# An Event-driven Approach to Address Reflectivity in Sensornets

Amirhosein Taherkordi and Frank Eliassen
University of Oslo, Department of Informatics
N-0314 Oslo, Norway
{amirhost,frank}@ifi.uio.no

## ABSTRACT

Next-generation sensor applications require system software and middleware that can be dynamically adapted to meet resource constraints on sensor nodes, as well as application-specific requirements. The *reflective* middleware model has emerged as a de facto standard to attain inspection and adaptation in dynamic software systems. However, prior reflective frameworks in the field of sensor networks come with some drawbacks, such as imposing high resource demands on sensor nodes, and lacking a precise design model ended up with a tailored programming model. In this paper, we consider reflection in sensornets from programming perspective and propose an *event-driven, component-based* framework addressing both *base-level* and *meta-level* inspection concerns within reflective sensor software. To efficiently utilize the event-driven nature of sensornets, we propose a new *meta-space model* for adaptive sensor systems, named *Notification*. The proposed framework also makes the implementation of the base-level simpler and more efficient.

## Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*

## Keywords

Sensornets, reflective middleware, event-driven programming

## 1. INTRODUCTION

Wireless Sensor Networks (WSNs) have been recognized as an emerging technology for monitoring and controlling a variety of phenomena, such as environmental surveillance, infrastructures, and medical environments. In contrast to early WSN applications that were developed over rigid system software with fixed functionality, recent applications are being hosted by operating systems and middleware services that can be dynamically reconfigured. This, in fact, indicates the capability of sensor software to deal with changing requirements in the system which are either sensor-specific or application-specific. The former refers to adapting sensor node behavior based on the amount of resources (*e.g.*, energy) available in the node. The latter, instead, emphasizes on the dynamic requirements of applications, *e.g.*, the network configuration may need to be changed along the application lifespan to satisfy QoS requirements.

The *reflective* middleware model is considered as an efficient way to deal with highly dynamic environments (*e.g.*, sensornets) and support the development of flexible and adaptive systems. By being reflective, the structure and behavior of systems can be inspected, adapted and extended at runtime. The ability to reflect software behavior can be very beneficial in dynamic WSN applications, needing to continuously observe system performance and react accordingly. Although this issue has already been addressed in a few approaches [3, 1], they either induce high resource demands on sensor nodes, or propose a very abstract architectural model, without a concrete development solution.

In this paper, we propose a programming approach that facilitates the development of reflective software systems over WSNs. This component-based programming model provides the programmer with a set of design choices that simplify the implementation of inspection tasks in reflective sensor systems, both at the base-level and at the meta-level. The key design principle, in our approach, is that we pay a special consideration to event-based paradigms in order to provide a practical and lightweight mechanism to implement reflective aspects in sensor networks. To achieve this, we devise a new *meta-space model* for adaptive WSNs, called *Notification meta model*, which represents a transition from passive inspection models handled by applications to a more precise and active inspection technique maintained by the reflective middleware. All these efforts are made using a component-based programming model we have recently proposed specifically for WSNs, called REMORA [5].

## 2. PROGRAMMING FRAMEWORK

In this section, we first give a very brief introduction to REMORA, then propose the extensions made to this component model to support reflective programming. REMORA is an extension of the Service Component Architecture model [4], specialized for developing WSN software systems. The specifications of REMORA components include: services, references, interfaces, producers, consumers, and properties. A service can expose a REMORA interface, describing the functions provided by the component. A reference can request a REMORA interface, which describes the operations required


*ARM 2010,* November 30, 2010, Bangalore, India.

by the component. Similarly, a producer identifies an event type generated by the component, while consumer specifies a component's interest on receiving a particular type of event.

## 2.1 Base-level

The base-level consists of components implementing the usual middleware services. Prior to addressing adaptivity, we need a mechanism to enable inspection on components. Ideally, the inspection model should be considered as a separate concern that can be incorporated to components at design stage and embedded to the final application code by the adaptation framework. This enables reusability of inspection modules in different types of components. To this end, we enhance REMORA with a new concept, named *Inspector Annotation (*INSAN*)*, to realize the above goal. INSAN enables components to inspect their own behavior at two distinct levels: service and component. The former is a detailed model of inspection limited to the scope of a particular component service, while the latter reflects the overall behavioral information of a component. An INSAN type may identify a number of parameters that can be manipulated by the programmer to further configure the inspection task.

