Mission maybe possible:
Improving the programming model for wireless sensor networks

Simon Dobson
School of Computer Science, University of St Andrews UK

simon.dobson@st-andrews.ac.uk
http://www.simondobson.org

Introduction

- Sensor networks for the people
  - Concerned with science and engineering, *not* computing
  - How can we place sensing capabilities in the hands of the scientists and engineers most knowledgeable about the “missions” they're engaged in?
- My aim:
  - What makes sensor network programming different
  - Some desiderata and work-in-progress on mission-oriented programming

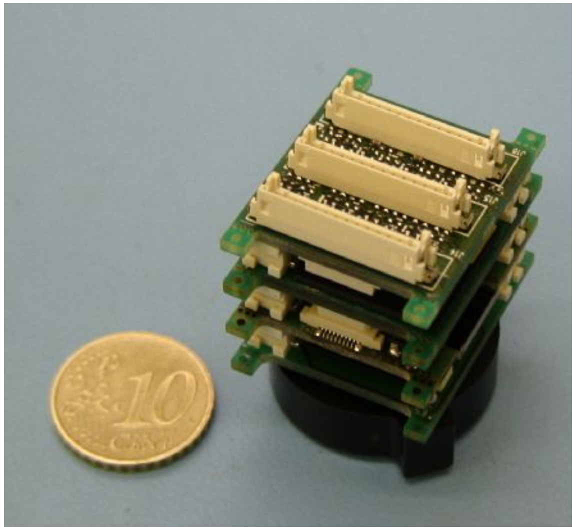
Will include no results, insights or hard conclusions...

Part I

The wider significance of sensor networks

Sensor and sense-ability

- The most exciting new frontier
  - Active data collection
  - Computing and communications
  - Tiny, low-power
  - Network them together to get capabilities
- Little or no direct user input
  - The environment is the interface



What this gives us – reach

- Embed computing into the real world, close to the phenomena of interest
  - Detailed, long-term collection
  - Work in hostile or unpleasant environments for long periods
  - A viable alternative to graduate students...

- Data capture is *active*
  - Change observations over time
  - Look for events, rather then just data

Of planetary importance

- Climate change, terrorism, pollution, food, energy, population growth, …

- Solutions
  - All depend on precise, timely, extensive data
  - ...and *only* computers let you collect, model and analyse the problems in a proper way
  - ...and therefore other subjects can't do *anything* unless backed by rigourous computer science
- So computer science is the only subject that can save the planet

The computer is the new microscope

Computer
Science

Microbiology

The third pillar

- Automation of observation and analysis
  - *Simulate* what we can't experiment on directly
  - *Mine* volumes of data for models
  - *Observe* phenomena at any scale
  - *Adapt* to what we see
  - Conceptualise change as *discrete processes*
  - Model *relationships* and *provenance*
  - Describe the *analysis* a scientist would make, allowing it to happen automatically in the field

Part II

How sensor networks differ from other systems we program

Data from all around

- Integrate a bewildering range of sensors
  - Precision
  - Accuracy
  - Timeliness
  - Robustness
  - Cost

- What does this do to programming?
  - GIGO

Not a new idea...

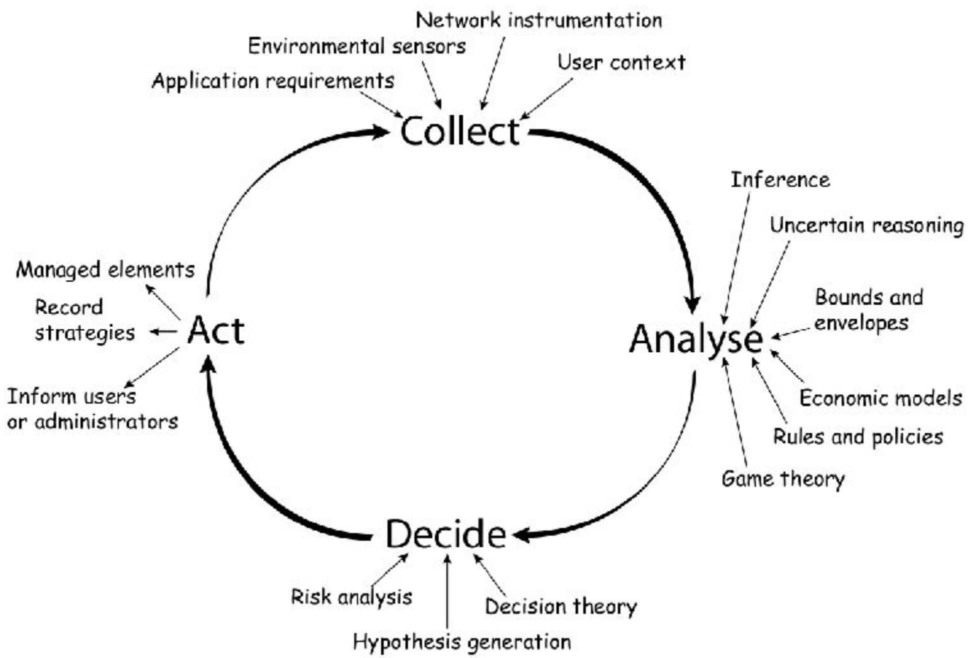
On two occasions I have been asked, "Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?" ... I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.

