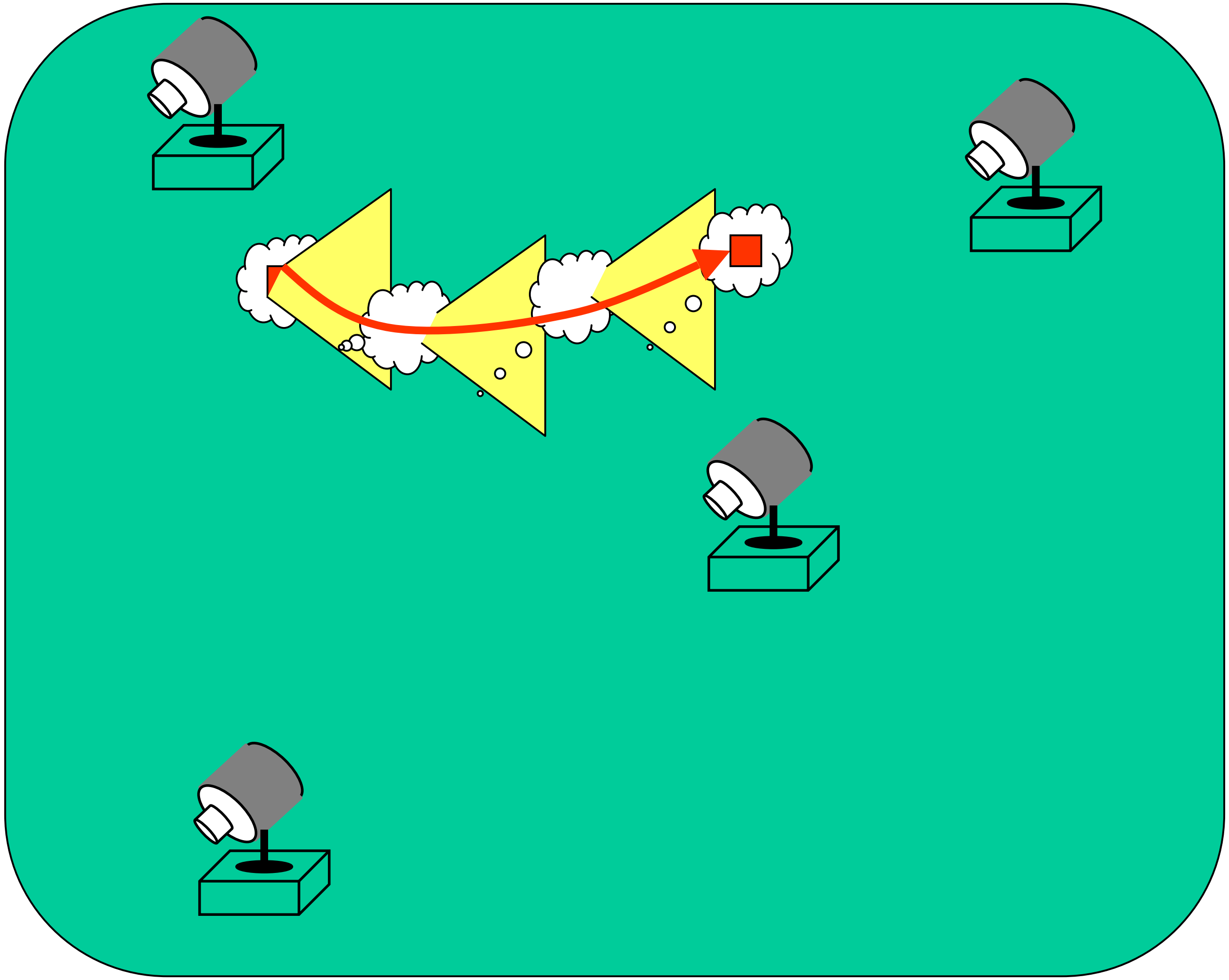
Time Synchronization & Local Coordinates

- Required to correlate observations from multiple nodes
  - local estimation of target position and velocity
  - non-local activity classification
- Fidelity depends on use and resources
  - high local accuracy is inexpensive
  - higher accuracy needed at higher state
  - more expensive to maintain over distance
  - higher level resources can refine accuracy
    - energy cost in doing so

