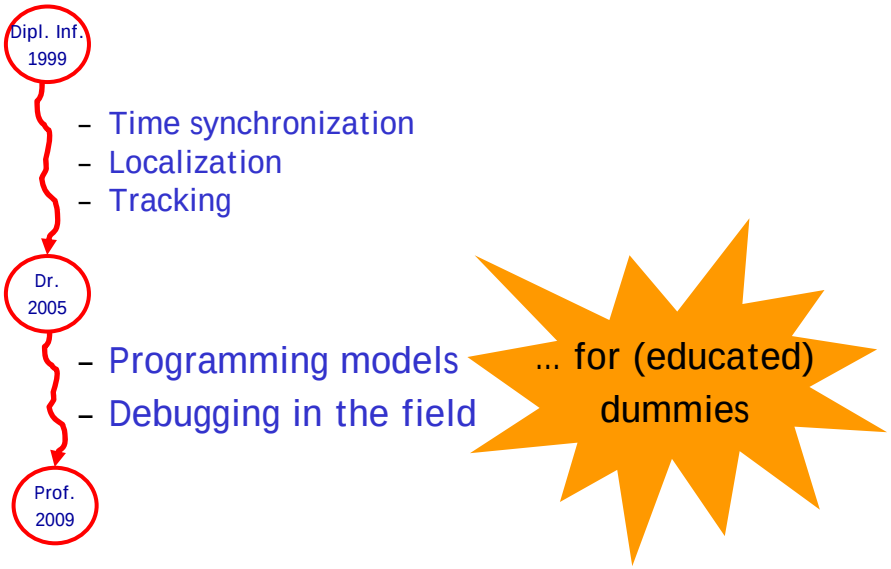
Sensornet Programming for (Educated) Dummies

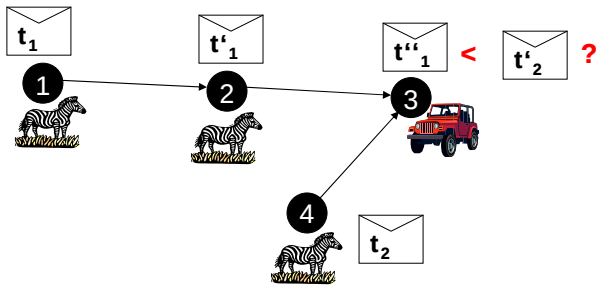
Kay Römer

The Golden Thread



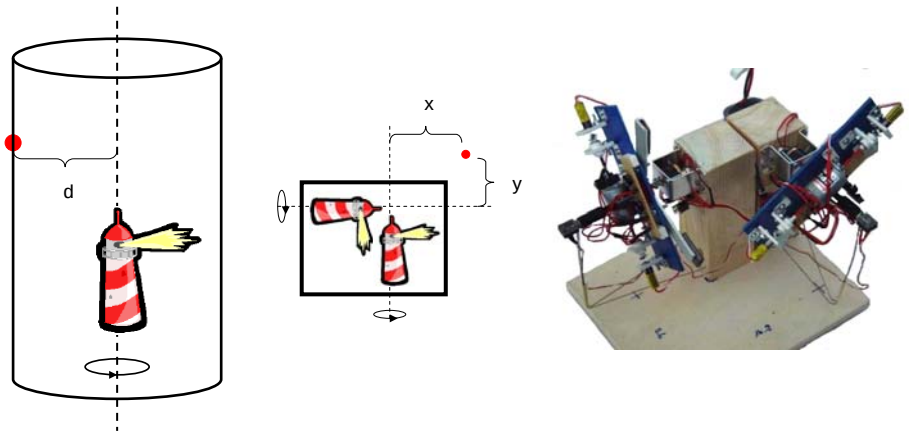
Sensornet Timesync

- Synchronization under sporadic connectivity
  - Nodes may connect only after they collected data
- Synchronize clock readings (timestamps) instead of clocks
- Timestamp = interval