Charles Babbage. Passages from the Life of a Philosopher. 1864.

Quoted from http://en.wikipedia.org/wiki/Garbage_In,_Garbage_Out

Control

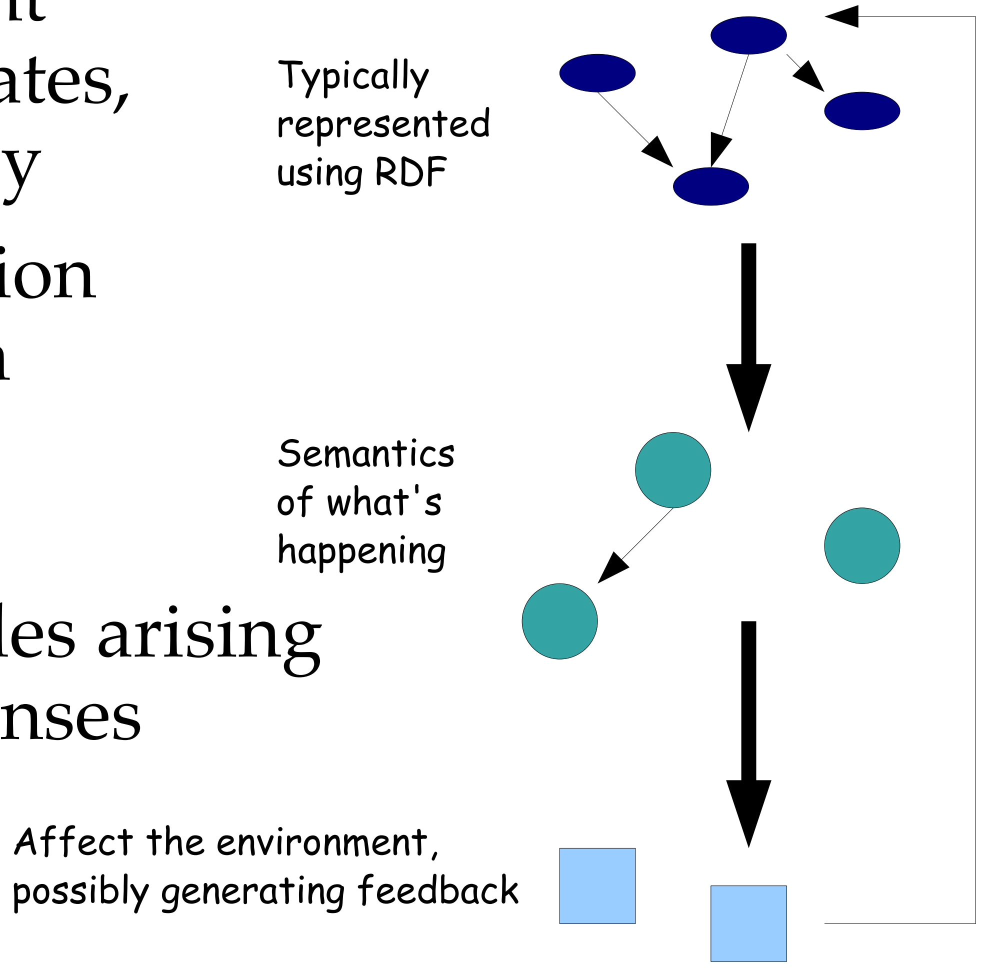
- Often need to do adaptive control in these environments
  - Change mode, duty cycle, processing, …
  - Ensure scientific (mission) goals are maintained across adaptations
- Basis for control is (imprecise) measurement



Dobson *et alia*. A survey of autonomic communications. ACM Trans. Auto. Adapt. Sys **1**(2). 2006.

Context and situations

- *Context*: the environment in which a system operates, understood symbolically
- *Situation*: an interpretation of the current context in terms of an expectation model of the world
- *Behaviour*: the observables arising from the system's responses

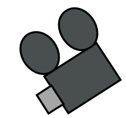


Sensor fusion

- Combine evidence from different sources
- Models of what we *expect* to happen
- Situation recognition

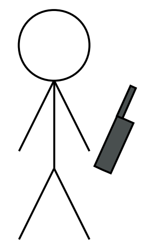
Diary says he should be here

...but he doesn't keep it completely up to date



Camera sees him here

...but he's got a really average face

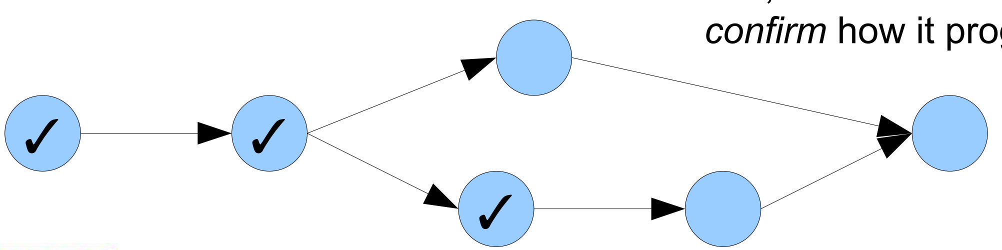


Cell towers see his phone here

...but that's only got a precision of 100m

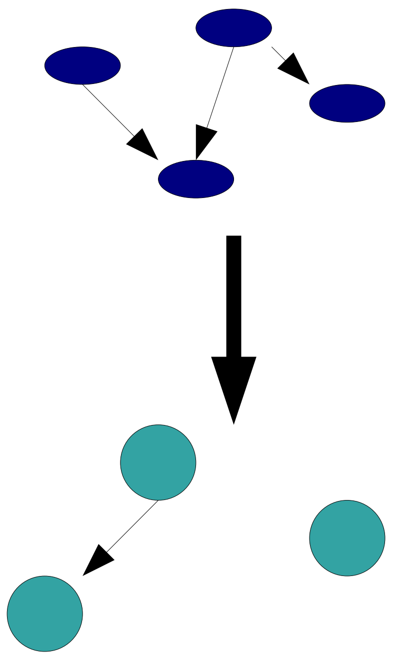
...and he might have had his phone stolen

