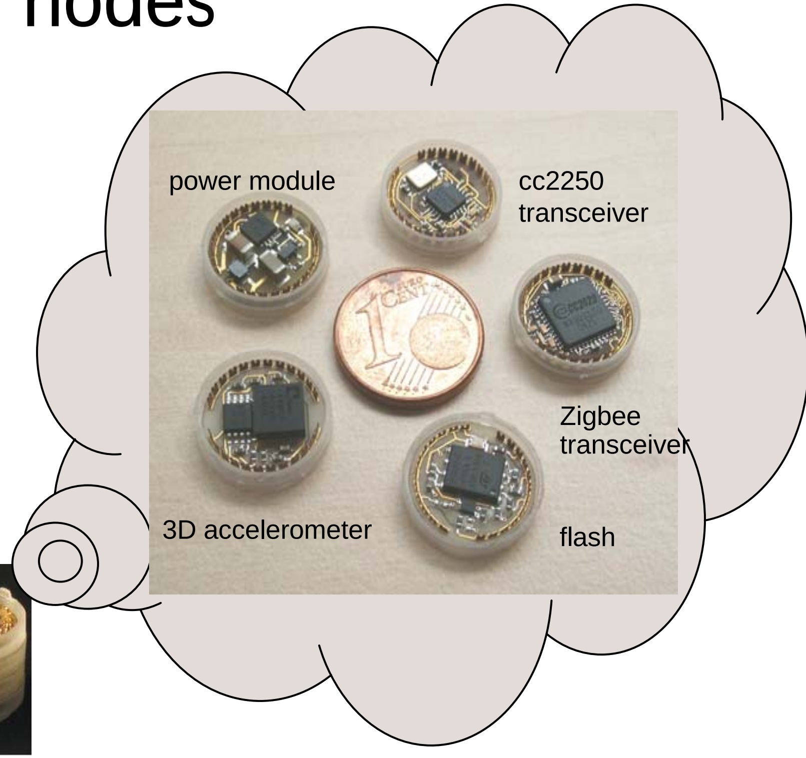
Wireless Sensor Networks: From Science to Reality

Kay Römer
ETH Zurich

Sensor Networks

- Ad hoc network of sensor nodes
  - Perceive (sensors)
  - Process (microcontroller)
  - Communicate (radio)
  - Autonomous power supply



modularization

miniaturization

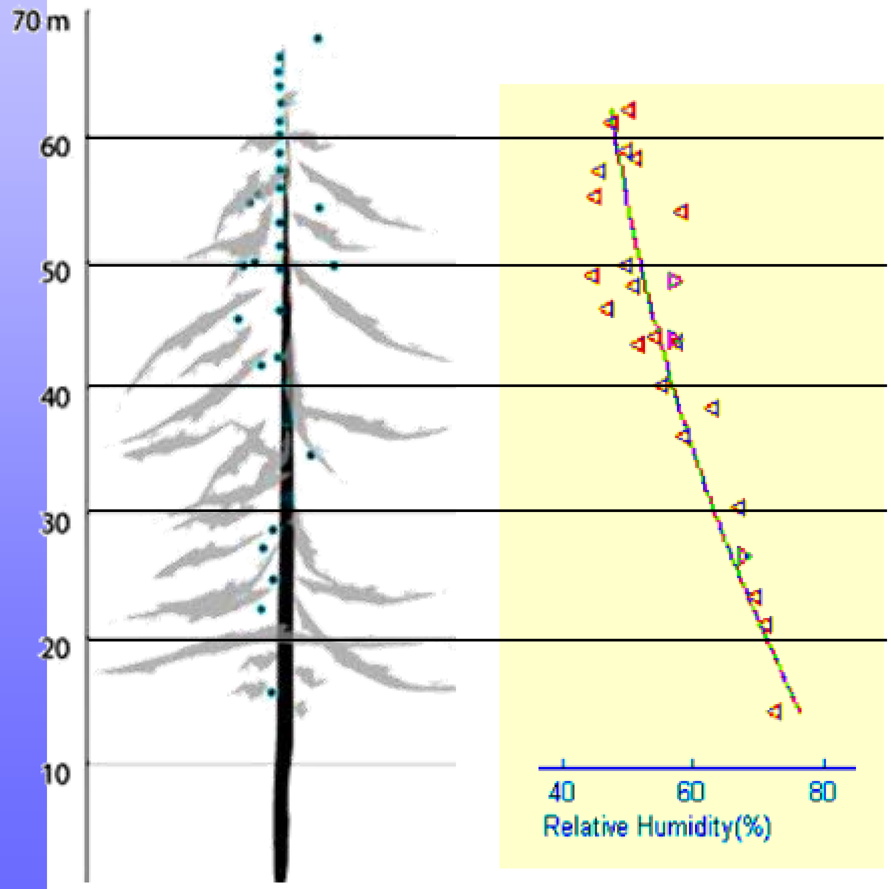
D. Culler [Berkeley]