Local Estimation of Target Position & Velocity

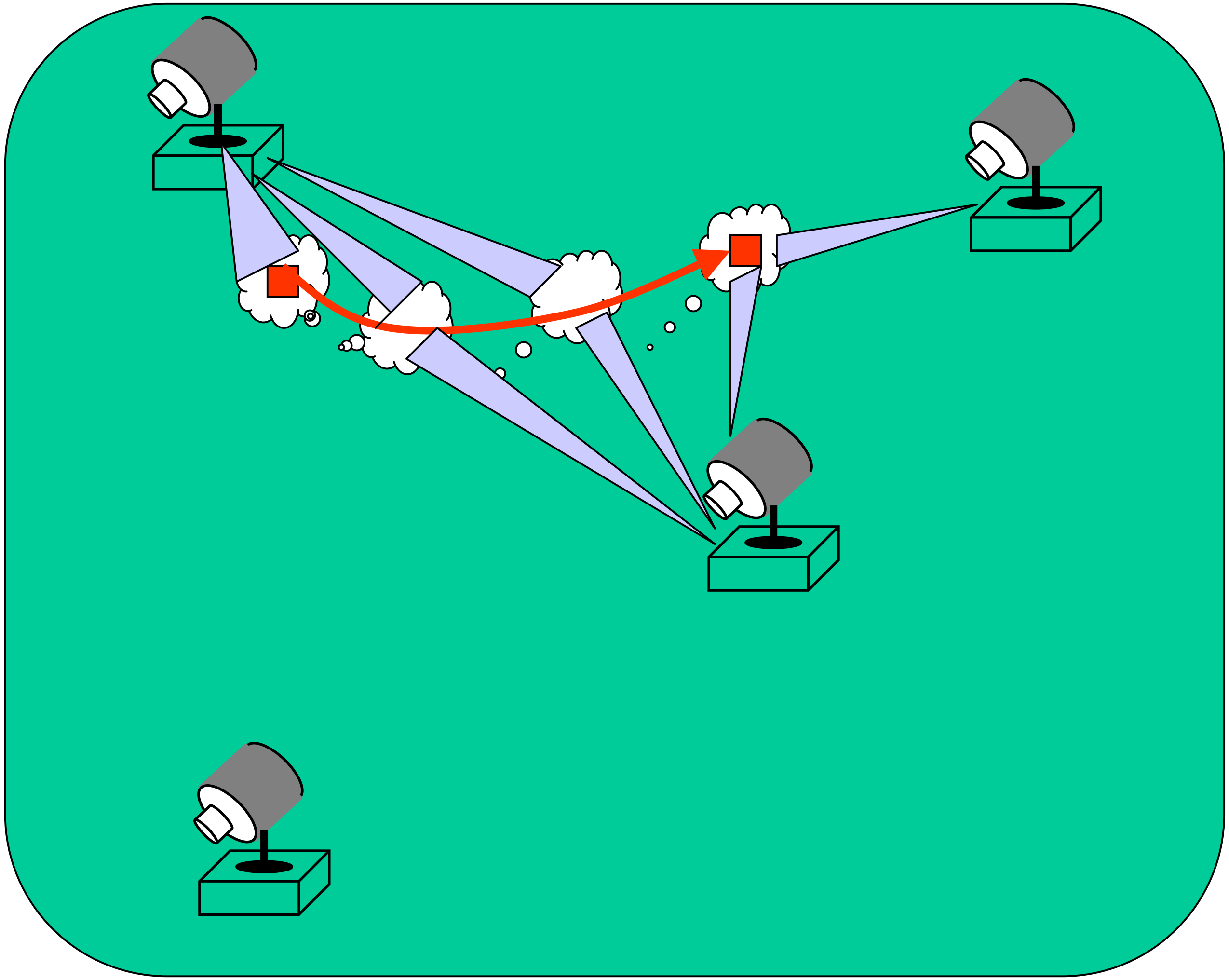
- Inputs
  - local sensor observations
  - local estimate of location and time + courser global reference
  - neighbors' observations and their loc. & time
  - refinements from global level
  - fidelity requirements
- Use of estimates
  - traversal of observation activity across network
    - see next slides
  - notification of candidate for classification
  - initial camera pointing

Local Observation Tracking



- Use estimates of position and velocity to alert potential next observers
- Focus activity to reduce energy
- Local algorithms robust to faults and delays

Tracking Drives Efficient Routing



- Multihop routing paths to higher monitoring nodes evolve
- Tracking and higher-level goals guide network scheduling
- Fault tolerance determines redundancy in routing

Higher Level Processing

- Given classification and assignment
  - control camera to maximize visibility of targeted objects
  - reinforce information fidelity from monitored sites
    - amount & timeliness of information sensed / communicated
  - suppress information fidelity from uninteresting sites
  - feed information back to enhance fidelity
    - time or location
- Reconfiguration: Given classification and old assignment, assign monitoring and targeting to powered resources
  - e.g., handoff to new cameras or monitors
- Reclassification
  - new objects become “among fastest”
  - pushing information out regarding feature thresholds
  - propagating potential triggers up

Issues that drive the NEST discussion

- Targeting of the cameras so as to have objects of interest in the field of view
  - tracking control is routine, assignment is issue
- Collaboration between field of nodes and platform to perform ranging and localization to create coordinate system with adaptive fidelity
- Adaptive routing structures between field nodes and higher-level resources
- Targeting of high-level assets
- Sensors guide video assets in real time
- Video assets refine sensor-based estimate
- Network resources focused on region of importance

Closed Loop at many levels

- Field nodes collaborate with power nodes to perform ranging and localization to create coordinate system
- Need to maintain associations between field nodes and power assets (monitors relation)
- Selection of low-level assets per object over time
  - determined by local sensor processing and high-level coordination
- Selection of power assets over time
  - determined by in-coming data and higher processing
  - determines dynamic association (incl. routing structures) over time
- Targeting of power assets
  - sensors guide camera assets in real time
  - camera assets refine sensor-based estimate in real time
- Network resources focused on regions of importance

