# Observations and directions for high-confidence sensor networks

Daniel Andresen John Hatcliff Gurdip Singh Steve Warren

Advances in communication and computing technologies are enabling deeply embedded, networked systems of sensors that can collect real-time data from a number of different, remote sources. Such embedded sensor networks have a wide range of military, medical, agricultural, environmental, commercial, and homeland security applications. As researchers at Kansas State University (KSU), we have great interest in aggressively pursuing sensor network technology in agro-medical domains because advances in these domains are important to the medical field, our local/regional economies and to KSU's mission as a land-grant university.

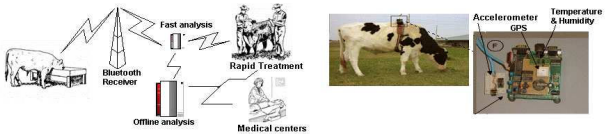


**Figure 1. Envisioned cattle health monitoring system (left), Existing cattle monitoring component (right)**

In this discussion, we draw our examples from major agro-medical applications in veterinary telemedicine which we have been working with for several years. sensing. Figure 1[1] shows a simple application that employs a network of environmental and motion sensors to remotely assess animal state-of-health. The broad goal of such systems is to provide remote, easily accessible, objective measures that highlight specific animals or groups of animals for the purpose of concentrating the full efforts of the labor and veterinary healthcare force. The benefits of such targeted information include (a) more efficient allocation of resources, and (b) increased efficiency for first responders and the industry as a whole. Furthermore, most of these technologies and infrastructure are highly transferrable to human medical care.

Given the fast emergence of novel sensor applications like these, techniques are needed that allow developers without extensive computer science backgrounds to rapidly design and prototype sensing applications. Any such development solutions must carefully take into account the following characteristics that differentiate sensor applications from traditional distributed applications:

1. In contrast to most information systems, development and knowledge required is not simply limited to the application layer. Rather, development requires knowledge about multiple aspects/dimensions of the system ranging from hardware, networking, software architecture, application logic, and domain knowledge (what information should we gather to make appropriate inferences for a particular domain).

2. Sensor network applications typically involve a *large number of heterogeneous* components with several crosscutting logical sub-systems involved in the detection/control of different physical phenomena or objects.

3. Sensor systems are *highly dynamic* and different subsystems may be active at any given time depending on what needs to be sensed and controlled. For example, in Figure 1, the GPS sensor subsystem may remain inactive until alerted by the motion sensors that movement is taking place (thus conserving battery power).

4. Sensor platforms are *resource constrained* with limited power, memory, transmission and computing capabilities.

5. Designing a sensor application typically is a *multidisciplinary* exercise involving a variety of domain experts each having a different view of the system. For example, a medical/controls expert may want to design the system in Figure 1 taking a centralized view which may conflict with the decentralized view a networking expert has to contend with during deployment.

1 Warren, S., D. Andresen, L. Nagl, S. Schoenig, B. Krishnamurthi, H. Erickson, T. Hildreth, D. Poole, and M. Spire. "Wearable and Wireless: Distributed, Sensor-Based Telemonitoring Systems for State of Health Determination in Cattle," 9th Annual Talbot Informatics Symposium, July 25, 2004.

6. Sensor applications typically have a combination of *soft and hard real-time* constraints. However, they often are deeply embedded where the underlying network links may be unpredictable with respect to latency and reliability.

The characteristics of sensor applications mentioned above make their design and implementation a difficult task, and the design teams are often faced with several challenges, as discussed in the following:

1. The *large scale, heterogeneous, multidisciplinary* nature of sensing applications makes it hard for designers to have a complete, coherent a priori view of the implementation. For example, the initial design of the application in Figure 1 may proceed with a electronics engineer designing the biomedical sensors and a veterinarian designing the system in terms of the level of temperature sensed, analysis to be done and the control actions to be taken (such as adjusting sensor range/frequency). Although one would like to present such an abstract model of programming, the design team cannot be isolated from system concerns such as how many sensors are needed, can a mix of sensors with high and low reliability be used, how data from multiple sensors is fused, and how long will it take for the data to be analyzed. **Implication:** frameworks to *design large-scale sensor systems with capabilities for modeling at different levels of abstraction* are needed. These frameworks must allow systems to be developed in a step-wise manner with different aspects and details of the system being introduced in the refinements.
2. Although the communication patterns and protocols used may differ from application to application, *commonalities across applications* need to be exploited to enable code-reuse and rapid development of sensor networks. For example, if the system in Figure 1 were to be re-deployed from a pen to open terrain where animals are randomly dispersed, then we would like to reuse as much of the design as possible. Likewise, we would like to exploit the commonalities of this application with others such as those for environmental monitoring that collect environmental data in an open terrain. **Implication:** a key challenge is to provide *development environments that allow designers to organize design knowledge and software* in a manner which enables systematic reuse of system components *and* architecture, as well of a variety of other development artifacts, in *families of similar sensor systems*.
3. Sensing and control activities performed in sensor applications are often distributed and dynamic in nature. **Implication:** *abstractions to support modeling and implementation of distributed interactions* are needed. These abstractions must be able to accommodate and reconcile the different views of an interaction adopted by different design team members, and facilitate development by non-experts.
4. Given their resource constrained nature, sensor platforms often provide minimal (or no) OS and protocol support and leave aspects such as reliability and synchronization to the application designers. **Implication:** support is needed to allow designers to *specify and manage QoS requirements*, and *implement protocols* required to satisfy these requirements.
5. Designers may be faced with several choices while mapping system requirements to available services or when designing new services. For example, for the application in Figure 1, the designer may have to determine the number of sensors and base stations required, the parameters of the individual sensors ($e.g.$, sensing frequency and range) and the logical topology (linear vs ring) to arrange the sensors. **Implication:** tools are needed to *aid the designer* in making these choices. In particular, development environments must allow a variety of *mathematical procedures and simulation tools* to be plugged in for analysis at different points to aid in the design process.
6. As mentioned above, rapid, end-to-end development of sensor network applications will require the support of a number of tools for modeling, analysis, code synthesis, and deployment. There has been a considerable amount of work in developing various tools to address these issues. **Implication:** programming platforms are required that allow existing tools to be leveraged in the unified manner for end-to-end system development.