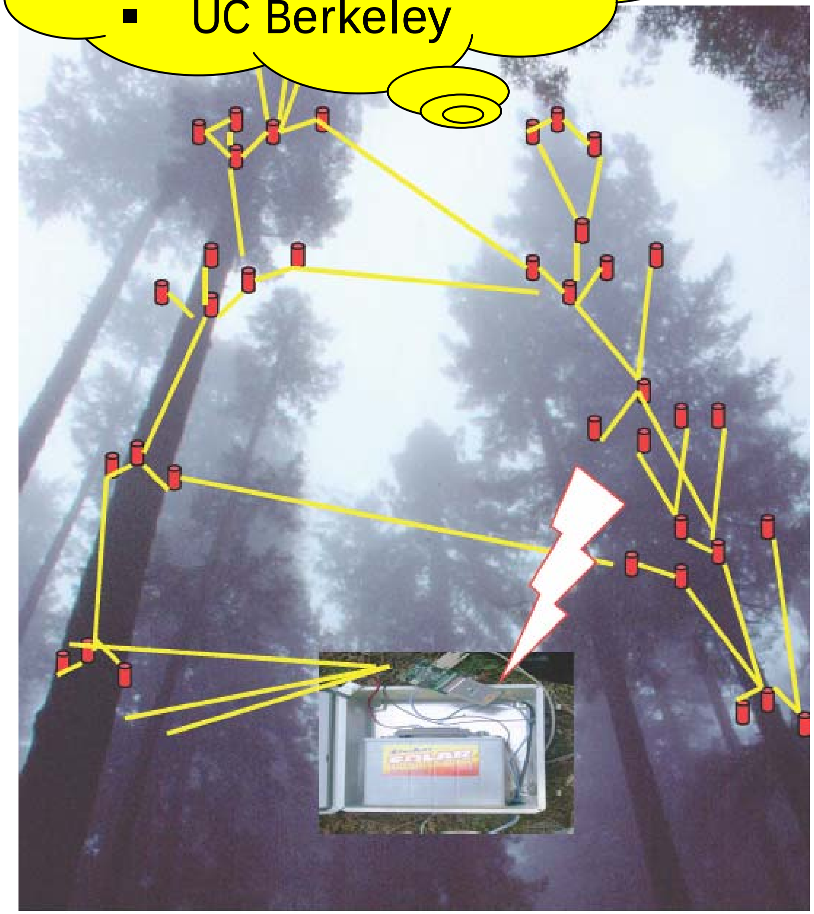
Application Visions

Reality #1

- Micro climate in Redwoods

- 44 days
- 50 nodes per tree
- UC Berkeley

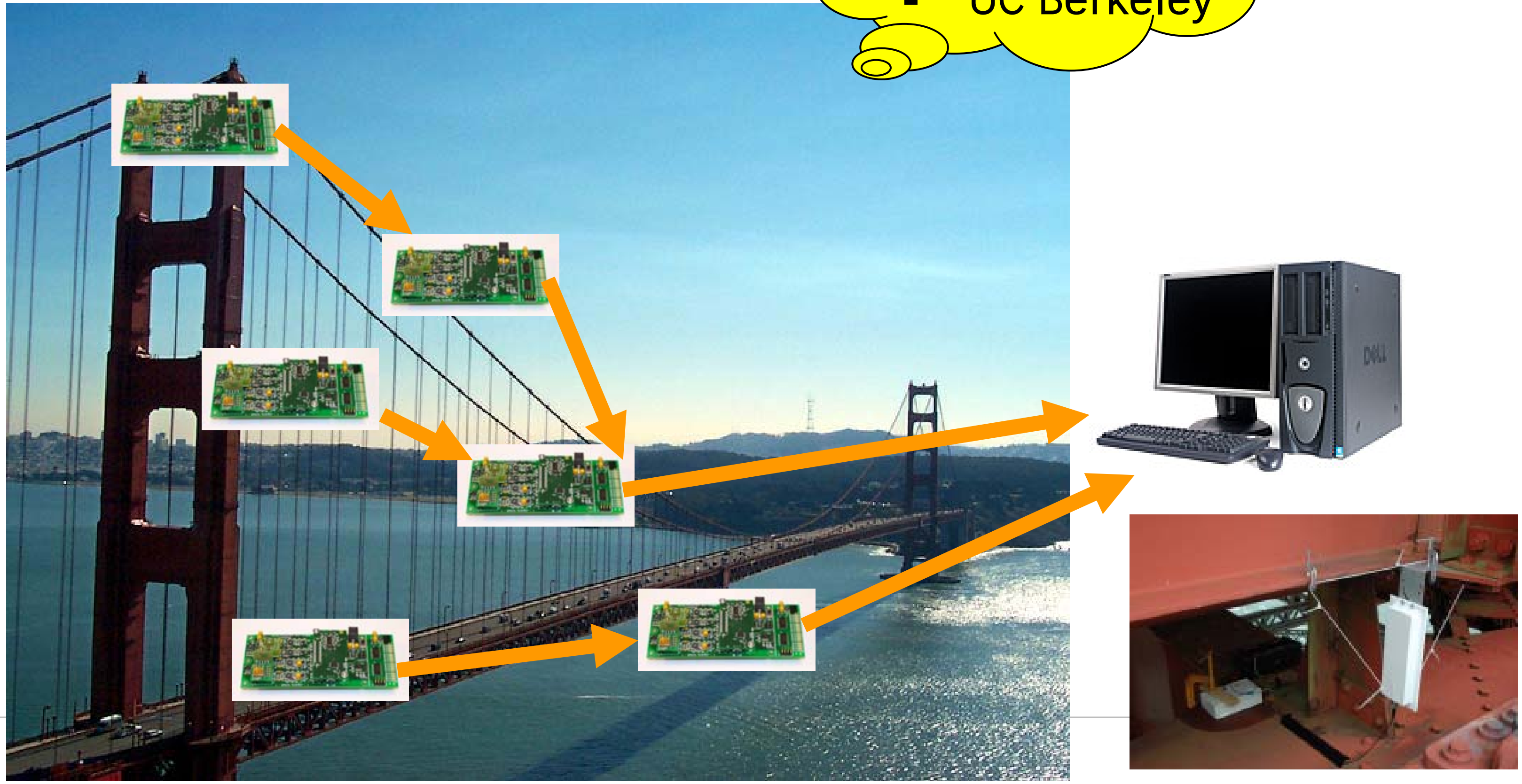




Reality #2

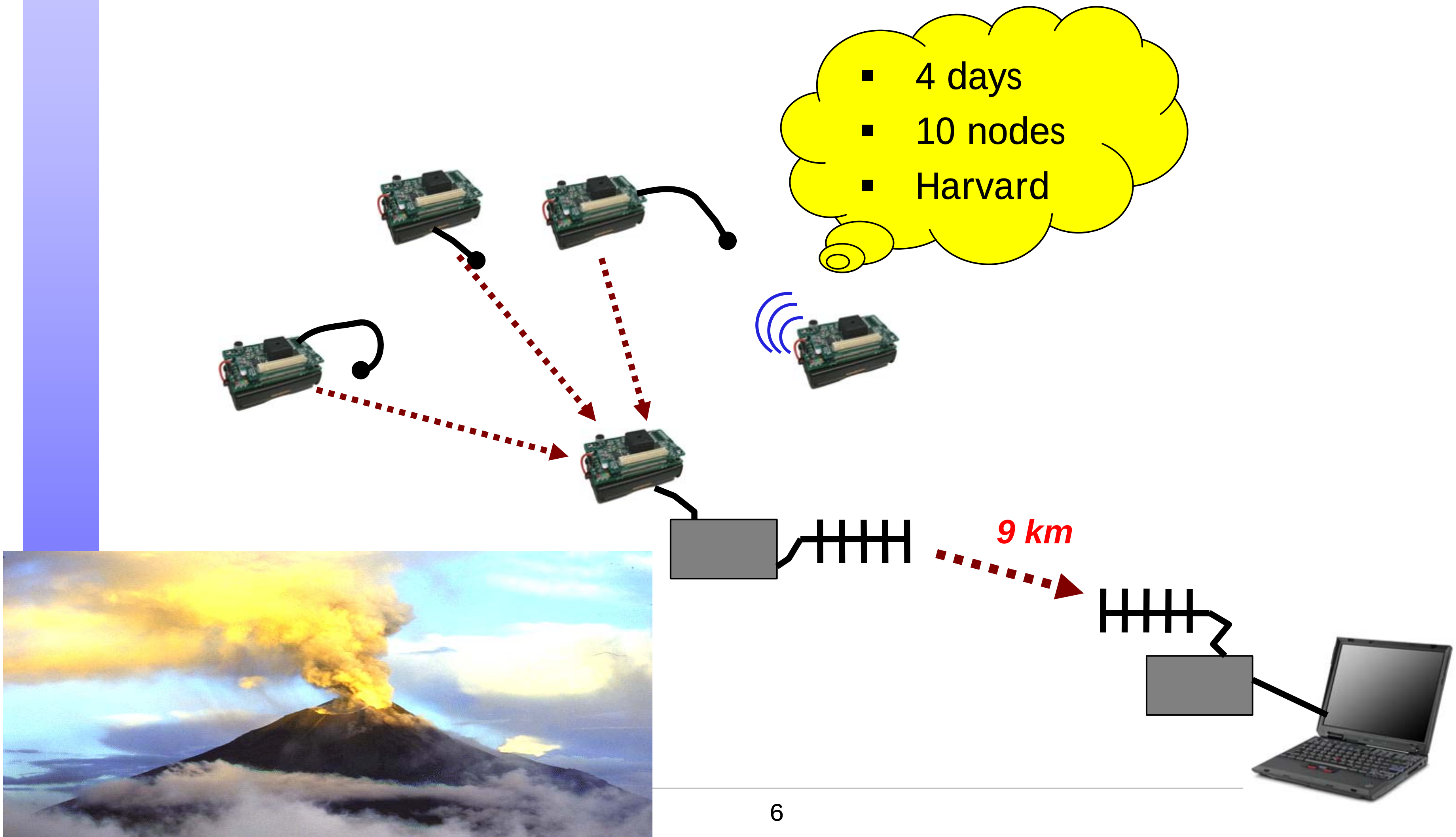
- Bridge vibrations due to wind / seismic

