Architecture-Independent Programming and Synthesis of Networked Sensing Applications

Amol Bakshi, Viktor K. Prasanna
Department of Electrical Engineering
University of Southern California, Los Angeles CA
{amol, prasanna}@usc.edu

Daniel Larner
Palo Alto Research Center
3333 Coyote Hill Rd, Palo Alto CA
larner@parc.com

Macroprogramming with the Abstract Task Graph (ATaG)

Macro @ application level

- **Defining and manipulating events** at high levels of abstraction
- Examples: "latest position of target", "boundaries of all regions where temperature is greater than threshold", "coordinates of all malfunctioning sensors"

Macro @ architecture level

- **Specifying distributed computation** in aggregate terms
- Examples: "apply f(x) to all variables x within 10m", "construct spanning tree rooted at node x"

Tradeoffs

- Expressiveness: How many lines of code are required to express a 'common behavior'?
- Efficiency: How much control does the programmer have?
- Reusability: How much reuse is available across applications and architectures?
- Automation: How much of the process can be automated?

Why data-driven computing?

Programming abstractions

- ✓ Tasks can use data items **at the desired level of abstraction** without worrying about who produces it and how
- ✓ Tasks are unaware of each other, leading to highly **extensible, reusable** programs
- ✓ Communication and coordination is performed in the underlying system and hidden from the programmer

Software and hardware

- ✓ Event-driven processing leads to **efficient resource utilization**
- ✓ Parameterized runtime system template is amenable for **automatic software synthesis**
- ✓ Runtime system can be ported to an entirely different platform while providing the same application-level interface

Key Concepts

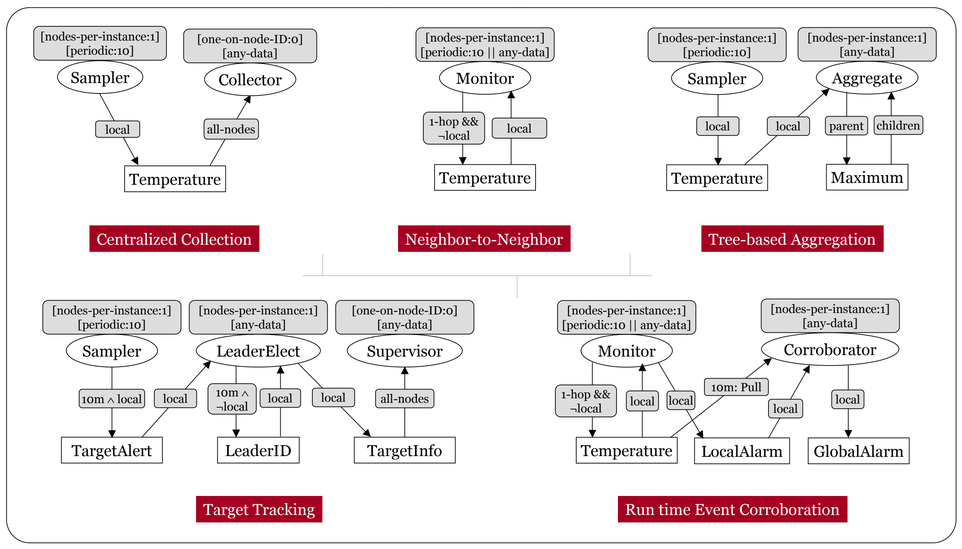
Data-driven control flow over a distributed data store

- An event can carry **information** about a phenomena (target location, temperature reading) or merely indicate its **occurrence** to the next phase of processing
- Task **firing rules** – periodic, any-data, and all-data – allow the specification of a range of sophisticated execution patterns
- Scheduling and communication is managed by a runtime
  - Programmer does not write networking code
  - System-level optimizations can be performed without requiring rewrite of application-level code
- **Pros**: Modularity, reusability, extensibility, low program complexity
- **Cons**: Asynchronous interaction, unpredictable delays, limited control over execution ordering

