Mission-oriented middleware for sensor-driven systems

Alan Dearle & Simon Dobson

University of St Andrews
School of Computer Science

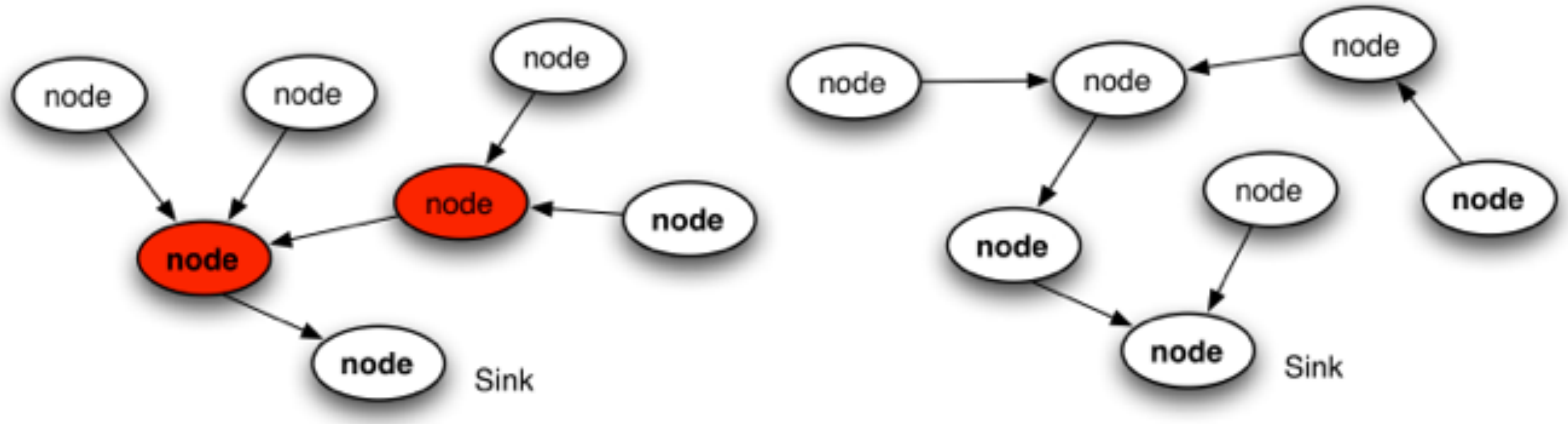
Sensor Systems are different

- Functionality lies in the network not in the nodes
- The flat network model doesn't work
- The application domain
- Where does autonomic behaviour execute

The network is the Computer

- In sensor networks the functionality lies in the network not in the nodes
    - Robustness, longevity, coverage etc. are network properties
    - **We need to program the network not the nodes**
    - Middleware has focussed on simplifying interactions between nodes
    - Network is usually considered to be flat & non-computational - only a transport
    - We need to program and understand the failure modes

The flat network model doesn’t cut it



- Failure mobility environment QoS
- Power Budgets, for send, timeout, route formation
- Staged Computational model

The unchanging application abstraction

- This abstraction is used in TinyDB and Flask

1. It is possible to describe the application objectives in a single specification
2. The topology of the network is known *a priori*
3. The application objectives are compiled away in the code on the nodes
4. Mechanisms exist for deploying the components on to nodes
5. Appropriate communications infrastructure is available to achieve the required data transmission, synchronisation and choreography of the application