  - Days
  - 59 nodes
  - UC Berkeley



Reality #3

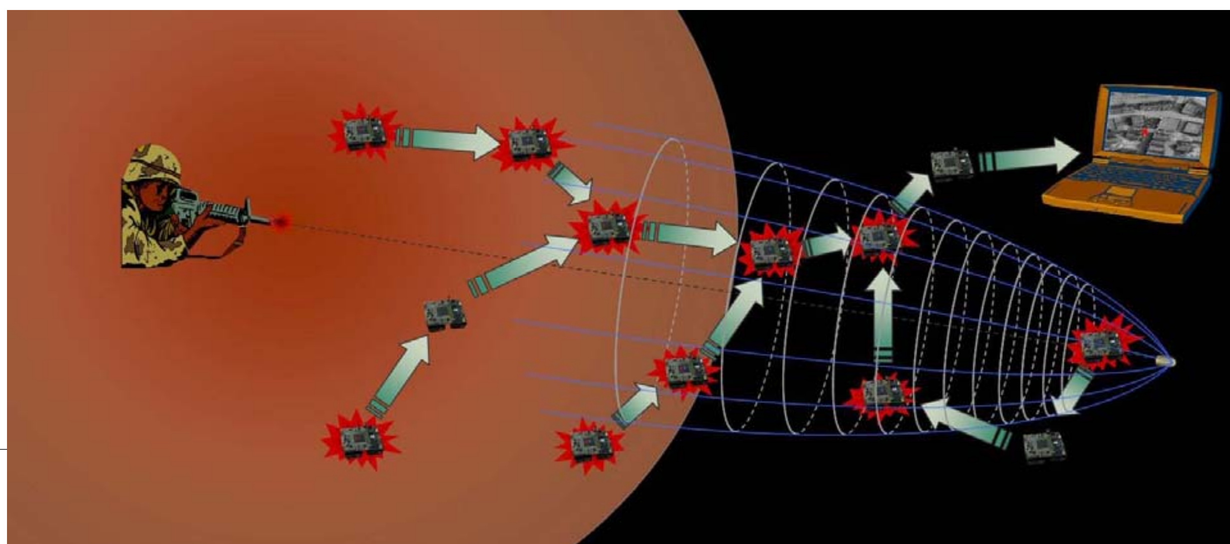
- Volcanic eruptions



Reality #4

- Sniper localization

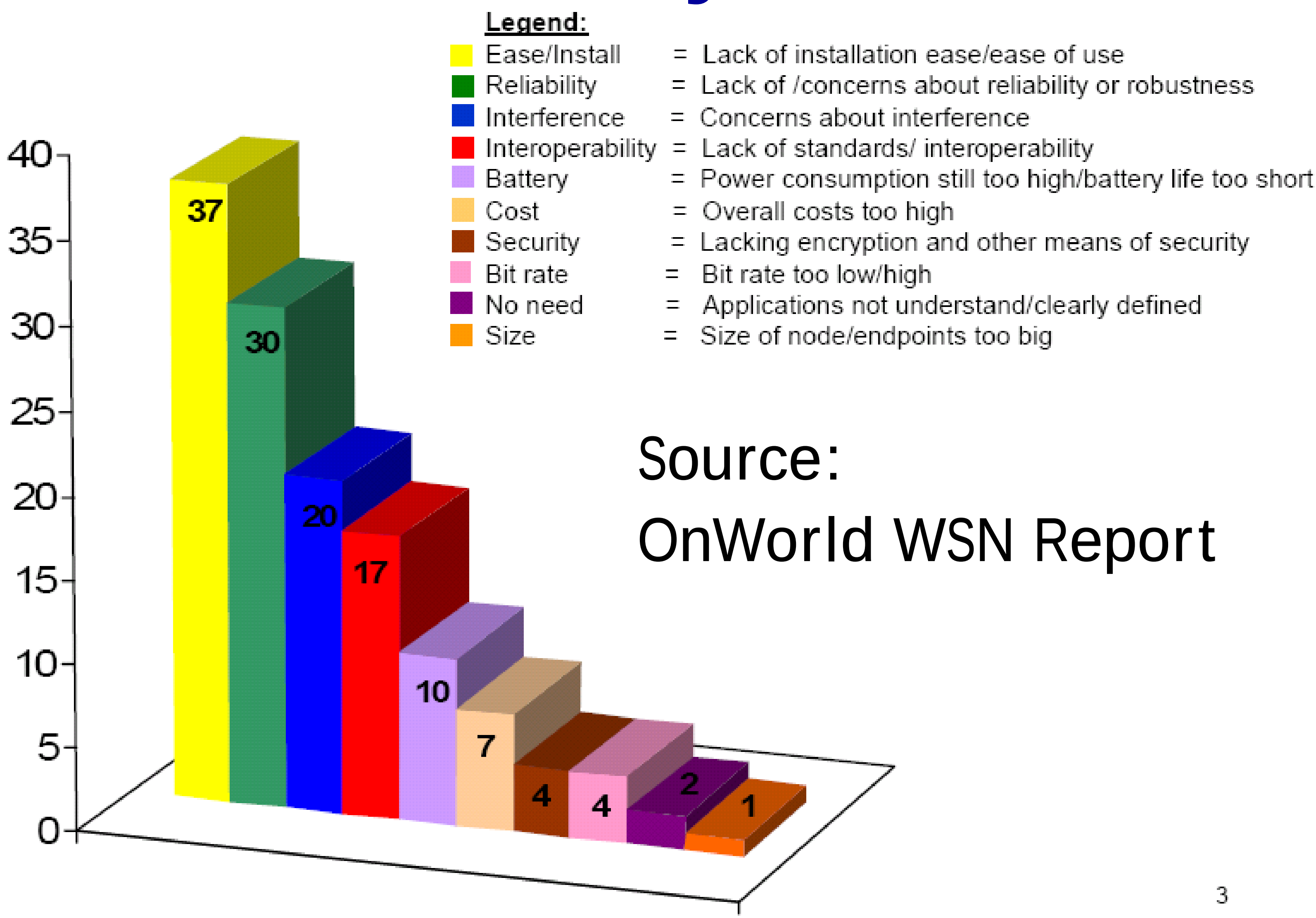
- Days
- 60 nodes
- Vanderbilt



Vision = Reality?

- Scientific experiments
  - Developed and deployed by experienced computer scientists
  - Small scale, short term
  - Supervised operation
- (Almost) no „real-world“ applications
  - Developed and deployed by application domain experts
  - Large scale, long term
  - Unattended operation

Why?