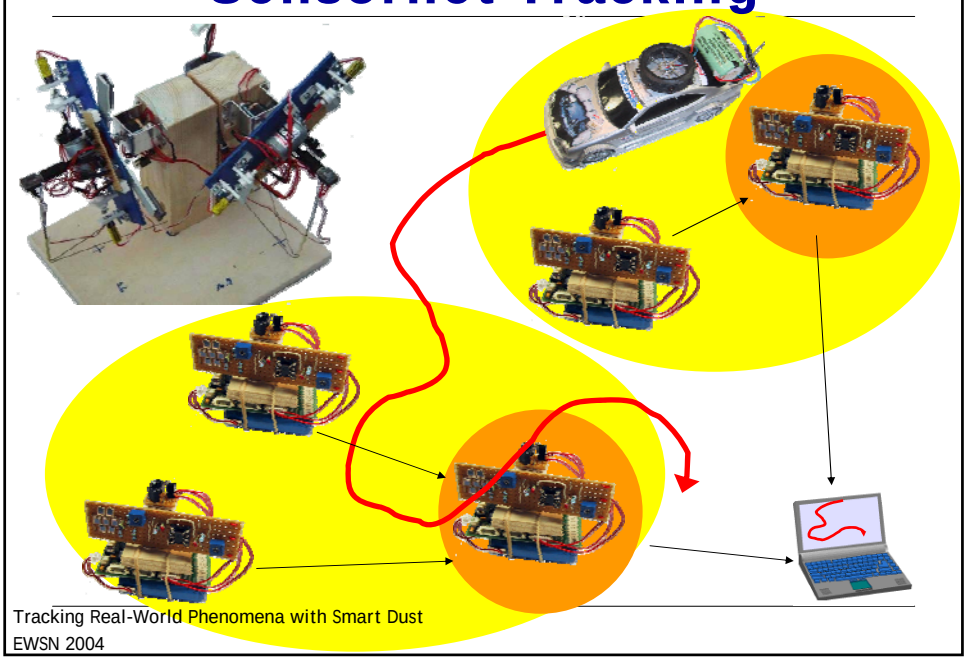


Sensornet Localization

- Localizing smart dust with a single anchor



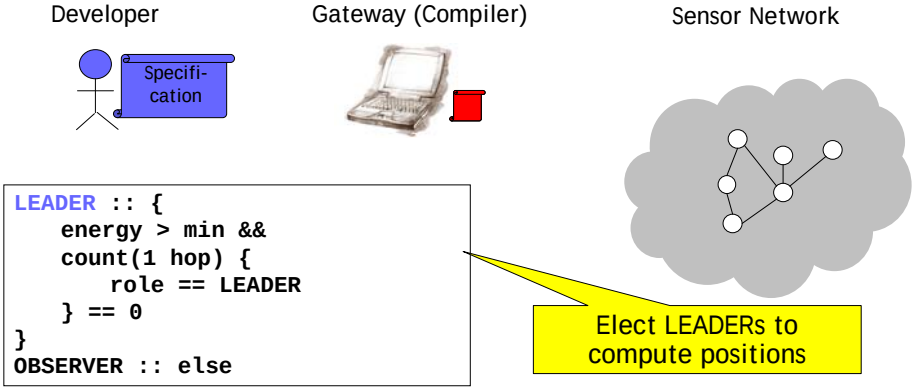
Sensornet Tracking



Tracking Real-World Phenomena with Smart Dust
EWSN 2004

Macroprogramming

- Simplify programming of sensor networks
  - Declarative, whole network
  - Generic Role Assignment



Algorithms for Generic Role Assignment in Wireless Sensor Networks
ACM Sensys 2005

Distributed Role Assignment

- Role initialization
  - role = UNDEFINED
- Property propagation
  - Extract scope
  - Scoped broadcast
- Constraint checking
  - Check all constraints locally
  - Assign first matching role
  - Re-propagate changed properties
- Scheduling
  - Randomized delays to break synchronization
- Notification
  - Notify application of „stable“ roles
- Distributed fix-point iteration
  - Convergence…

```
LEADER :: {
    count(1 hop) {
        role == LEADER
    } == 0
}
OBSERVER :: else
```

Role Initialization

- Distributed Role Assignment
  - All nodes start with role UNDEFINED
- Probabilistic role initialization
  - „Guess“ initial roles for each node
  - Repair wrong guesses with distributed algorithm
  - Hope: faster convergence
- Two variants
  - Use only static information
  - Use runtime information

Static Initialization

A priori knowledge

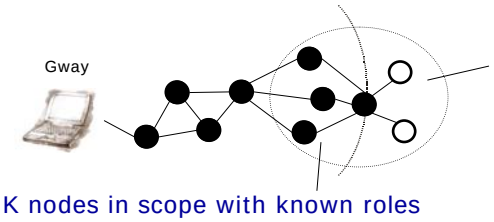
- Basic approach
  - Given: role specification, network density N
  - Compute: P[r] = P[node assumes role r]
  - Role init.: assign role r with prob. P[r]
- Translate spec. to equation system
  - P[L] = P[no neighbors are L]
    $= (1 - P[L])^N$
  - P[O] = 1 – P[L]
  - Solve for P[L], P[O]

```
L :: {
    count(1 hop) {
       role == L
    } == 0
}
O :: else
```

9

Dynamic Initialization

- Static doesn't work well in many situations
  - E.g., property values hard to predict offline
- Piggyback initialization on specification flood
  - Adjust probabilities
  - Use actual roles where known




```
L :: {
    count(1 hop) {
       role == L
    } == 0
}
O :: else
```