Time-Bounded Synthesis

Configurations/Schedules

- Resource Assignment
  - given classification, allocation and rate of change, compute new allocation
  - time and energy to affect change
  - energy and visibility cost as targets move away from current assignment
- Multihop Routing Resource Scheduling
  - given selection of monitored sites and mapping to higher level nodes, compute (rough) communication schedule in time and space

Application-Requirements Constraint

- Constraint:
  - the assignment of cameras to targets is “optimal” (see later)
- Design decisions:
  - go for “naïve” local improvement scheme
  - “data diffusion”: each node maintains nearby-world-state estimate
- Constraint is maintained by:
  - (code making sure that) camera changes field of regard whenever this improves the assignment quality
- Subsidiary constraints:
  - nodes know nearby target states (position and velocity)
  - nodes know nearby camera assignments

Optimality Metric

- Boundary conditions on metric:
  - the faster a target, the more important it is that some camera view it
  - nearby cameras are better for viewing than far-away cameras
- Formula:
  - sum over targets of: (target weight) x (target visibility)
  - target weight = $(\text{target speed})^2$
  - target visibility = zero if no camera assigned; or minimum over assigned cameras of 1 / distance(target, camera)
- Remarks:
  - formula uses estimates for position and speed
  - suitable for local anytime optimization
  - simplified for purpose of exposition
  - untested; may need tweaking for satisfactory results

Information-Consistency Constraints

- Generic constraint:
  - neighboring nodes agree on overlapping information
- Design decisions:
  - bootstrapped information-quality decay estimators (for example)
  - max likelihood reconciliation (for example)
- Constraint is maintained by:
  - nodes obtain sensor measurements whenever information quality would fall below threshold
  - nodes update estimates using new information
  - nodes transmit overlapping information to neighbors
- New constraints:

  NONE

Specifically for Tracks:

- Data exchanged:
  - set of (time, position, speed) for targets; one element per detected target
  - data includes uncertainty information
- New data:
  - obtained from sensors (including cameras)
- Reconciliation:
  - performed independently by each node
  - sensor data is brought into same framework of (time, position, speed) + uncertainty, and added to the data set
  - obsolete data (too old or superseded) and data on "irrelevant" targets (too far) is discarded
  - node computes the most likely track data for the present situation explaining the data set, giving a new data set to be communicated to neighboring nodes

Specifically for Camera Assignment:

- Data exchanged:
  - each data-set element is extended with: set of cameras assigned to this target + for each camera: when assigned
- New “data”:
  - only camera proxy nodes revise assignments: determine the best target assignment for *this* camera given known data
- Reconciliation:
  - the track-data computation is extended with: find the most likely current camera assignment

Run-Time Adaptation

- Mode change (in both Motes and Power Nodes) due to major changes in resource and/or environment conditions
  - mode 1
    - mission 1
    - fault tolerance goal 1
  - mode 2
    - mission 2
    - fault tolerance goal 2
- Activation and deactivation of components

  e.g., vibration sensors are not needed in this application.
- Adjustment of parameterized components

  e.g., the RF signal strength of this level is adequate in this application environment.

Composition

Inputs to Composition

A. Libraries of

- various coordination and other middleware service schemas
- information-consistency maintenance schemas
- anytime optimization schemas
- application-specific schemas

where a schema consists of a parameterized triple:

1. constraint to be maintained
2. (symbolic) maintenance code
3. subsidiary constraints

B. Application requirements expressed as top-level constraint (typically a conjunction of many simple constraints)

Constraints are *soft* and typically involve temporal operators (“everywhere eventually always . . .”)

Construction Process

- General construction approach
  - at design time constraints are matched to schemas
  - instantiation results in production of maintenance code (to be executed at run time) and new (“subsidiary”) constraints
  - repeat until no constraints left
- Information-consistency constraints
  - real-world information as maintained by nodes is consistent with sensor readings
  - shared information is locally consistent
- Information maintenance design-time decisions
  - choice of data-fusion algorithms
  - frequency of updates and other trade-offs