Source:
OnWorld WSN Report

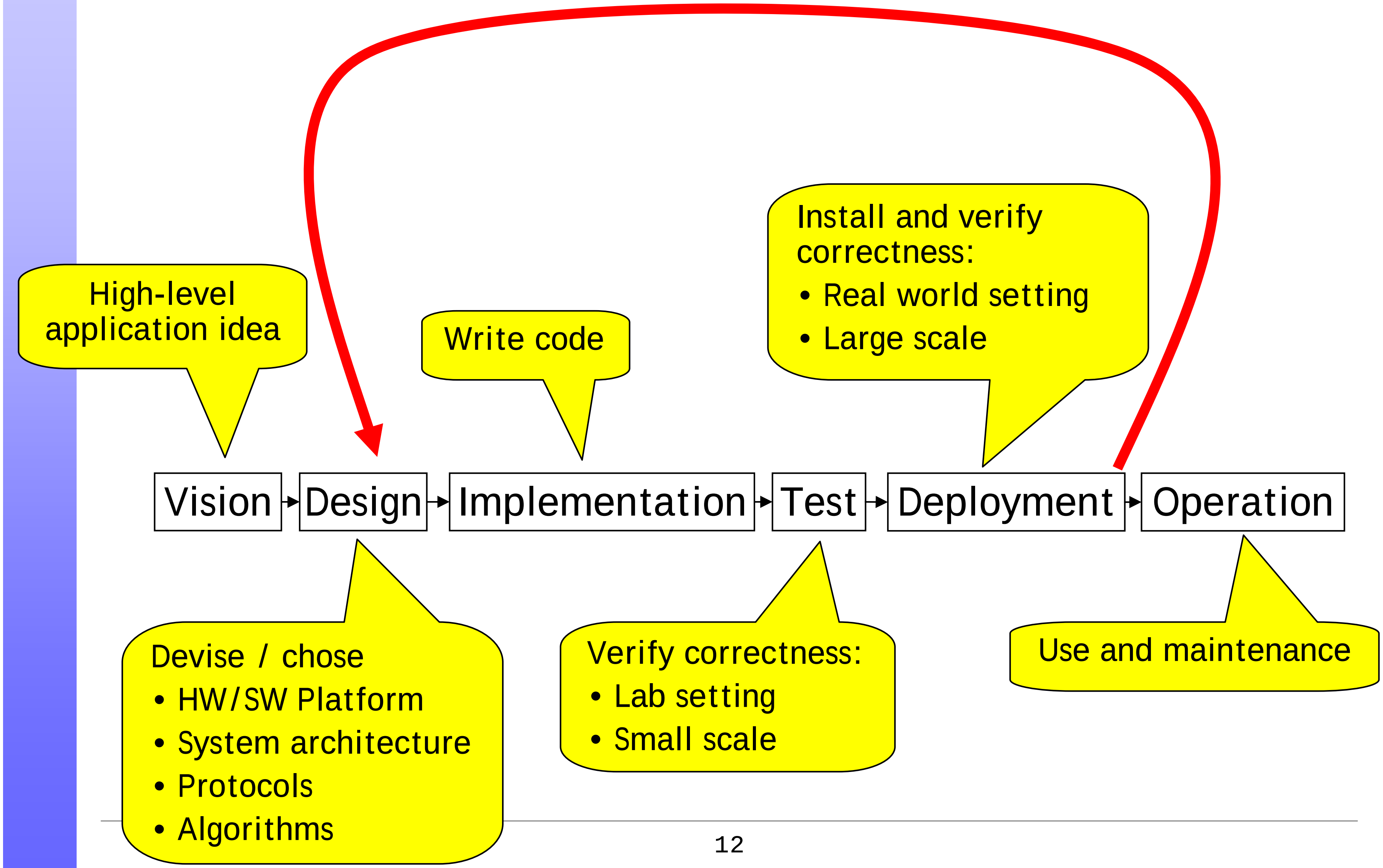
Ease of Use / Robustness

- Some exemplary citations from people who developed and deployed sensor networks
  - *Depends on individual skill of developers* [Cerpa01]
  - *Many iterations of system design / implementation required* [Mainwaring02]
  - *Involves significant manpower* [Hemingway04]
  - *Involves a certain amount of luck* [Szewczyk04]
  - *Everything that could go wrong did go wrong* [Langendoen06]

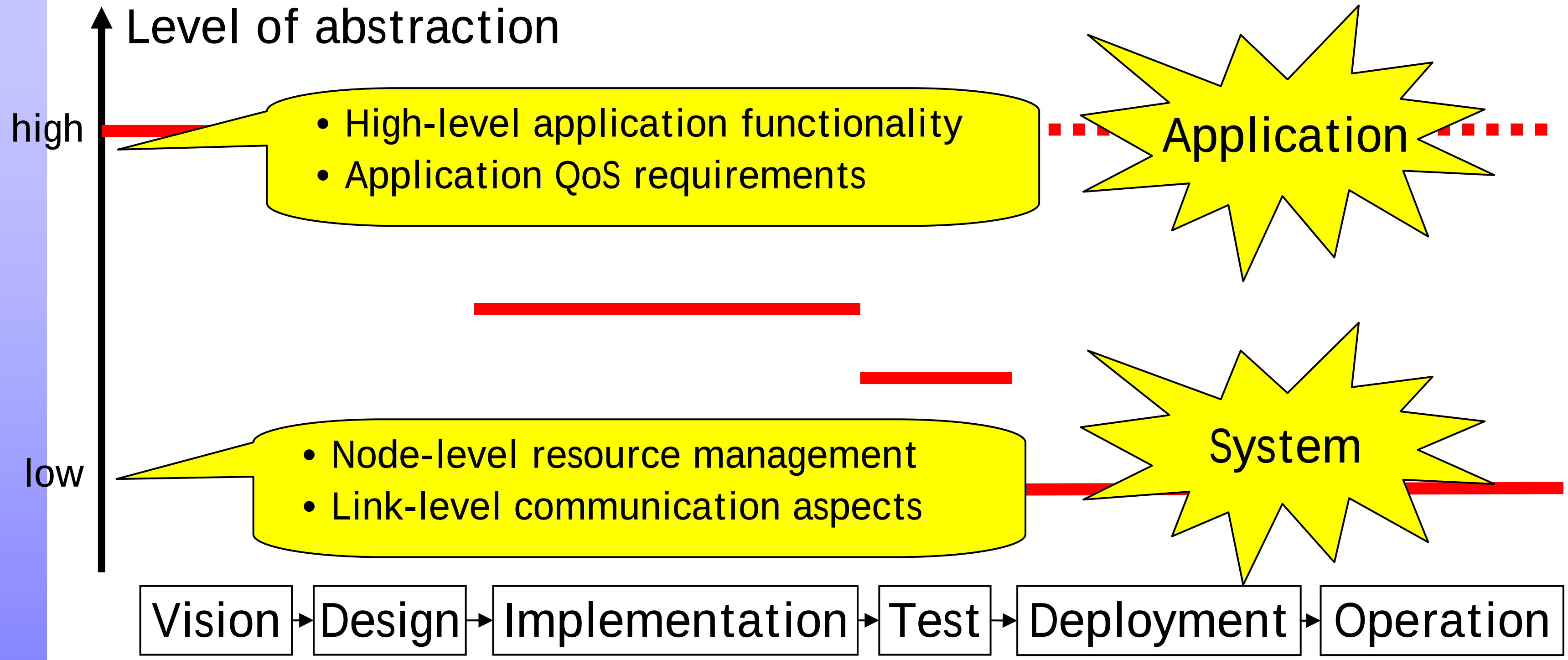
What is different?

- Worst of distributed and embedded worlds
- Dynamic, unreliable networks
  - Links come and go
  - Nodes come and go
  - Mobility
- Constrained resources
  - Simple OS
  - System-centric programming
  - Many competing optimization goals
  - Limited visibility
- Application development and deployment very difficult!

