Toward Stable Software Architecture for Wireless Sensor Networks

Nader Mohamed
Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030, USA
nmohamed@stevens.edu

Haitham S. Hamza
Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln, NE 68588-0115, USA
hhamza@cse.unl.edu

Abstract
This research aims at developing an approach for architecting and communicating software systems in the context of wireless sensor networks. In particular, it focuses on developing software systems that can adapt and evolve while reducing the communication overhead and energy consumption needed to disseminate the developed software over the network.

1. Introduction
Networked sensor systems [2] are emerging technologies that have a multitude of applications in military, health care, national security, environmental monitoring, business environments, and many others. Wireless Sensor Networks (WSNs) consist of a large number of small nodes (sensors) networked via wireless links.

A sensor is a programmable device that consists of five components: sensing hardware, memory, battery, processor, and transceiver [1]. Sensor nodes have limited computation and communications capabilities that are constrained by the limited resources such as processing power, memory, and battery power.

The limited capabilities of the sensor nodes impose several challenges that distinguish WSNs from traditional networks. For example, known routing algorithms and Medium Access Control (MAC) protocols cannot be applied directly to sensor networks. In general, the development of both hardware and software for WSNs is commonly driven by the main requirement of minimizing energy consumption. The recent few years have witnessed an explosion of interest in WSNs evident by call for proposals, conferences, and published papers.

Similar to the challenges faced by routing algorithms and other networking functionalities, developing software for sensor networks is also bounded by the same limitations imposed by sensor nodes. However, the development of the software infrastructure for WSNs has relatively received much less attention in both sensor networks and software engineering communities. The objective of our research is to bridge the gap between network and software engineering communities by developing an approach for architecting software applications under the constraints of sensor networks and nodes.

2. Motivation and Contributions
Many real-life WSN applications may need to replace, extend, or adapt the software installed in deployed networks in response to evolving requirements or environmental changes. Without a careful engineering of the software, even a simple modification can cause the entire software to change. Consequently, the new software needs to be disseminated throughout the network. Depending on the software size, this process can be both time and energy consuming, which will adversely affect the lifetime of the sensor nodes and the overall network.

Our objective is to develop an approach to modularize sensor networks software into different types of modules based on the likelihood of their change throughout the lifetime of the network. By doing so, we can update, modify, or extend the required portion of the software without the need for communicating the whole software. Consequently, communication overhead, and thus energy-consumption can be significantly reduced.

3. Software Stability: A Background
Separation of concerns is a well-known and powerful mechanism to tackle complex software systems by dividing a complex problem into more manageable subproblems. In our approach, we differentiate between stable modules and other modules that are less stable or even unstable. Stability in the context of this work is defined as the likelihood of the software to change over the time in response to natural or predicted needs in the system.

To identify stable modules in the system, we adapt the notions of Software Stability Model (SSM, for short) [3]. SSM is a layered approach for developing software. In this
approach, each class in the system is classified as one of three layers: the Enduring Business Themes (EBTs) layer, the Business Objects (BOs) layer, and the Industrial Objects (IOs) layer. Based on their nature, classes are classified into one of these three layers.

EBTs are the classes that present the enduring aspects of the underlying business. Therefore, they are extremely stable and form the nucleus of the stable model. BOs map the EBTs of the system into more concrete objects. BOs are internally stable and can be externally adapted through hooks. IOs map the BOs of the system into physical objects.

4. The proposed Approach

The proposed approach consists of two main tasks: Architecture Development (AD) and Software Communication (SC). AD provides an approach for analyzing, identifying, and implementing the required software for sensor nodes and for the applications that will be deployed in these nodes. On the other hand, SC is the task of routing and disseminating the required software module(s) from a control unit (e.g. a base station) to the required sensor nodes.

4.1 Architecture Development (AD)

The architecture consists of a set of classes organized in three layers. Each layer provides services for the above layers using well-defined interfaces. The layers are sensor resources layer, nodes interaction layer, and applications layer, as seen in Figure 1.

The sensor resources layer consists of all classes that allow control the resource of individual node. For example, we can have an object that provides access to each sensor in the node. Similarly, a set of objects are used to monitor and control battery power. Other objects can deal with other functions such as sensing, communications, and storage.

The nodes interaction layer contains all classes that are related to generic services that deal with other nodes in the network. These generic services provide interfaces to implement different applications. Examples of these service classes are scheduling class, network resource discovery classes, adaptation class, and protocols classes. Therefore, all classes that can be used by different applications to deal with a group of nodes should be part of the nodes interaction layer.

Finally, we have the applications layer with application classes that construct the different task in the network. For example, the modules that constitute the application of sensing temperature should be included in this layer.

The software in each layer is then developed using the concepts of SSM. Thus, EBTs, BOs, and IOs for each software module in each layer are identified. When a change occurs, only the BOs updates and the new IOs will be routed over the network, instead of sending the overall software each time a small change (e.g. a single IO) is needed.

It should be noted that the defined three layers of Figure 1 have different levels of stability. That is, software modules of the sensor resource layer are more likely to be more stable compared to the modules of the application layer which will probably change in new and modified requirements. Modules of sensor resource layers will likely need to change only if we need to change the nodes themselves. The second layer will be less stable but it will not be changed unless we change a generic service or protocol. The applications layer will more likely change over time.

Figure 1. Stable software architecture for sensor

4.2 Software Communicating (SC)

When the sensor network system is designed using the approach discussed in Section 4.1, software changes will be localized to a limited and well-defined set of classes. Thus, future changes in the system will impact only these identified classes. These classes can then be individually transferred to the target node(s) to update their system. In our approach, communicating the required class can be achieved using the concept of active networks. In particular, we adapt and extend the Java-Object-Router (JOR) [4] to route individual classes through the network. JOR provides content-based object routing. Applying this technique will significantly reduce the cost of system and application updates across the sensor network.

References