Model the process we *expect* to see, use sensor information to *confirm* how it progresses



Ye, Dobson and McKeever. *Situation identification techniques in pervasive computing: a review*. PMC. To appear.

Approaches



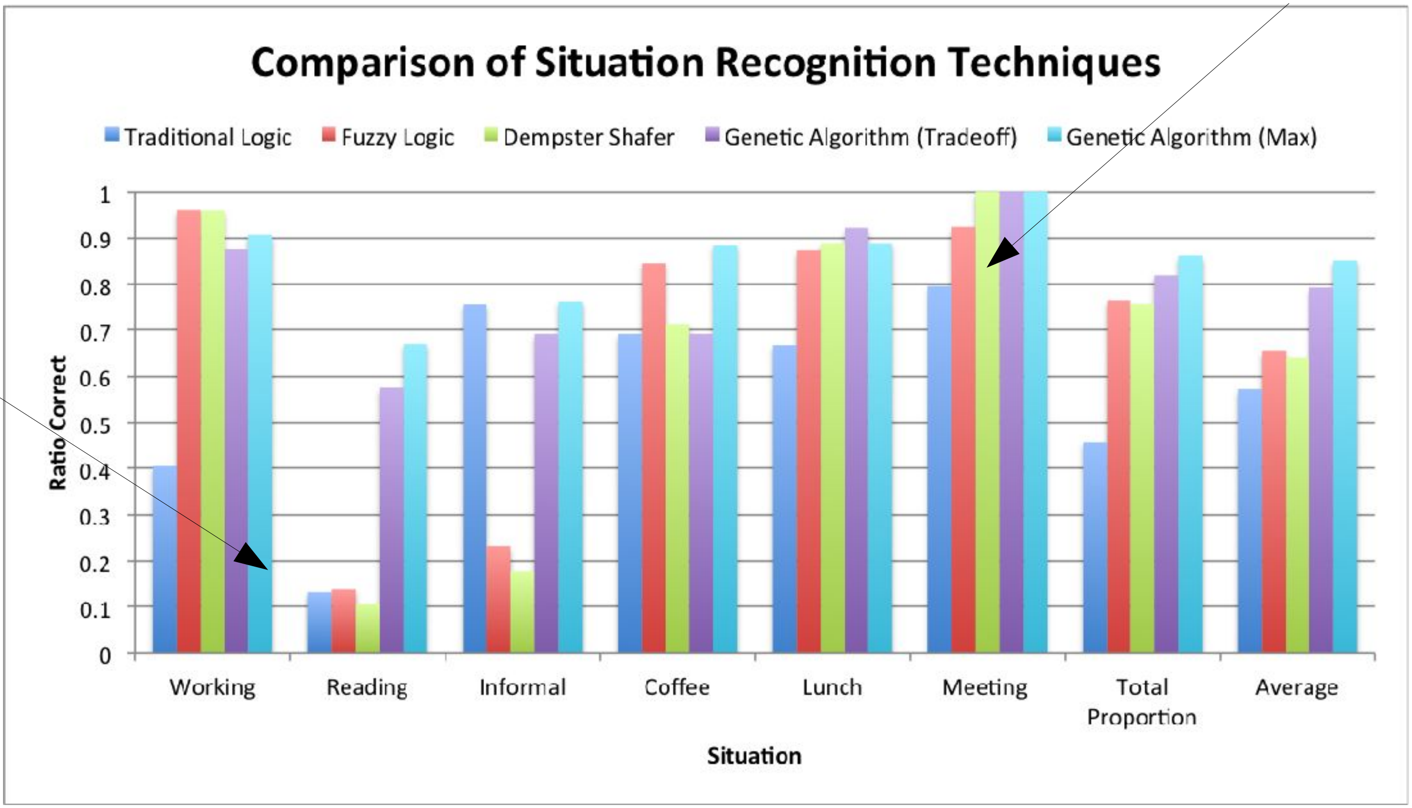
- Predicates
  - What ranges of data map to what
- Bayesian inference
  - P(S | C) – being in situation given a particular set of observations
- Dempster-Schafer evidence theory
  - Distribute mass of belief
- Case-based reasoning
  - Use similarity to past, human-classified cases