WSN Application Lifecycle

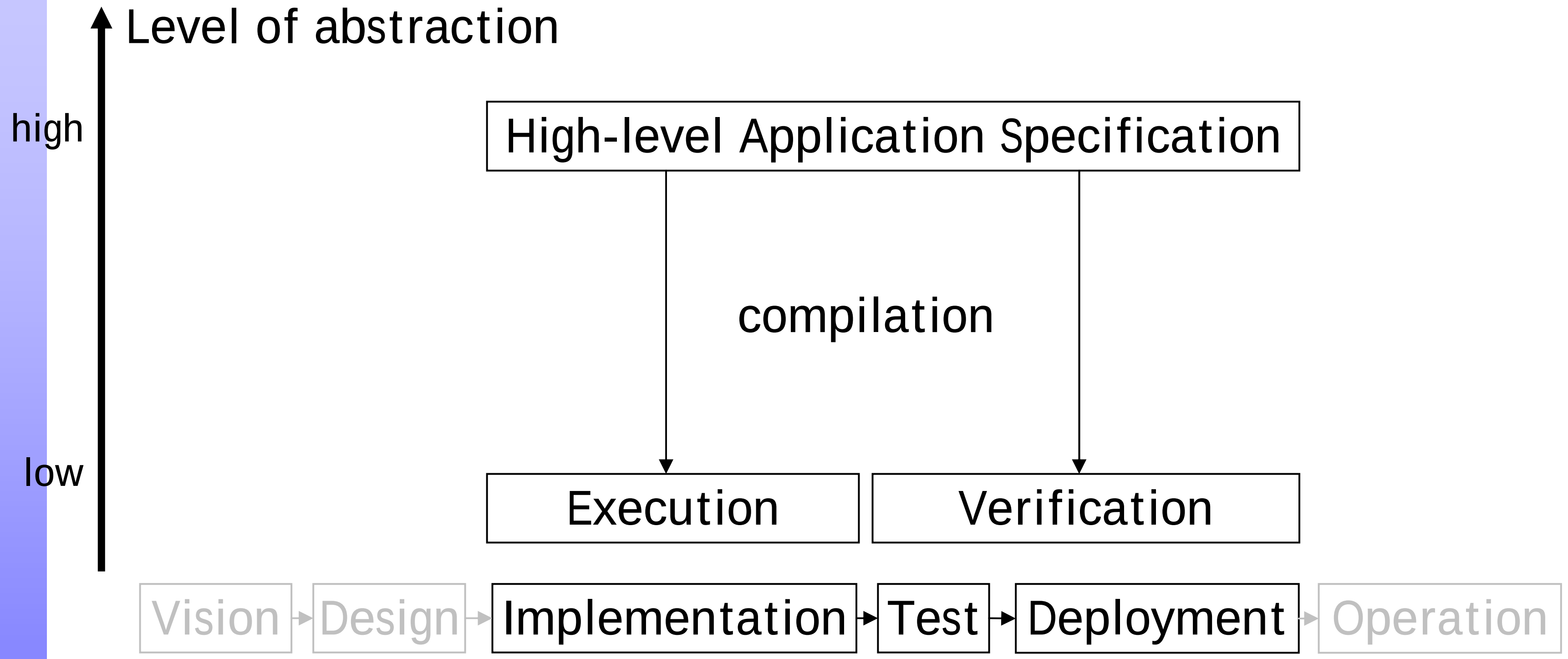


Abstraction



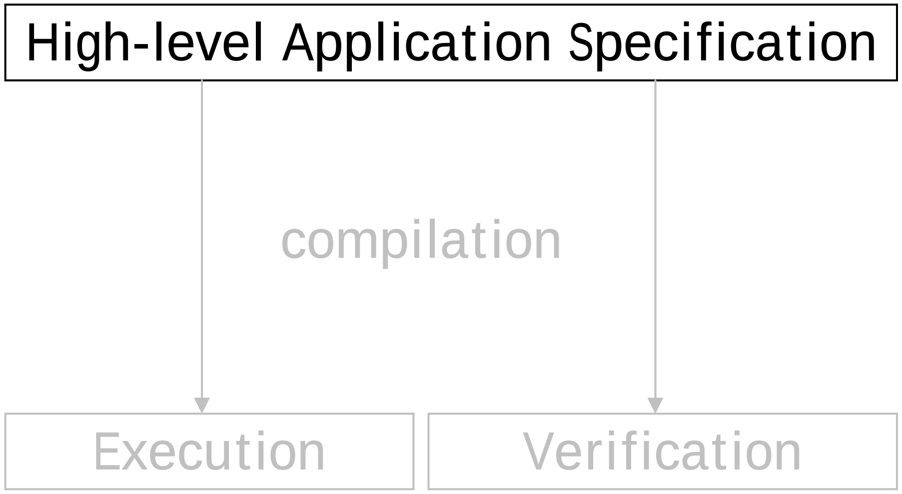
- Isolated solutions for lifecycle phases
- Low and varying level of abstraction

An Integrated Approach



- A single high-level application specification drives
  - Implementation
  - Test
  - Deployment

High-level Specification



Desirable Languages

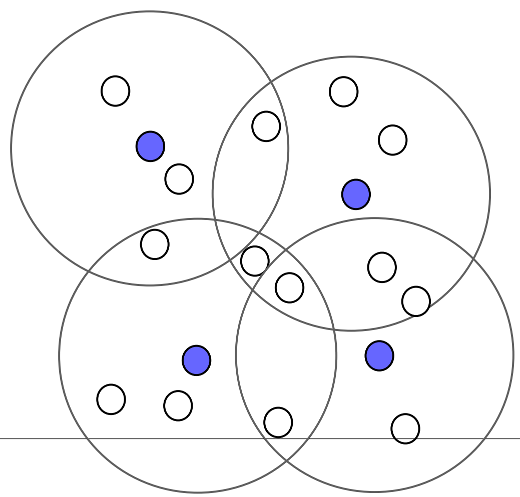
- Declarative
  - Specify desired application behavior
  - Not: how to achieve this behavior
- Node ensembles
  - Specify behavior of a group of nodes or whole network
  - Not: individual nodes

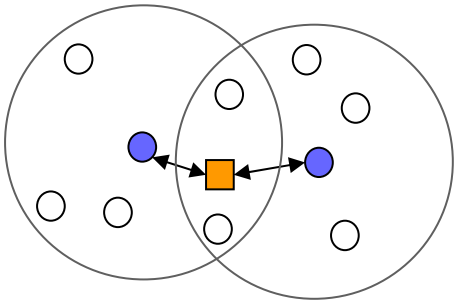
„# neighbors with a temperature sensor“

1. Send request containing …
2. Wait 10 seconds
3. Perhaps retransmit
4. Count distinct replies

Example: Role Assignment

- Support for self-configuration
  - Initially, all nodes are (more or less) identical
  - Nodes take on specific functions
- Examples
  - Clustering: HEAD, SLAVE, GATEWAY
  - Coverage: ON, OFF
  - Aggregation: SOURCE, AGGREGATOR