Figure 1 shows the description of the DataLogger component extended with an INSAN dedicated to the `logger` service. This inspector observes the current state of sensor external memory in terms of overhead induced by a service in a given period of execution time. Likewise, this INSAN can be defined to inspect the whole component. This allows the application being instantly updated with the perilous situations caused by underlying services and therefore making appropriate adaptation decisions, *e.g.*, replacing the DataLogger with another version that ignores logging data with low significance. For each INSAN type, an equivalent event type is specified. Whenever any unusual behavior is detected by an INSAN, an instance of the corresponding event type is instantiated, then the REMORA framework delivers the event to the adaptation framework.

```
<?xml version="1.0" encoding="UTF-8"?>
<componentType name="sys.util.DataLogger">
  <service name="logger">
    <interface.remora name="sys.api.ILogger"/>
    <inspection resource="EXTMemory">
       <!-- set inspection parameter values here -->
    </inspection>
  </service>
  ...
</componentType>
```

**Figure 1: An excerpt of DataLogger component annotated for memory inspection.**

## 2.2 Meta-level

The meta-level encompasses reflective facilities to expose implementations to the programmer, enabling inspection and adaptation. There are a number of reference meta-levels indentified in the literature, such as *Interface*, *Architecture* and *Interception*. Beyond the existing passive meta models, we propose a new active meta model, called *Notification*, as sensor nodes need to instantly react to changing environmental conditions and resource variations in dynamic and harsh deployments. It is designed to improve the reflection process in WSNs through an event-driven abstraction. This meta model is configurable by the application and able to trigger the application once a new reflection event is observed. Using this approach, sensor applications can be easily integrated to reflective middleware services through a uniform event distribution layer. More importantly, the inspection process becomes significantly efficient and lightweight as the event-driven nature of components avoids imposing any extra overhead to application modules for inspection concerns. The implementation of the Notification meta model is a simple REMORA component listening to the relevant events, specified by the programmer and translating them to high-level events identifiable by application components. Figure 2 depicts the architecture of a reflective WSN software with meta-space models and middleware at the base-level.

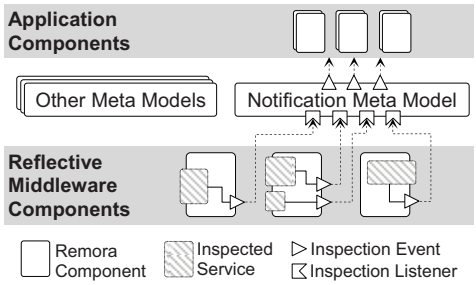


**Figure 2: The overall architecture of reflective programming model.**

# 3. A SAMPLE USE CASE

Energy is undoubtedly the primary concern of any sensornet. By being able to estimate energy consumption of sensor nodes, the application and system software can prolong the lifetime of the sensor network by making appropriate adaptation decisions. In [2], a software-based on-line mechanism has been proposed for sensor nodes to estimate energy usage in a high accuracy. This framework can surround a particular part of code and estimate the total energy usage, as well as the energy consumed by a particular hardware component, such as microcontroller and radio (in receive and transmit modes). As a concrete use case of our reflection framework, an INSAN can be developed to measure the energy consumed by a component and its services in a given period of time. This INSAN can be easily added to the description of REMORA components to reflect the energy overhead induced by a component/service. The Notification meta model serves as a broker reflecting the energy situation of the system to the sensor application.

# 4. CONCLUSIONS AND FUTURE WORK

We presented an event-based programming model to simplify the development of reflective components in WSNs and facilitate the service inspection process. Our future work includes formulating inspection aspects common to the sensor applications and evaluating their accuracy.

# 5. REFERENCES


[1] F. C. Delicato et al. Reflective middleware for wireless sensor networks. In *SAC'05*, New Mexico.

[2] A. Dunkels et al. Software-based on-line energy estimation for sensor nodes. In *EmNets'07*, Ireland.

[3] P. Grace et al. Dynamic reconfiguration in sensor middleware. In *Proc. of MidSens'06*, Australia.

[4] OSOA. `http://www.oasis-opencsa.org/sca`.

[5] A. Taherkordi et al. Programming Sensor Networks Using REMORA Component Model. In *DCOSS'10*, USA.