Interpretation

- No certainty with which to do control

Very well-characterised activity

No direct sensing of these activities



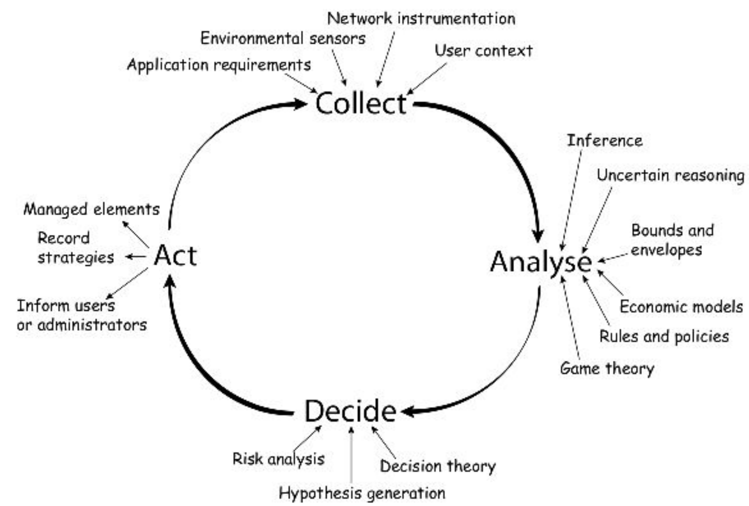
- What do you do when you can't trust any of the inputs and can't ask a user?

Part III

What to do when you can't trust any of the inputs, and you can't ask a user

Characterising the problem

- Autonomic control in the presence of rich sensor data
  - Multi-modal
  - Uncertain reasoning
  - Stability and agility



- Maintain a rich model of the system as it is deployed and evolved
  - Use to manipulate science *and* engineering aspects of a sensor network across its lifetime

Missions

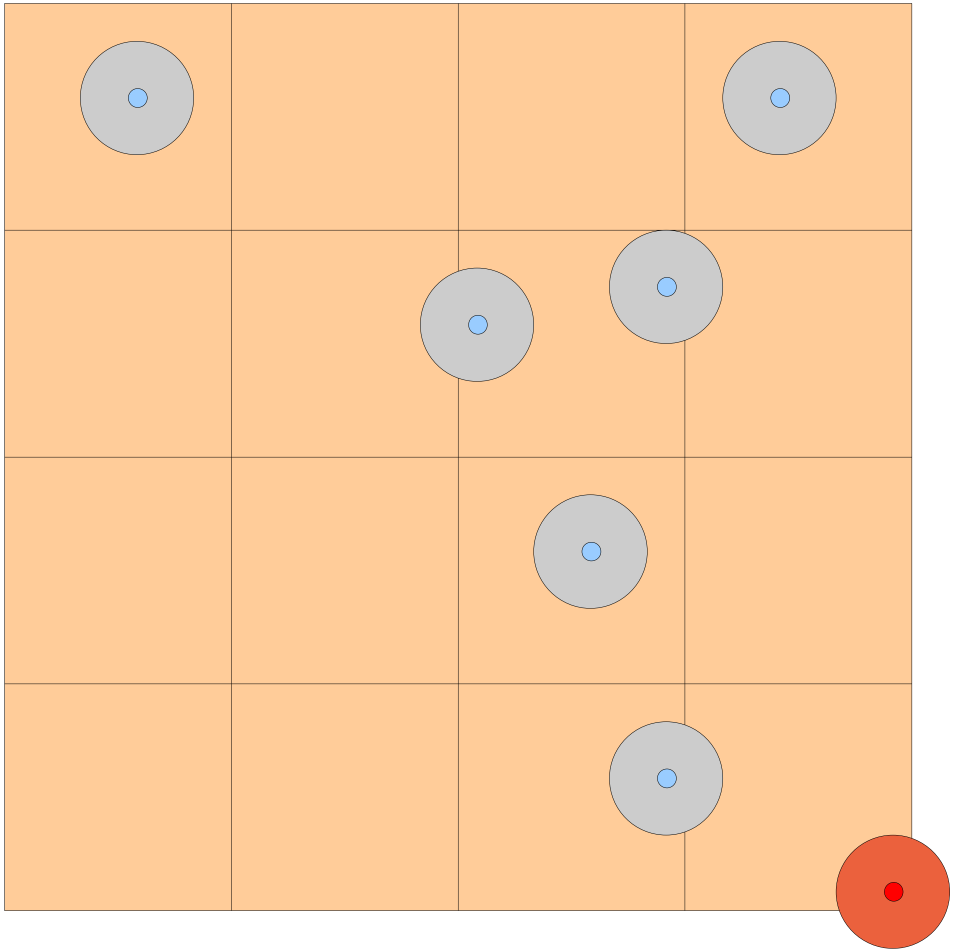
- Sensor networks are deployed for a reason
  - The *mission* the network is to accomplish
- Perspectives
  - Scientific: collect at particular resolution; adapt to changing observations; maintain/log statistical properties
  - Engineering: adapt to failures; maintain communications; manage power
- These perspectives are entwined