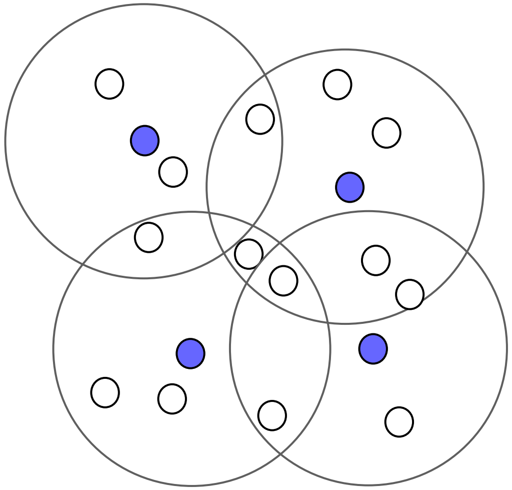
Generic Role Assignment

- Supports automatic assignment of roles to sensor nodes
  - Maintain valid assignment as network changes
- Declarative role specifications
  - Definition of roles
  - Definition of rules (constraints) for assignment
  - Rules refer to node properties

```
battery = 80%
pos = (12.3, 3.4)
role = ON
…
```

Coverage [cf. PEAS]

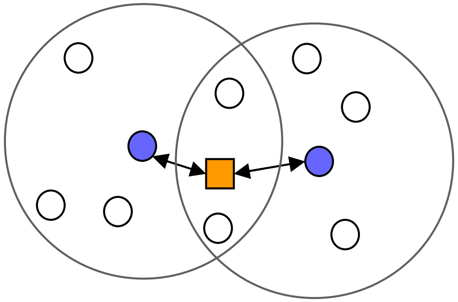
```
ON :: {
   battery >= threshold &&
   count(1 hop) {
       role == ON &&
       dist(pos, super.pos) < R
   } == 0
}
OFF :: else
```



- count(scope) { pred }
  - Counts nodes matching *pred* within *scope*
  - *super.x* equals property x of referring node

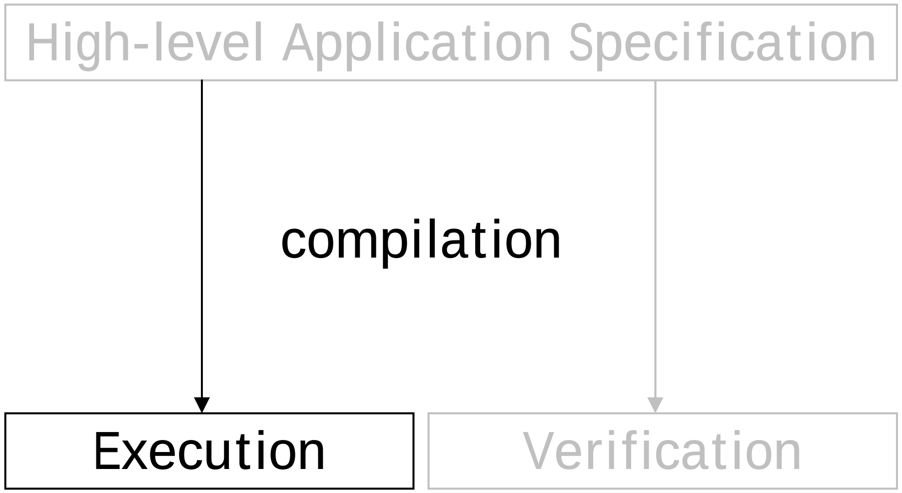
Clustering [cf. Passive Clustering]

```
CLUSTERHEAD :: {
    count(1 hop) {
        role == CLUSTERHEAD
    } == 0 }
GATEWAY :: {
    cheads == retrieve(1 hop, 2) {
        role == CLUSTERHEAD
    } &&
    count(2 hops) {
        role == GATEWAY &&
        cheads == super.cheads
    } == 0 }
SLAVE :: else
```



- retrieve(scope, num) { pred } == cheads
  - At least *num* nodes in *scope* must fulfil *pred*
  - Bind the 2 nodes to cheads

Execution



- Map high-level application functionality to node-level behavior
  - Resource constraints
  - Network dynamics

Distributed Algorithm

- Property propagation
  - Derive scope
  - Scoped broadcast
- Rule evaluation
  - Evaluate all rules locally
  - Assign first matching role
  - Re-propagate changed properties
- Scheduling
  - Random delays to break synchronization
- Notification
  - Notify application of „stable roles“
- Distributed fix-point iteration
  - In practice very few iterations (see paper)

```
ON :: {
    count(1 hop) {
        role == ON
    } == 0 }
OFF :: else
```

Role Initialization

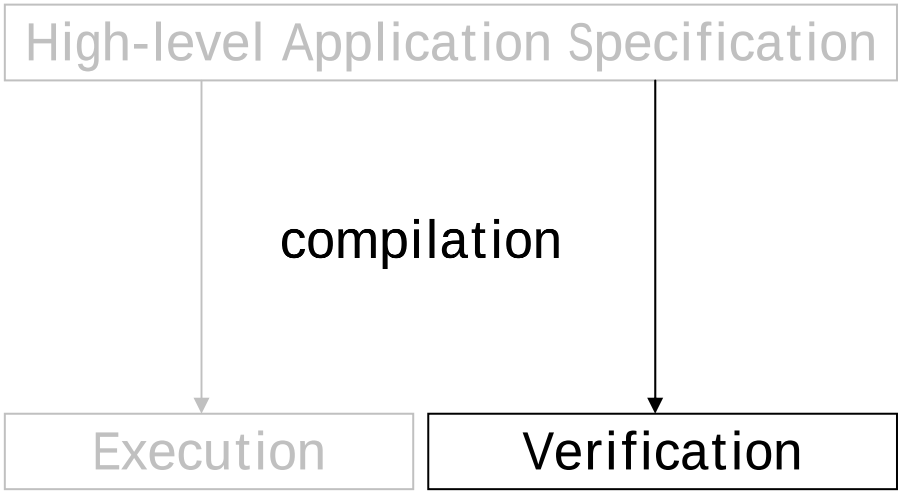
- Base algorithm
  - All nodes start with role UNDEFINED
- Probabilistic role initialization
  - „Guess“ initial roles for each node
  - Repair wrong guesses with base algorithm
  - Goal: faster convergence
- Two variants
  - Use only static information
  - Use runtime information (see paper)

Static Initialization

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

```
ON :: {
   count(1 hop) {
      role == ON
   } == 0 }
OFF :: else
```

Verification



- Verify system behavior against high-level specification
  - Resource constraints
  - Limited visibility of network state