Mixed imperative-declarative program specification

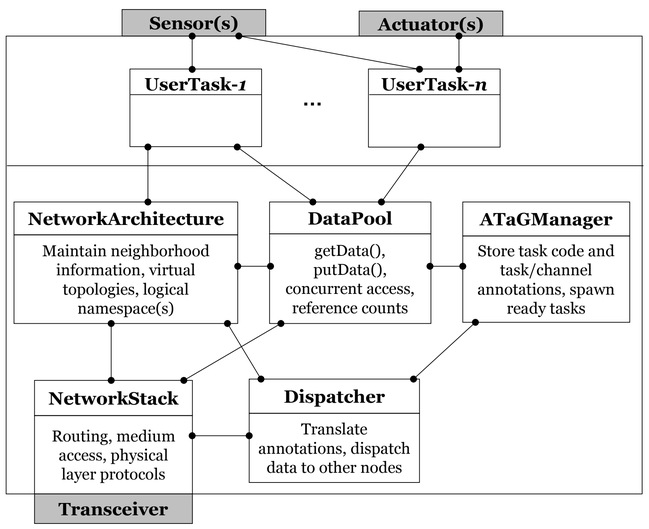
- **Separation of concerns**: "what" of processing vs. "when and where"
- Imperative portion is a traditional (C/Java) sequential program interacting with the data pool
- Declarative portion is interpreted – at compile time and at run time – in the context of a particular network architecture
- Imperative and declarative parts can be modified independently

Programming Idioms



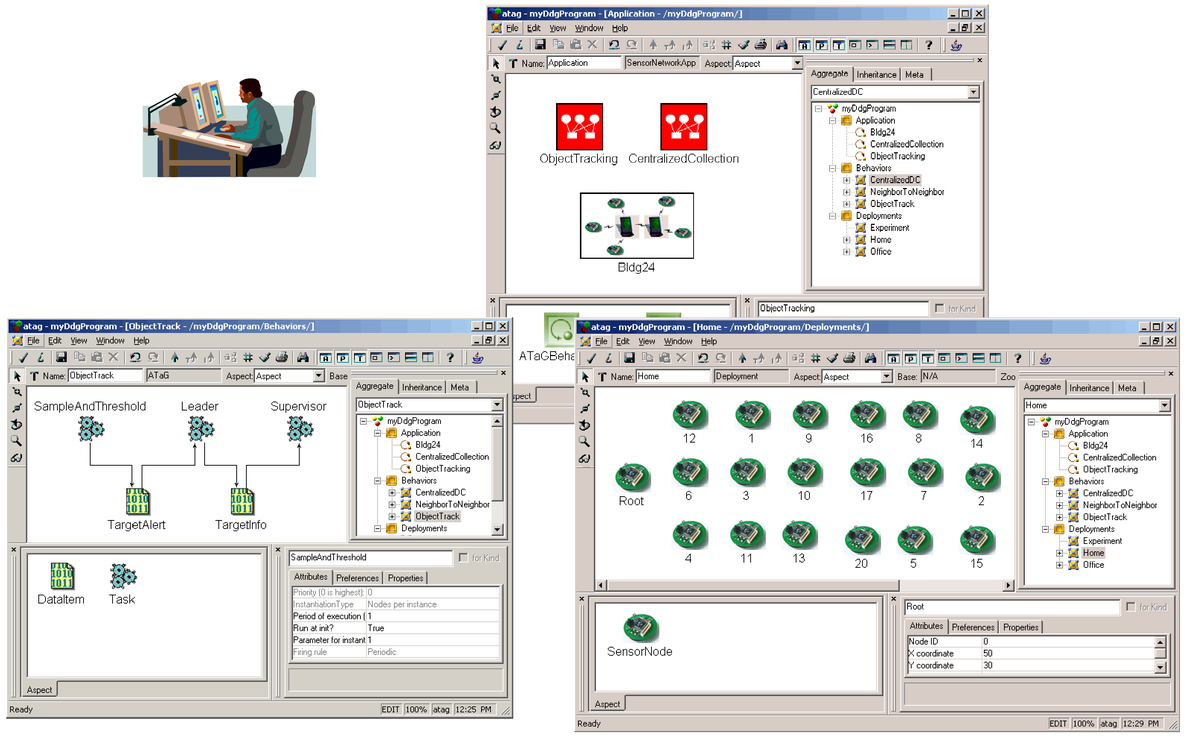
- "The Abstract Task Graph: A Methodology for Architecture-Independent Programming of Networked Sensor Systems", A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner. *Workshop on End-to-end Sense-and-Respond Systems (EESR), in conjunction with MobiSys, Jun. 5, 2005.*
- "System-level Support for Macroprogramming of Networked Sensing Applications", A. Bakshi, A. Pathak, and V. K. Prasanna, *International Conference on Pervasive Systems and Computing (PSC), Jun. 27-30, 2005.*