Understood by the mission scientists

Understood by the network engineers and developers

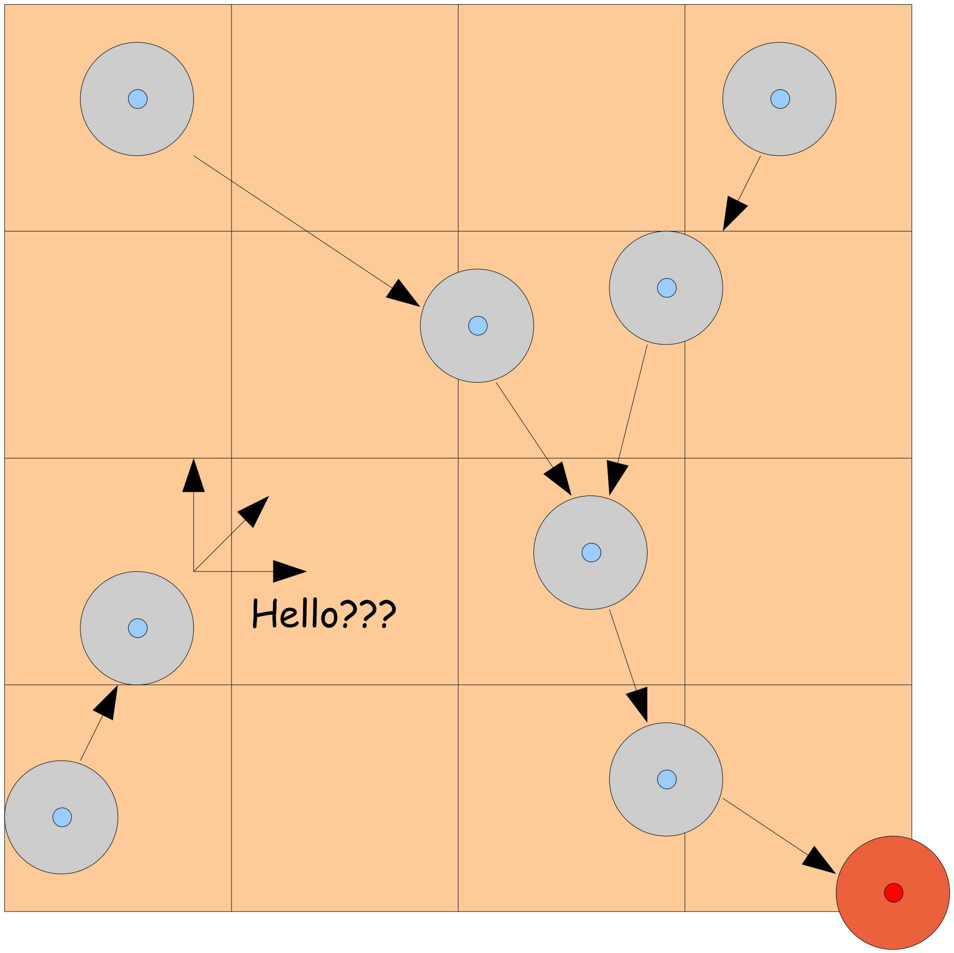
Example: placement – 1

- Looking for data on a grid; getting data from irregular sample points
  - Can often deal with this *as long as we know*



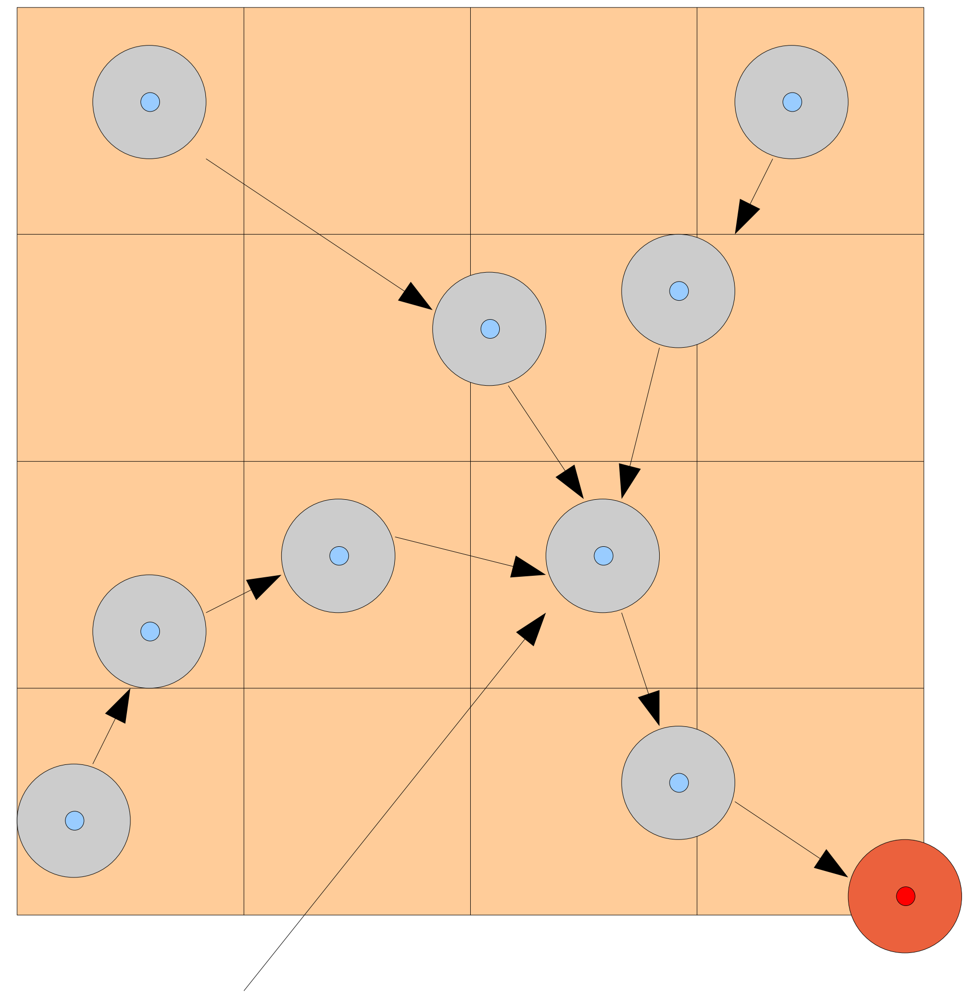
Example: placement – 2

- Looking for data on a grid; getting data from irregular sample points
  - Can often deal with this *as long as we know*
  - Changes may not all make engineering sense