Verification Challenges

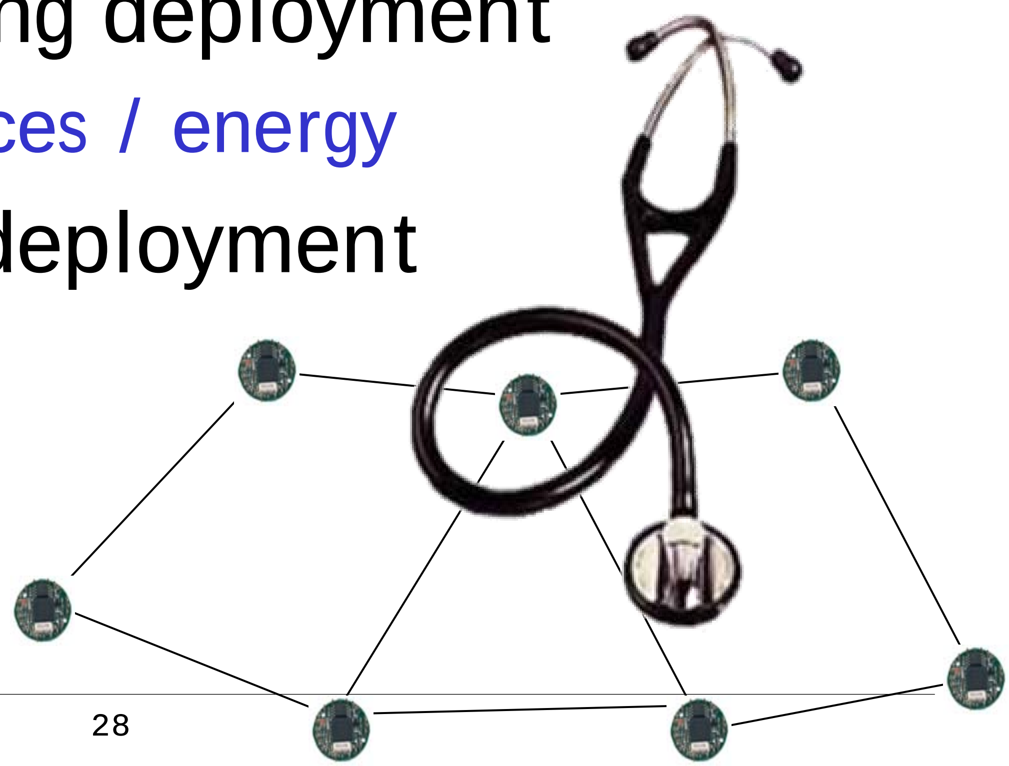
- Not a binary YES/NO answer
  - If NO, what and where is the problem?
- Verification of deployed network
  - Behavior differs to lab setting due to radio channel, sensor input, physical strain
- Key challenge: limited visibility of the network state
  - Once deployed, how can we access the state of nodes?
  - Limited resources: no space/bandwidth for verification
  - Heisenberg effect: measurement changes system behavior

Passive Verification

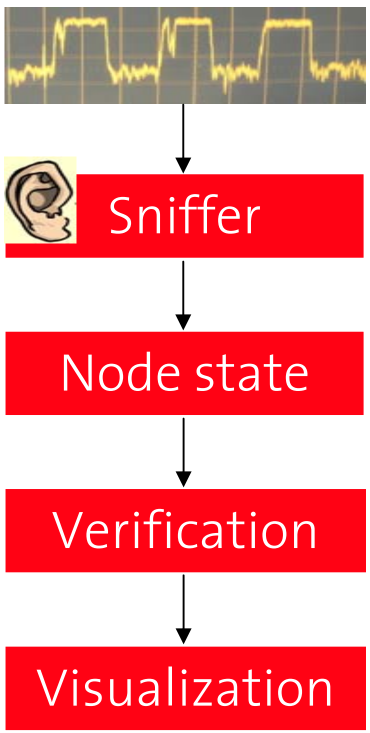
- Wireless traffic reveals parts of network state
  - Message contents (e.g., node role)
  - Message timing
- Approach: overhear network traffic
  - Pro: No modification of sensor network
  - Con: Additional hardware, incomplete information

A Stethoscope for WSN

- A tool to support passive verification of sensor networks
- Co-deployed with WSN
- Only active during deployment
  - Plentiful resources / energy
- Removed after deployment
  - Reuse for other deployments



Stethoscope Architecture



WSN radio communication

captures and decodes packets

infer node state from packets

of node states with high-level specification

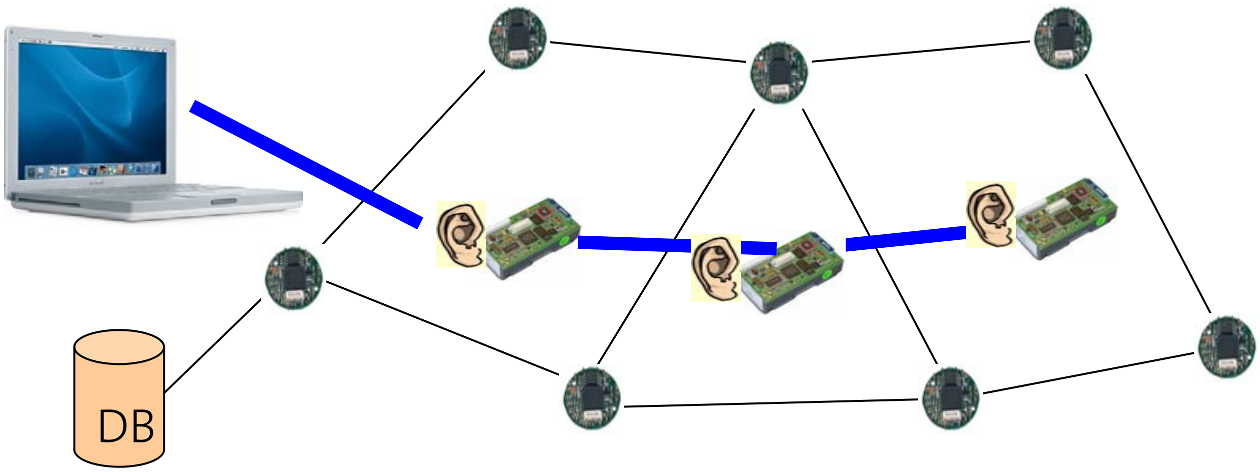
of node states and verification results

Sniffer

- Additional node with compatible radio
  - Always on to capture all packets without participating in MAC protocol (e.g., sleep scheduling)
  - Placed next to WSN
- Forward packet stream to base station
  - For centralized evaluation

Sniffer Network

- Single sniffer cannot observe complete WSN
  - Network of sniffer nodes (synchronized)
- Sniffer nodes have a second radio
  - High-bandwidth, robust (Bluetooth, WLAN, cable, …)
  - Free of interference with WSN radio