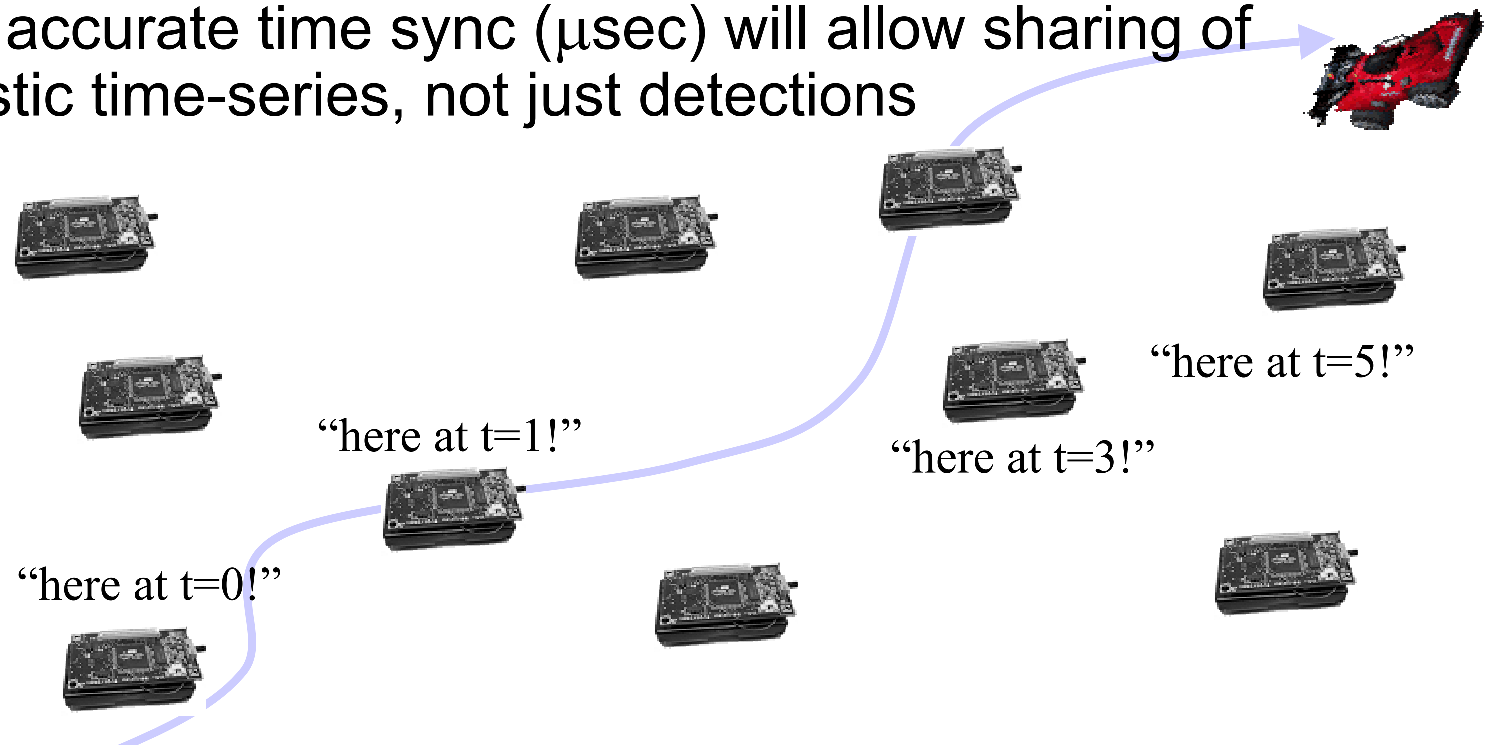
Code Generation

- High-level, symbolic code produced by construction process is collected
- Iterative symbolic simplification, pruning and high-level optimization (e.g. incrementalizing information-updates by data differencing)
- Mapping to low-level executable code