Example: routing – 1

- Re-arranging for routing may not then make scientific *or* engineering sense

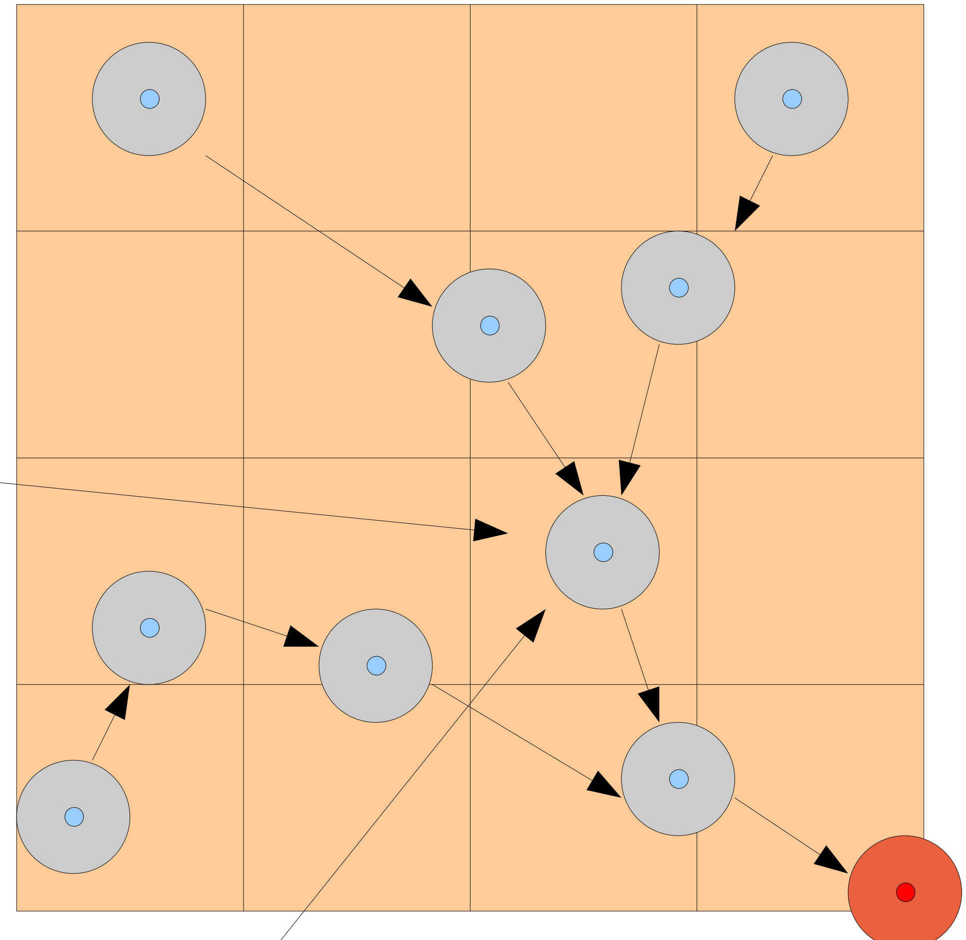


Example: routing – 2

- Re-arranging for routing may not then make scientific *or* engineering sense

Now can't perform aggregation at this point, so need to change the functional logic

- Functions and communications are multiplexed onto the same devices



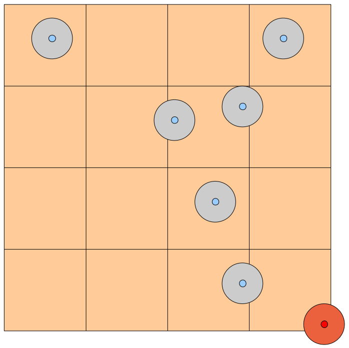
Overloading this node and/or making failure more likely/significant

Dearle and Dobson Mission-oriented middleware for sensor-driven scientific systems. J. Int. Serv. Apps. 2011.