There has been a significant amount of research done in the past few years to develop protocols addressing issues specific to sensor networks such as limited power, sensor failures, and unreliable communication. Leveraging these underlying protocols, a number of frameworks to program sensor networks have been proposed to address the challenges listed above. While node-centric frameworks simplify programming, they may not scale to large systems as the designer must explicitly code distributed interactions such as discovering neighbors, sharing data with neighbors and coordinating sensing activities. also lack abstractions for modeling dynamic, group interactions. Recent work on middleware-based and macro-programming approaches are a step in the direction of programming multi-node interactions. While these abstractions have been shown to be reusable in a variety of applications, they represent programming at a specific level of abstraction. Frameworks are needed not only to allow specification of systems at differ-

ent levels of abstractions, but also to provide their implementations as different design choices, expose the properties of each abstraction (e.g., message cost, latency, level of consistency of shared data) at the design level, and provide guidelines and analysis engines to determine which one is more suitable for a specific application.

We suggest complementing and advancing this previous work by embracing an approach that combines two key enablers of recent successes in development of families of distributed systems. Emerging *model-driven design and implementation technologies* are using information captured by models, constraints, and analyses to provide automated design advice and to automatically configure and optimize components and underlying services. Large scale development efforts are also increasingly being based on *software product lines* - a development process for families of similar products that aims to drive down development time and costs by systematically reusing infrastructure and common application components across many applications within a particular product family. For example, a framework for *model-driven, product line-based* design and implementation of sensor network applications could be built by enhancing Cadena, an integrated environment we have developed for designing real-time embedded systems.

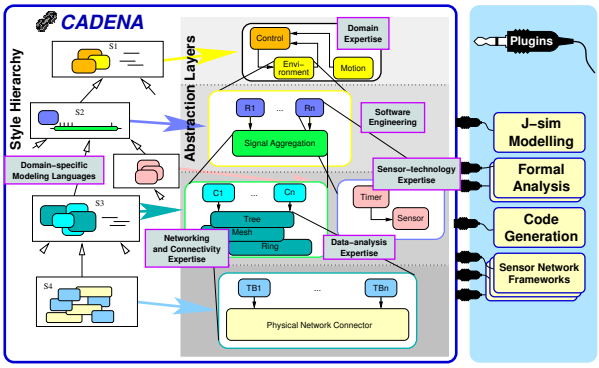


**Figure 2. CADENA managing the multi-layered development process**

To enable systematic reuse across families of products within multiple sensor network domains, we suggest an infrastructure for building product lines (i.e., a *product line* for *product lines*) to support product line-based development for families of sensor networks in multiple sensing domains. Within each product line, our infrastructure would enable a novel *domain-specific modeling language hierarchy* that allows programming at different levels of abstraction within a sensor system. Components and other architectural elements appropriate for each specification level will be identified. This approach exposes different programming views at each level, appropriate for different domain experts in the design team, and allows various aspects and system details to be introduced in a step-wise manner. To provide building blocks common to many sensing applications, we require the investigation of mechanisms to model and implement dynamic, distributed interactions among sensor network components. For example, Cadena uses the notion of *software connectors* to model inter-component interactions and the underlying middleware services. The *connector* mechanism could be enhanced to capture different forms of multi-component interactions. Figure 2 shows different types of connectors (Signal Aggregation, Tree, Mesh) being used at different specification levels (for example, level S2 describes interactions assuming a centralized view whereas level S3 takes a decentralized view but assumes a logical topology). A library of pre-defined connectors for different product lines, mechanisms to define new connectors and to arrange them in a refinement hierarchy should be developed.

To configure underlying middleware and communication services, an infrastructure should be developed to associate *domain-specific attributes and QoS properties* with components and connectors, and to specify *constraints on attributes and properties* spanning multiple components and connectors at the same abstraction level and across levels. To aid the designer in making design choices as we move down the hierarchical model, the tool should include a variety of *mathematical analysis and simulation engines* as plug-ins such as *J-Sim* and *TOSSIM*, both which are component based and include capabilities to model various types of sensor nodes and communication channels.

## Biographies

Our authors have wide-ranging experience in building and applying model-driven development tools, sensor applications development, distributed algorithm, middleware and wireless communication. Hatcliff, Singh and colleagues have worked on building the Cadena integrated development environment for model-driven design of component-based systems. Andresen has led several multidisciplinary projects in the areas of Veterinary Telemedicine. Warren has extensive experience developing sensors and protocols for medical applications.

*Daniel Andresen*, Computing & Information Sciences, Kansas State Univ., 785-532-7914, dan@k-state.edu.
*John Hatcliff*, Computing & Information Sciences, Kansas State Univ., 785-532-7950, Hatcliff@k-state.edu.
*Gurdip Singh*, Computing & Information Sciences, Kansas State Univ., 785-532-7945, gurdip@k-state.edu.
*Steve Warren*, Electrical & Computer Engineering, Kansas State Univ., 785-532-4344, swarren@k-state.edu.