Coordination Service Approaches

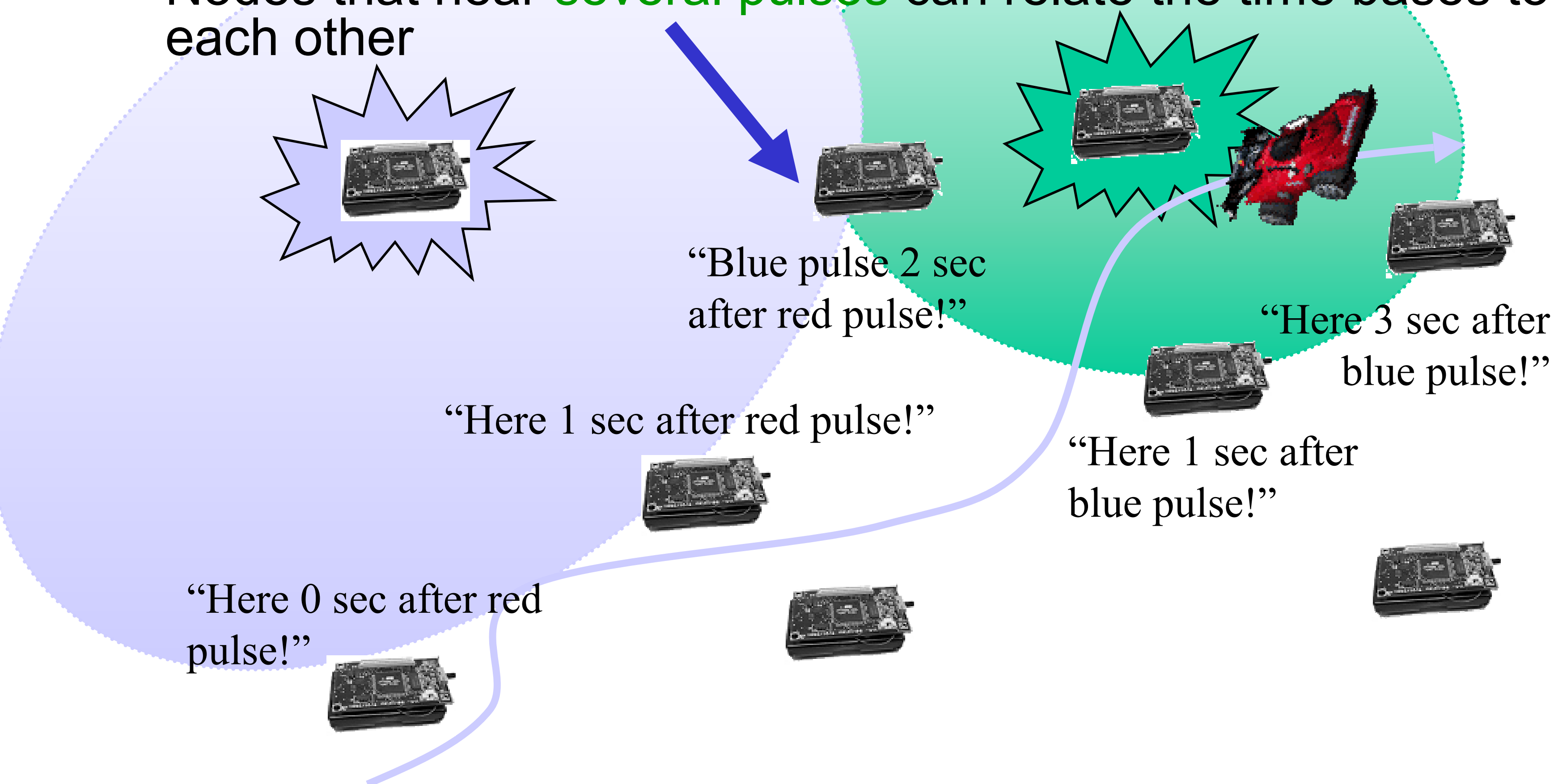
Time Synchronization

- Time sync among motes allows them to compute target tracks collaboratively
- Target detections are communicated (along with position of detector in derived coordinate system) with approximate global or shared time
- More accurate time sync (μsec) will allow sharing of acoustic time-series, not just detections