Capturing mission

- What we need is to *capture the mission* in a way that we can use for *both* scientific and engineering management
  - Changes have goals, costs and consequences
  - Mission science has constraints that *must be* and *preferably should be* maintained
  - Preferences for different set-ups

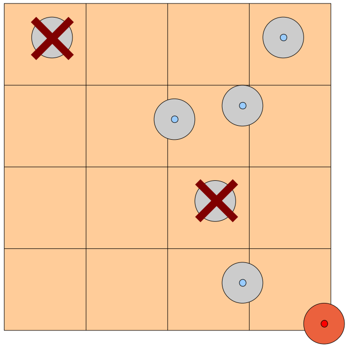
Example mission



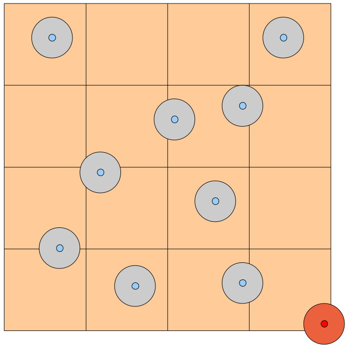
- Goals
  - Sense the levels of a pollutant in a field
- Constraints
  - Estimating pollutant levels on a grid from a sparse set of points
  - Each data point comes with provenance as to its location, time, precision etc
  - Reliability of estimate of data degrades with space and time
- Maintain view of metadata properties

Changes

- Losing a sensor
  - Changes error bars of estimates
  - May destroy connectivity with some (or all) of the network

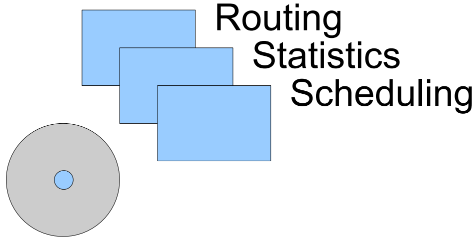


- Adding a sensor
  - Improves (hopefully) estimates
  - Changes connectivity
  - May also change functional capabilities

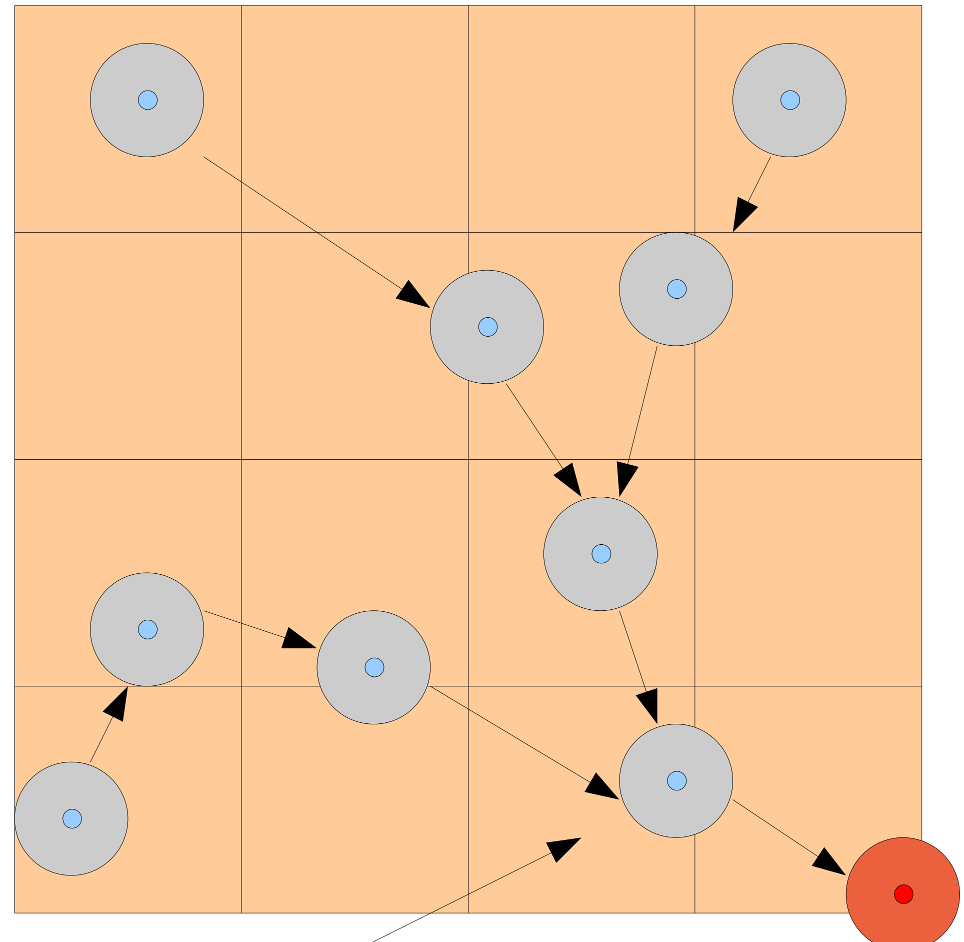


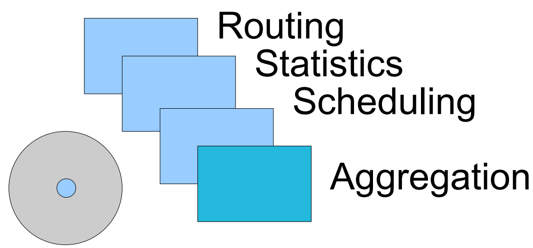
Impact on components

- Each mote hosts some components providing the various functions



- Change in routing induces change in in-flight processing





Porter *et alia*. Type-safe updating for modular WSN software. DCOSS. 2011.