Node State Inference

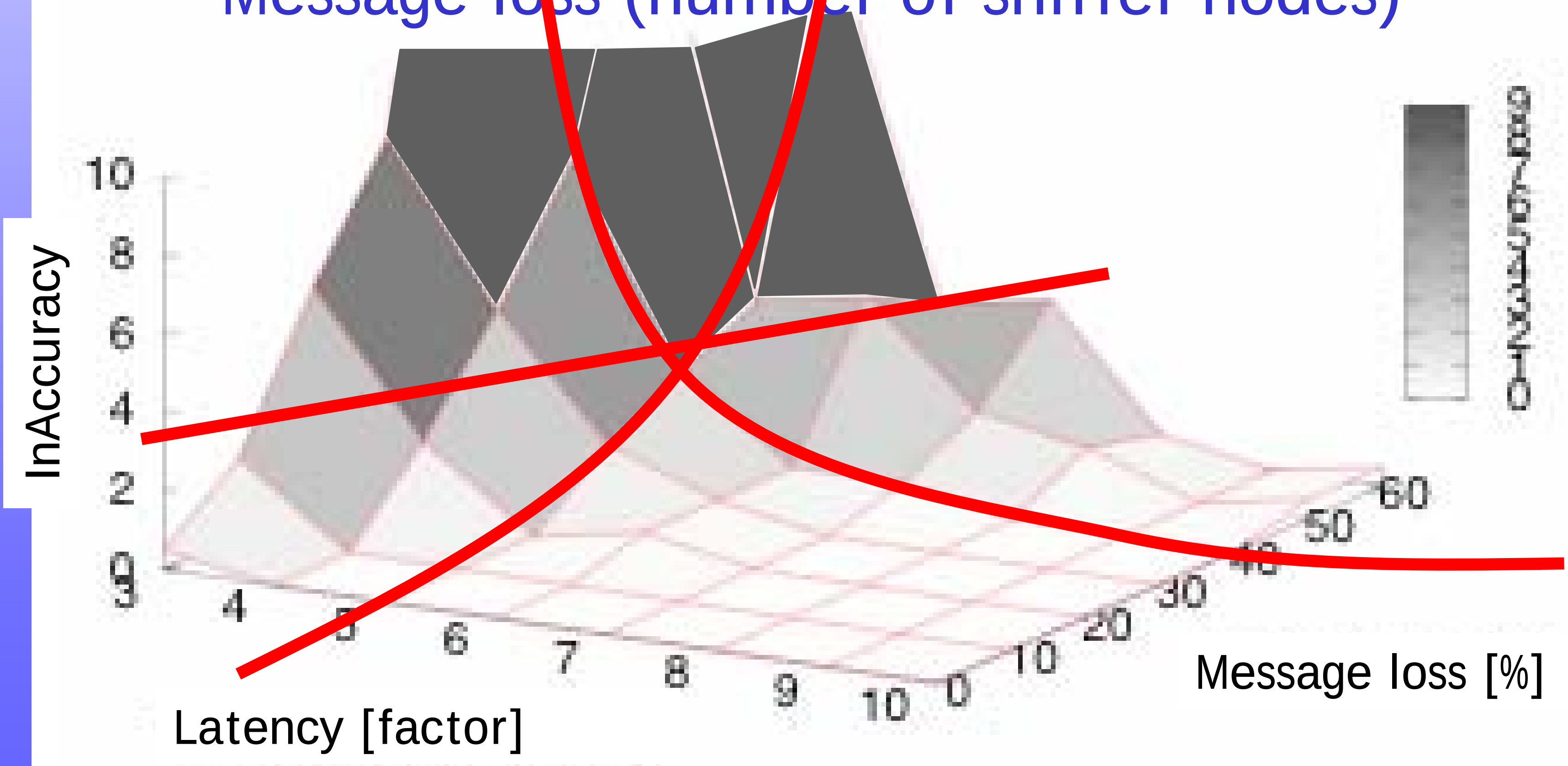
- Infer relevant state of individual nodes from overheard messages
  - E.g., role of each node, network neighbors
- Also basic node state
  - Node death: no messages
  - Node reboot: seq number reset
- Key problem: incomplete information
  - Message loss
  - Missing information in WSN protocol

Incomplete Information

- Missing information (e.g., neighbors)
  - Generation of protocols from high-level spec under our control
  - We are free to include information in protocol as long as it is small enough
- Message loss
  - Cannot be avoided, but detected!
  - Sequence number in each message (received n, but not n-1)
  - Timing irregularities (expected transmission at t not received)

Fundamental Trade-Offs

- Main parameters of state inference
  - Accuracy (correctness of inferred state)
  - Latency (delay of state inference)
  - Message loss (number of sniffer nodes)

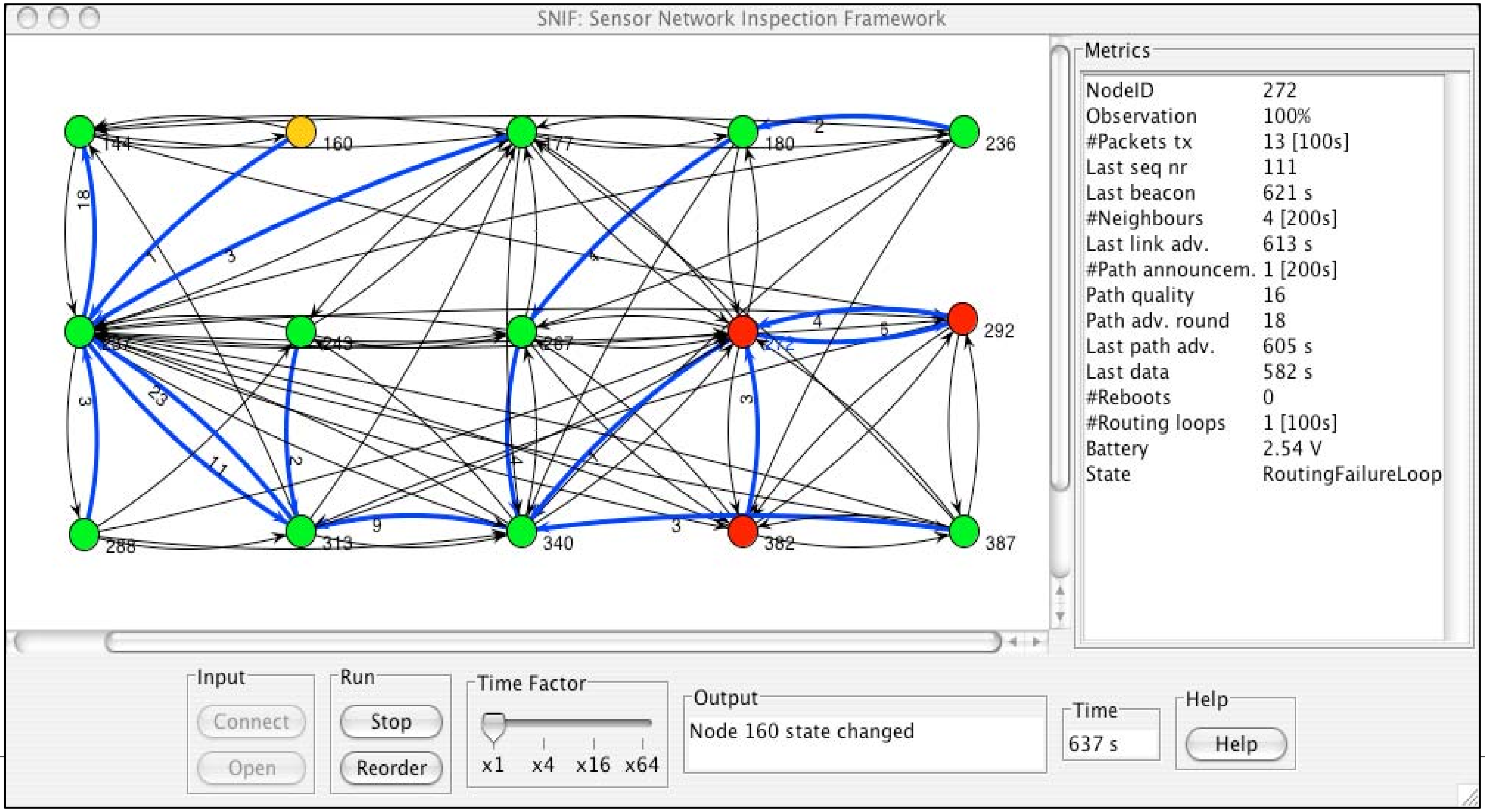


Verification

- Map high-level specification to checker
  - Deal with incomplete information
  - Distinguish errors and potential errors
- Verification easy compared with execution
  - Centralized instead of distributed
  - Checking instead of producing a role assignment

Visualization

- Node state
- Correctness at node level
  - OK, Warning, Error



Summary

- Gap between application visions and reality
- Two reason: ease of use / robustness
- WSN application lifecycle
  - Low level of abstraction
  - Isolated solutions
- An integrated approach
  - Single high-level application specification drives implementation, test, and deployment
- Example: Generic role assignment
  - Declarative specification language
  - Role assignment algorithms
  - Passive verification

Thanks!

- More details:
  - *Algorithms for Generic Role Assignment in Sensor Networks*, Sensys 2005.
  - *Passive Inspection of Sensor Networks*, DCOSS 2007.