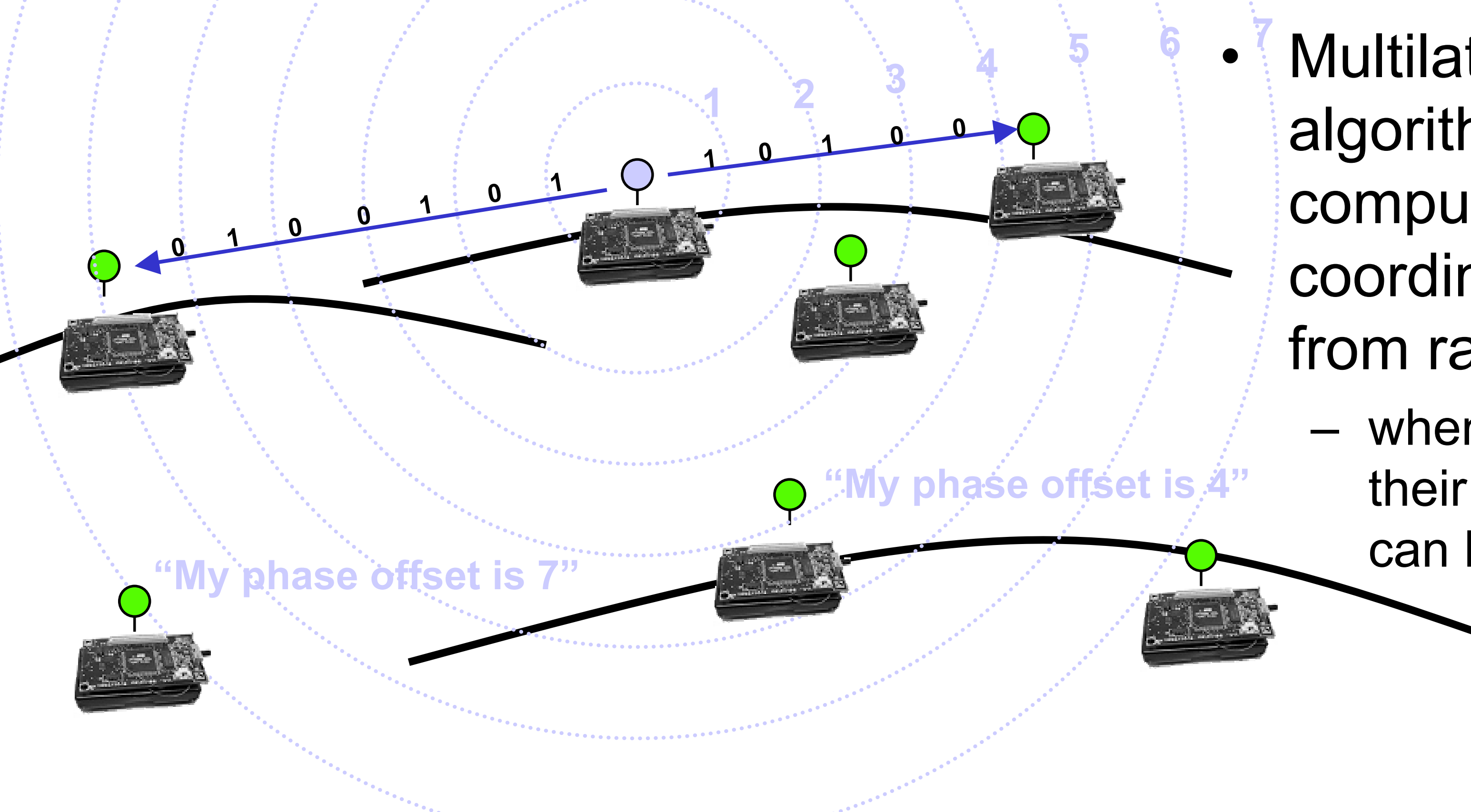


Multi-Hop Time Sync

- Some nodes broadcast RF synchronization pulses
- Receivers in a neighborhood are synced by using the pulse as a time reference. (The pulse *senders* are *not* synced.)
- Nodes that hear several pulses can relate the time bases to each other



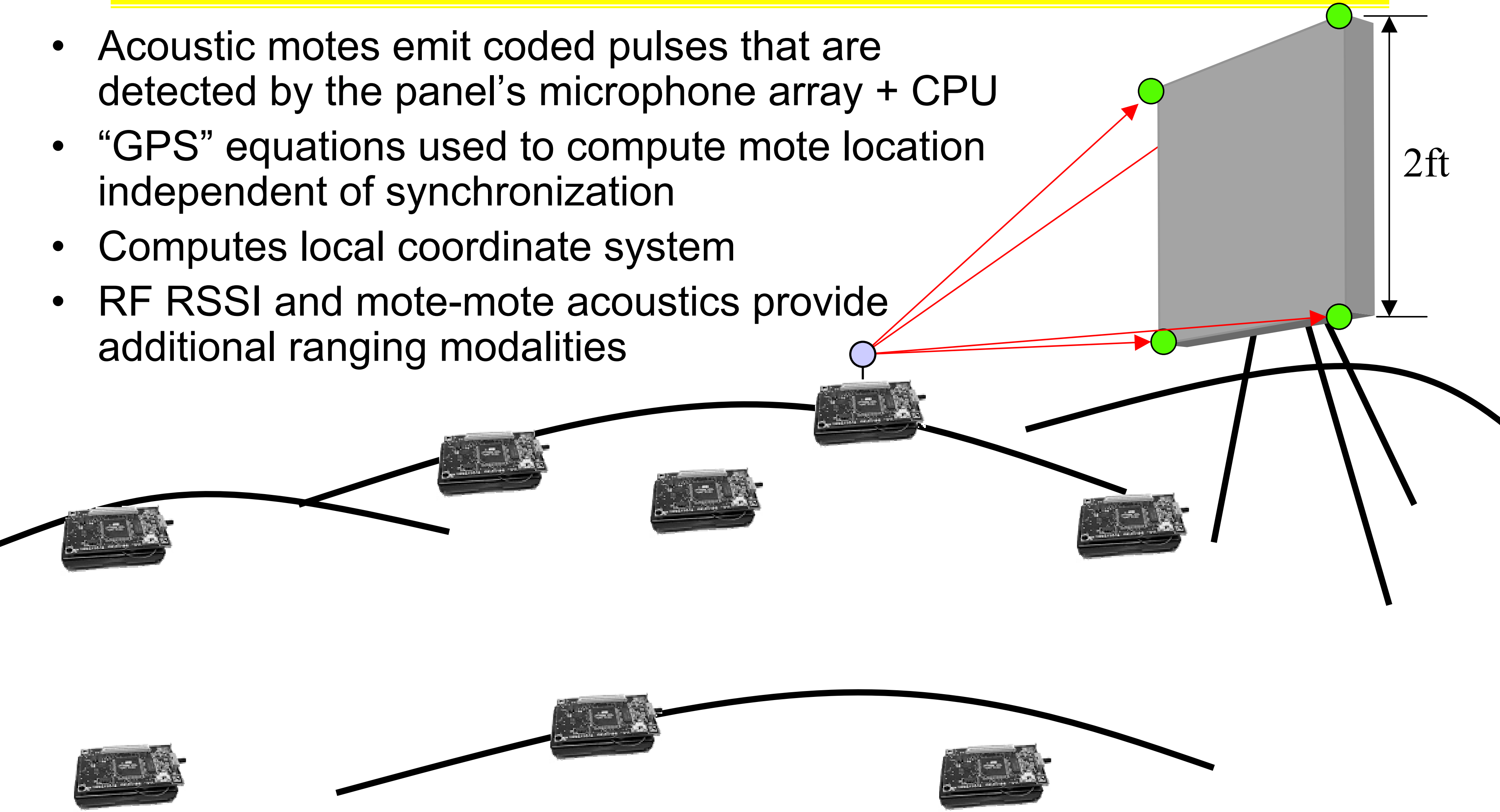
Local Coordinate System (1)