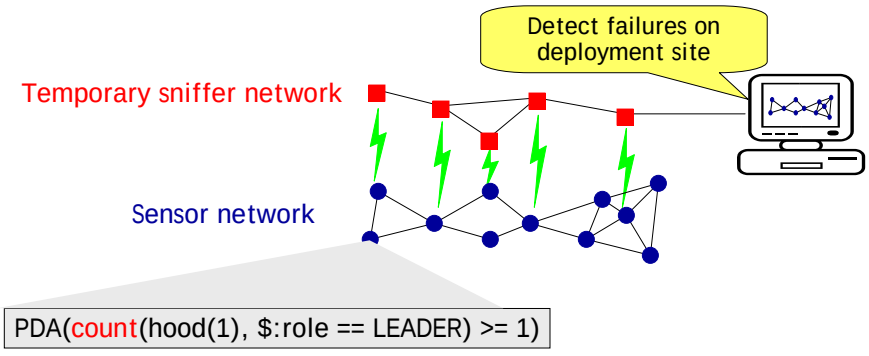
10

Dynamic Initialization

- Adjust probabilities
  - Similar equation system as for static
  - Replace probabilities with known info
  - $P[L] = (1 - P[L])^N$
  - $P'[L] = \underbrace{(1 - P[L])^U}_{\text{unknown}} \underbrace{\Pi_k(1 - L(k))}_{\text{known}}$
  - Where L(k) = 1 iff node k is LEADER
- Example
  - If one of the known neighbors is LEADER, then P'[L] = 0

Debugging

- Find failure causes in already deployed network
  - Passive distributed assertions



Passive Distributed Assertions

- Assertion: hypothesis about program state

```
int i;
...
assert(i > 50);
```

- Assertions over distributed program variables
  - Value of variable i should equal value of variable k on node 100

```
int i;
...
PDA(i == 100:k);
```

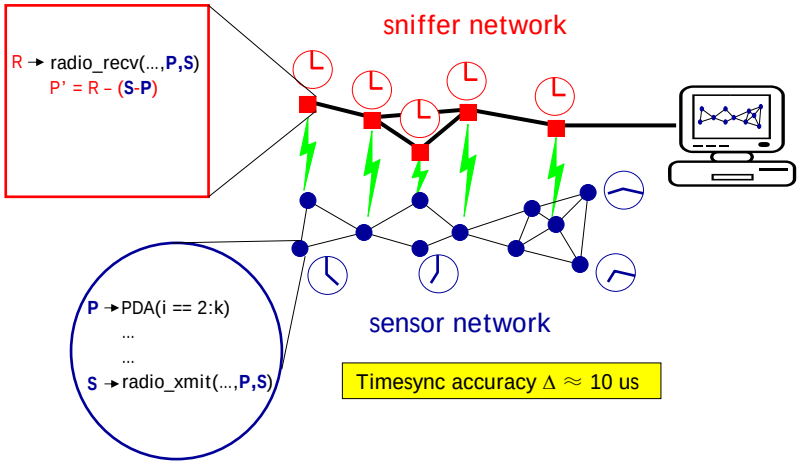
Variable k on node 100

- Checked by semi-passive inspection
  - Nodes broadcast assertions (**PDA msg**) and changes of relevant variables (**SNAP msg**)
  - Overheard by sniffer network

```
k = ...;
SNAP(k);
```


Passive Trace Synchronization

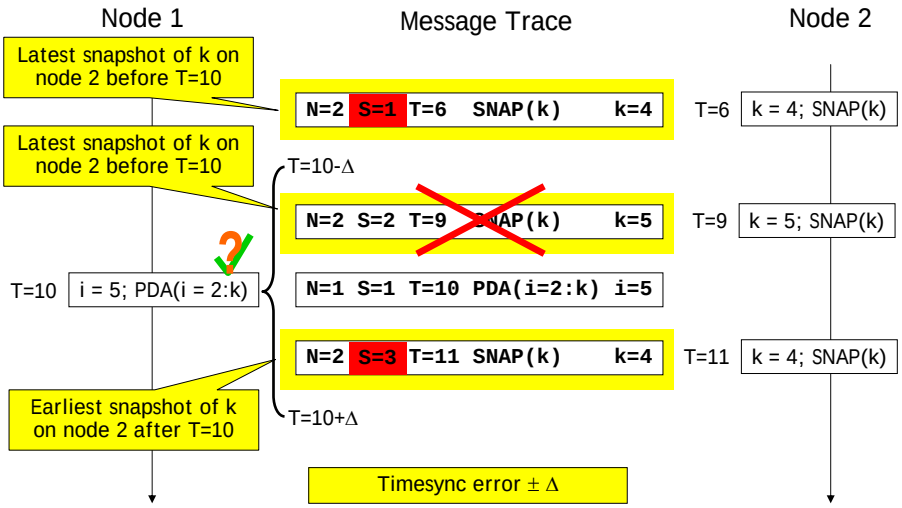
- Accurate global timestamp for PDA, SNAP *execution* (not message transmission)



Assertion Evaluation

- Backend evaluates trace
  - Reconstruct variable values from SNAP
  - Evaluate PDA on reconstructed values
- Key challenge: incomplete information
  - Missing messages
  - Inaccurate synchronization
- Evaluation result
  - OK: definitely holds
  - FAIL: definitely fails
  - UNDECIDABLE: either OK or FAIL

Assertion Evaluation Example



Summary

- Getting sensornets right is hard
  - Massively distributed
  - Environment is evil
  - Limited observability
- Need for abstractions to put sensornets into hands of (educated) dummies
  - Programming models
  - Debugging
  - Integration thereof
- Zero programming? Correct by design?