DART: Data Driven ATaG Runtime

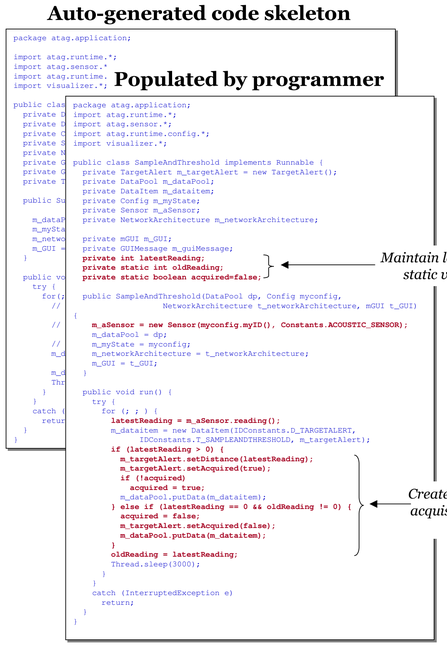


- Encapsulates 'system-level' functionality and the related optimizations
- Application-independent control and coordination mechanism is separated from application-specific configuration
- Modular structure with well-defined inter-module interfaces: (a) simplifies software synthesis, and (b) enables replacement or enhancement of intra-module protocols with minimal system redesign
- Requires support for multi-threaded execution and fixed-priority preemptive scheduling – available in most RTOSes (e.g., µC/OS-II, RT Java)

Visual ATaG Programming and Software Synthesis



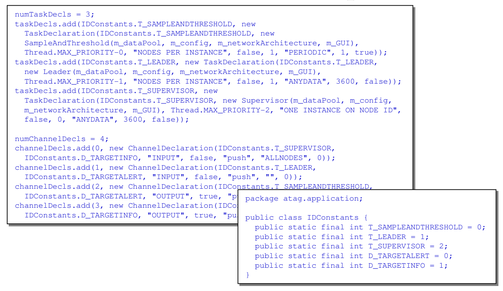
Auto-generated code skeleton



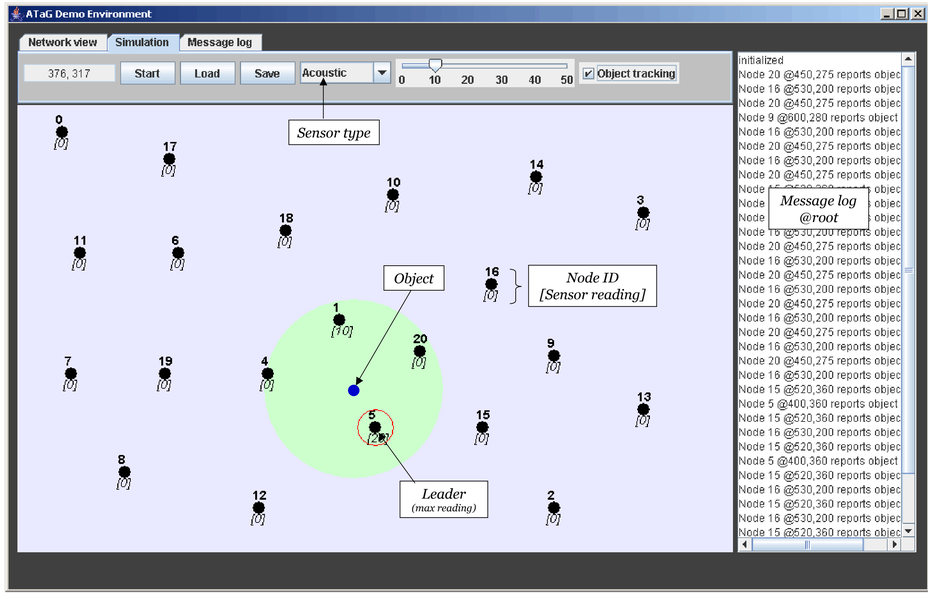
Config file for each node

```
-myID 0 -ndata 2 -ntasks 3 -hopscope 0 -distancescope 300 -assignedtasks 0 1 2 -end
-myID 1 -ndata 2 -ntasks 3 -hopscope 0 -distancescope 300 -assignedtasks 0 1 -senddata 1 0 -end
⋮
-myID 20 -ndata 2 -ntasks 3 -hopscope 0 -distancescope 300 -assignedtasks 0 1 -senddata 1 0 -end
```

Customization of the DART template



Functional simulation and visualization



Fourth International Symposium on Information Processing in Sensor Networks (IPSN), Apr. 25-27, 2005.

http://indus.usc.edu/atag/