- Time of flight and phase offsets used to compute many-to-many ranges
- Multilateration algorithm computes local coordinate system from ranges
  - when nodes know their location they can help track

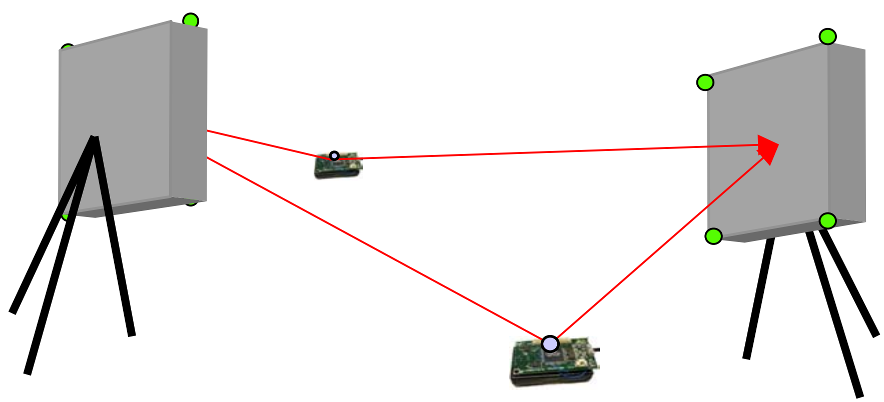
Local Coordinate System (2)

- Acoustic motes emit coded pulses that are detected by the panel’s microphone array + CPU
- “GPS” equations used to compute mote location independent of synchronization
- Computes local coordinate system
- RF RSSI and mote-mote acoustics provide additional ranging modalities



Relating Local Coordinate Systems

- As for time sync, motes that can receive several panels can relate the local coordinate systems to each other
- For the 2D case this requires a non-colinear constellation of two panels + two motes that were heard by both panels