Architecture – 1

- To design a mission
  - A set of *components* and their *placement*
  - A *description* of the system's behaviour along *axes of interest*
  - A set of *adaptations* taken in response to different *situations* identified from the sensor input
  - *Implications* of each adaptation in terms of the axes
  - A set of *invariants* to be preserved across adaptations

Architecture – 2

- Life cycle
  - Maintain the description on-line
  - Adaptations affect components, their parameters *and* maintain the description
- What's left unsaid
  - Open axes: don't care what they are, only that we can observe them and their changes
  - Invariants: might be complex
  - Languages: keep components in whatever language is appropriate

We're currently looking at using extensible languages and virtual machines for components and missions

Entwining

- Many (most) of the adaptations will have an impact on motes *and* their results
  - Mote failure changes routing
  - ...which might cause another aggregator to be deployed elsewhere at a strategic point
  - ...which has an impact on power consumption
  - ...and also on the precision and certainty of data collected and calculated

Stability *vs* agility

- Conflicting forces
  - Stability: stay within a predictable envelope
  - Agility: adapt quickly to changes
- Can we balance these two in a principled manner?
- Can we analyse a set of adaptations to check whether they're stable wrt the axes?
- Can we model the effects of all adaptations we might want to make?
- What are the costs incurred?

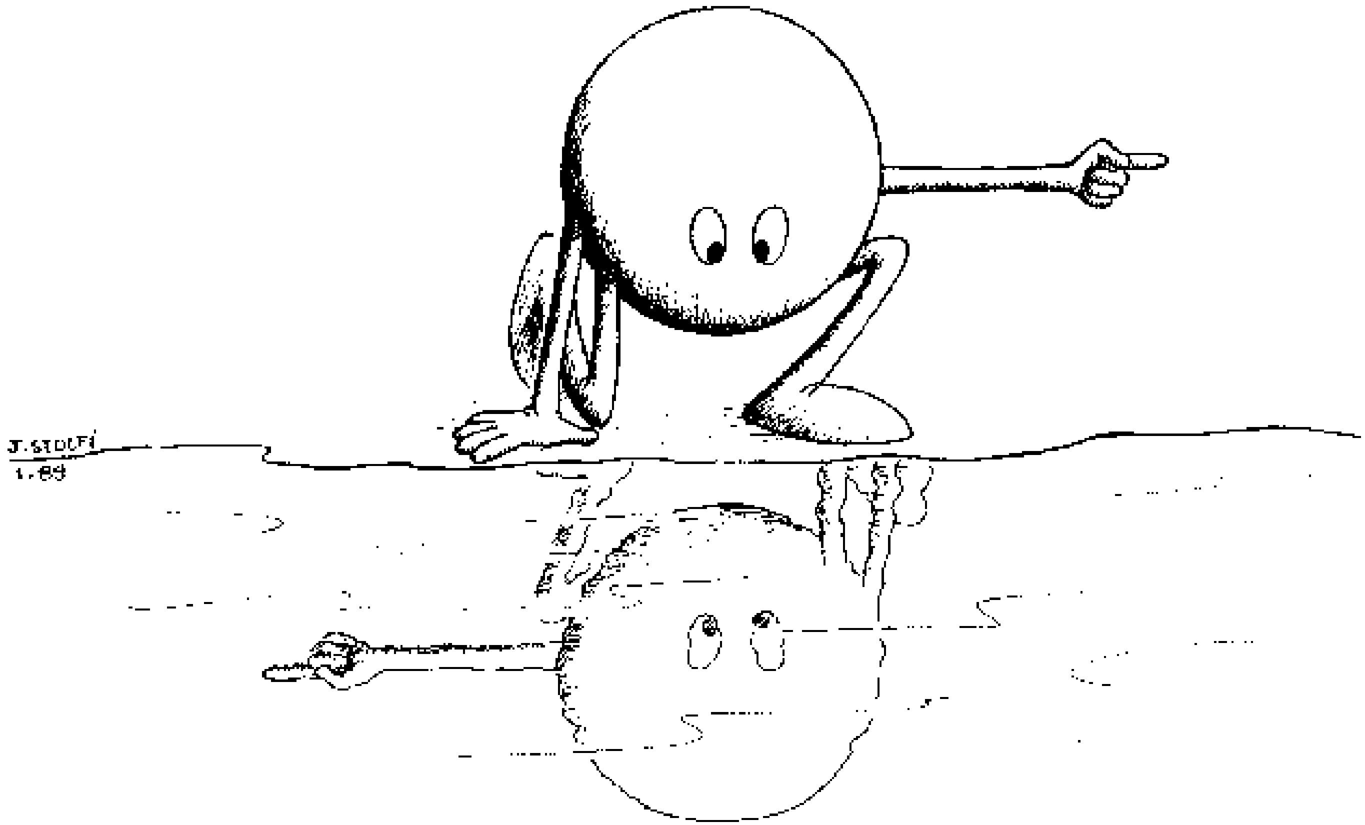
Present state

- We're confident we can *build* a mission language; less confident we can *analyse* one
  - Language design
  - Match against what's checkable
  - As static a set of guarantees as possible
- Missions seem to make sense architecturally
  - Round-trip engineering, keeping an on-line description
  - Keeping checks lightweight will be challenging
  - ...along with everything else...

Three things to take away

- Sensor networks offer unique opportunities
  - Changes the *way we do science* and the *science we do*
- Coupling science and engineering
  - Ensure that mission goals are kept even while allowing flexible adaptation and clever computing
- Describing the mission to the run-time system provides a basis for informed adaptation

Thank you



In theory, there is no difference between
theory and practice. But, in practice, there is.

Jan L.A. van de Snepscheut