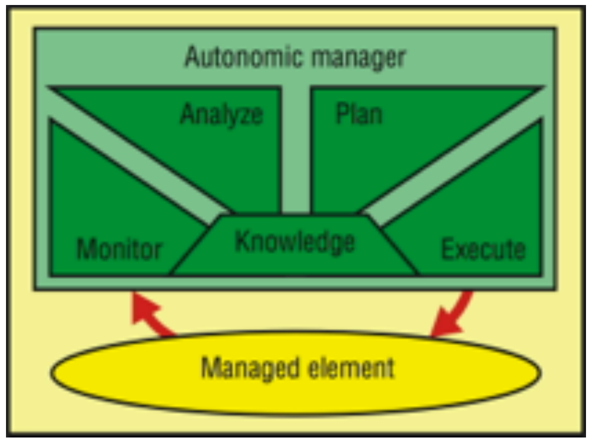
A different application domain

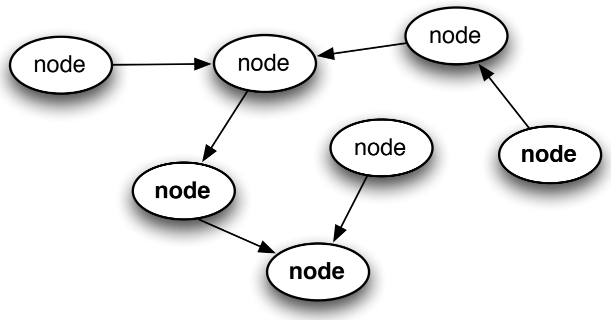
- Is the single *unchanging application abstraction* what we need?
- Pressures:
    - Multiple missions for sensor nets
    - Evolution of purpose
- This model is fundamentally flawed since the network topology, the set of operation nodes and the software placed upon them must be able to evolve to respond to environmental changes, node and routing failures

Where does the autonomics run?

- To control autonomic behaviour information needs to be collected
- How do we choreograph change [error, budgets, inability to address, understand configuration]
- How do we understand how local changes affect whole system
- Need an understanding of:
  - System level
  - Node level
  - Component level

  at all levels of the system





Enough moaning, what can we do about it?

- First two questions:

  1. What functions do we want to be provided by the operating systems, middleware and languages on WSNs?

  2. What system and language structures best provide these functions?

  Answering these questions allows us to re-imagine how we should provide system and development support for the next generation of WSNs

what can we do about it?

- Eliminate a priori abstractions
- Support introspection
- Capture and understand mission
- Use belief systems
- Build in provenance
- Take control

No a priori abstractions

- Eliminate traditional layering
    - These abstract over failure, hide locations, create brittleness
- Components must be independent of each other, self contained, replaceable
    - e.g. Lorien,Insense
- Code needs to carry certificates specifying (guaranteeing) behaviour

Introspection

- Without the ability to introspect, in-system evolution is not possible, the only evolution possible is externally imposed heteronomy based on reported information or observations about the system's behaviour

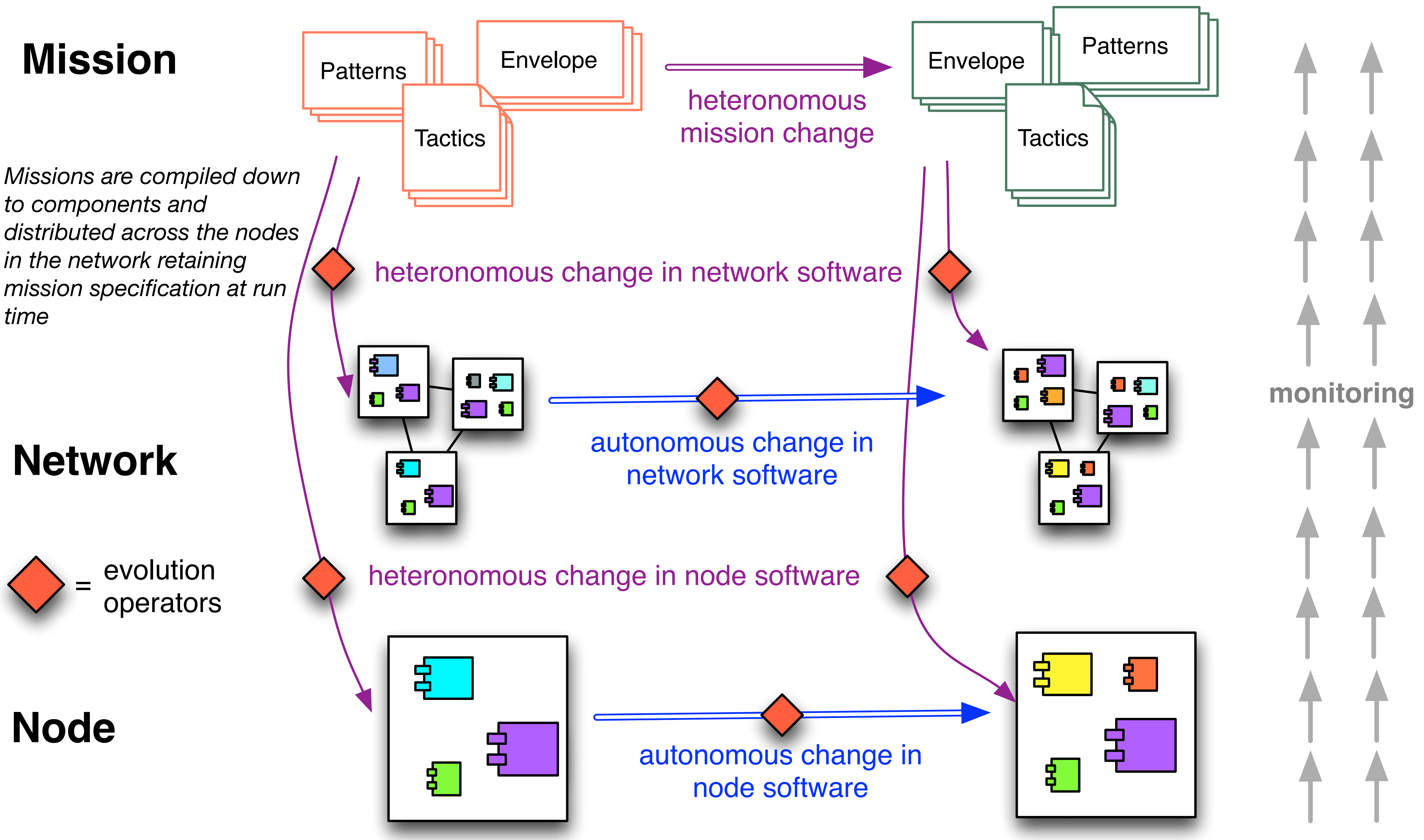
- There are a number of system attributes that require introspection:

  1. The node state, including the state of the devices, battery, the hardware and software
  2. The mission or missions on which the node is employed
  3. The local network environment, including the node's neighbours

Encoding and preserving Mission

- Adaptation and evolution are crippled if they must occur only at the behest of (and using information known reliably to) a central autonomic controller
- Breaking this restriction requires that the required **system-wide behaviours** must be **defined** using **explicit mission specifications** made available to the network at run-time
- A mission specification **defines the envelope** within which deployed software may autonomously evolve, and constrains heteronomous evolution to respect timing, distribution and other constraints
- The mission specification must be **available throughout the system's lifetime to all** the **processes** that operate within it; to processes making decisions about the locations of computational elements; and to those making routing decisions and configuring MAC-level hardware components

Mission


Mission
Patterns
Envelope
Tactics
heteronomous mission change
Envelope
Patterns
Tactics
Missions are compiled down to components and distributed across the nodes in the network retaining mission specification at run time
heteronomous change in network software
Network
autonomous change in network software
monitoring
= evolution operators
heteronomous change in node software
Node
autonomous change in node software

Belief

- Wireless sensor nets are inexact environments
- Nodes must base many decisions on their beliefs about these factors and must function in or recover in situations where this belief turns out to be incorrect
- Often exact knowledge is in-conflict with mission goals - network longevity for example
- However, the encoding of mission may be used to inform requirements and understand the boundaries of what is known and what may be inferred

Provenance

- When a sensor network reports that the average temperature in some area is 21°C the provenance of this data needs to be understood:
    - the accuracy of the sensing devices,
    - the number and spread of the devices that have contributed to the result,
    - the standard deviation, mean, modal values and other derived signals,
    - how recently the data was collected,
    - the computation that has been applied to the raw readings before they are reported
- To be able to report such information requires provenance considerations to be built in from the ground up in the same manner as belief systems
- The encoding of missions and models helps with these tasks, since they help to inform decisions on data handling and to record the fact that these decisions have been made

In conclusion

- The future of WSNs will involve complex design and management choices being made in the face of uncertain information, overlapping and evolving missions
- New developments are needed in autonomic systems architecture
- We need to move descriptions of the system's architecture and mission down into the systems level to inform and control adaptations over time
- Thus ensuring that the systems evolve within an acceptable mission envelope
- *To what extent would all complex distributed systems benefit from some of the techniques we have advocated here?*