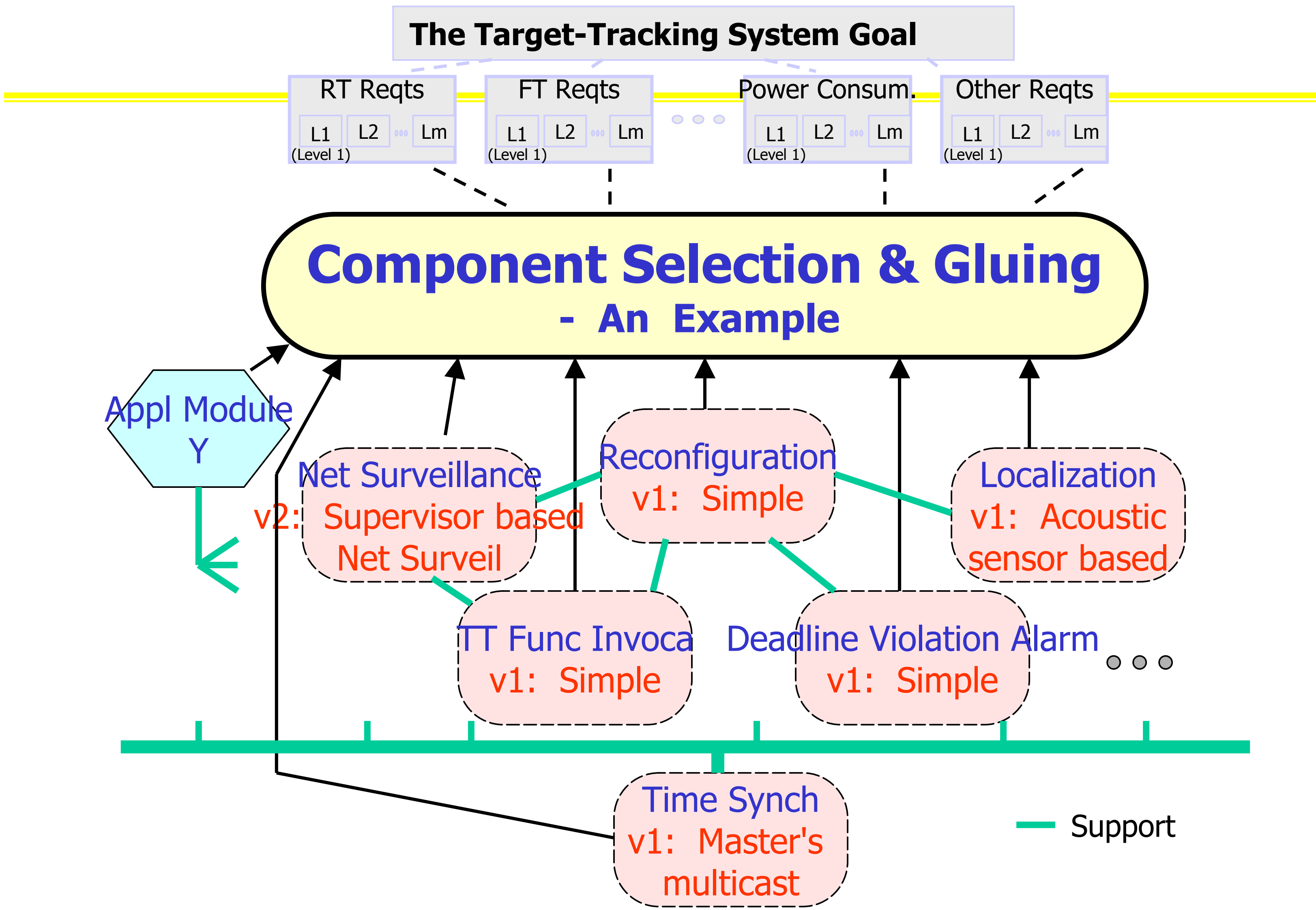


- Messages passed between regions with different local systems can be translated in transit to new local system

Real Time and Fault Tolerance

Major Factors in Component Selection

- Quality requirements (imposed on components)
  - real-time service qualities
  - fault tolerance contributions
  - power consumption contributions
  - memory requirements
  - scalability
- For each component type, multiple versions with differing qualities exist.
- Compatibility (or composability) among the chosen component (versions)
- Analyzability of the qualities of composed systems or subsystems



The Target-Tracking System Goal
RT Reqts
L1
(Level 1)
L2
Lm
FT Reqts
L1
(Level 1)
L2
Lm
Power Consum.
L1
(Level 1)
L2
Lm
Other Reqts
L1
(Level 1)
L2
Lm
Component Selection & Gluing
- An Example
Appl Module
Y
Net Surveillance
v2: Supervisor based
Net Surveil
Reconfiguration
v1: Simple
Localization
v1: Acoustic
sensor based
TT Func Invoca
v1: Simple
Deadline Violation Alarm
v1: Simple
Time Synch
v1: Master's
multicast
Support

System Parameters: Platform

- Sensor network features:
  - average nodes distance, area covered
  - max sampling period
  - time and energy cost per estimation (fidelity)
  - time and energy cost per communication
- Power node features:
  - camera range, motion, quality
  - computational capacity
- Target features:
  - max number of targets
  - maximum speed, acceleration

System Parameters: Fault Types & Rates

Faulty Component	MTBF		
		MTBN	MDN
Mote - processor	secs		
Mote - sensor 1	secs	secs	msec
Mote - sensor 2	secs	secs	msec
…	secs	secs	msec
Mote - outgoing comm link	secs	secs	msec
Mote - incoming comm link	secs	secs	msec
PowerNode- to-Mote link	secs	secs	msec
PowerNode- from-Mote link	secs	secs	msec
PowerNode-to-PowerNode link	secs	secs	msec
PowerNode - processor	secs		
PowerNode - Camera	secs	secs	msec

- MTBF: Mean time between failures

- MTBN: Mean time between naps
- MDN: Mean duration of each nap

Performance Goals at a Lower Level:

- Detection latencies $< \kappa$ msec
- Recovery time bounds
  - Max difference between a normal task execution time and the time for a task execution involving fault detection and recovery events

From	To	Recovery Time <
1st detection by a sensor in a whispering mode	Order to a camera for chasing + alerting motes	η msec (e.g., 200 msec)
-----	-----	--- msec

- Time overhead during fault-free operations
  - Time costs of enabling fault detection & advance prep for recovery

From	To	Time Overhead <
1st detection by a sensor in a whispering mode	Order to a camera for chasing + Alerting motes	η msec (e.g., 200 msec)
-----	-----	--- msec

Conclusions

Conclusions

- For an application of this level of sophistication we currently have no analytical tools to quantify the expected system performance for a *realistic* model of the system *in its physical environment*
- Lacking such tools (which require development of new theory) the ability to run a simulation will be essential for the application designer
- Will the generated code actually fit in 128KB ? Some code reduction — at a yet unknown cost in performance — is possible by simplifying parts of the approach

Conclusions (continued)

- The tracking application appears to be fully scalable — the crux being defining an “Optimality Metric” not precluding full scalability
- We believe that our solution is robust (resilient for transient failures; limited effect of localized permanent failures) but have no proof of this
- “Coordination” and “Time-Bounded Synthesis” have a fuzzy boundary; the distinction is not a principled one

Conclusions (continued 2)

- Interface between components is not a conventional API but describes and *names* information to be maintained (conceptual model: services are “daemons” that deposit info when and were it is needed
- Middleware-service algorithms need to be “decompiled” / reverse-engineered into a maintenance pattern
- The results of the exercise suggest that there may be a small set (perhaps even less than a dozen) of *basic* patterns from which almost all *fully scalable* middleware-service algorithms can be generated as a composition of instantiations