DAViM: a Dynamically Adaptable Virtual Machine for Sensor Networks

Sam Michiels, Wouter Horré, Wouter Joosen, Pierre Verbaeten
IBBT-DistriNet Research Group, Department of Computer Science,
K.U.Leuven, Celestijnenlaan 200A, B-3001 Leuven, Belgium
{sam.michiels,wouter.horre}@cs.kuleuven.be

ABSTRACT

Sensor networks are being deployed for substantial periods of activity, and are being used by multiple applications with possibly diverse requirements. Since manually upgrading or updating sensor software is often impossible, run-time software reconfiguration represents a considerable success factor for many practical usage scenarios of sensor networks. This paper presents DAViM, the Distrinet Adaptable Virtual Machine and describes how it allows to customize sensor behavior, to extend its functionality and to execute multiple applications in parallel. We have evaluated the proposed architecture by implementing a proof-of-concept prototype on micaZ hardware. First results indicate that it is already feasible to run the DAViM core on micaZ hardware, while memory requirements of the full DAViM implementation are close enough to fit on more recent sensor hardware.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*Distributed Applications*; D.2.11 [**Software Engineering**]: Software Architectures—*Domain-specific architectures*

General Terms

Design, Management

Keywords

Sensor middleware, software architecture, adaptability

1. INTRODUCTION

Sensor networks must be dynamically adaptable to survive software bugs and changes in network conditions, application requirements, or available resources [16]. Sensors are typically deployed for substantial periods of activity and integrated in office buildings or warehouses, implanted in animals, or spread out in nature [4]. By consequence, installing a new sensor network may imply a considerable investment. In addition, sensor networks may produce data that are relevant for multiple purposes, which implies that multiple applications should be able to reuse the same sensor infrastructure.

From this perspective, a sensor network becomes a lightweight service platform on which service providers can install a wide variety of applications. Figure 1 sketches the context of such a service platform and illustrates how it is linked with end-user applications. Think, for example, of a chemical factory in which a sensor network has been deployed to manage operations and safety. The same sensor network can be used, for instance, by (1) a facility management application to control the temperature in rooms where inflammable substances are stored, (2) a stock management application to check in real-time the availability of particular supplies and to monitor incoming and outgoing products, and (3) a safety monitoring application to localize dangerous products and to detect when incompatible substances approach within a pre-defined perimeter. These applications make use of various services offered by the sensor network such as temperature monitoring, data querying, sensor localization, and distance measuring. In addition, each application may require different behavior and accuracy related to data collection, data aggregation, localization, or timing.

In this context, one of the key research challenges is to manage sensor networks in such a way that they can be dynamically customized to various (unanticipated) circumstances. Since resource limitations prevent sensors to have an extensive set of services pre-installed, sensor software should be dynamically reconfigurable [13]. This implies not only customizing the behavior of a sensor node, i.e. changing how a sensor operates by using the same functionality in different ways, but also adding and removing functionality, i.e. changing what the sensor node does.

The contribution of this paper is the presentation of DAViM, the Distrinet Adaptable Virtual Machine. DAViM is partly inspired by ASVM [10], but offers more possibilities for dynamic reconfiguration of sensor networks. DAViM allows not only running lightweight applications, but also allows adding or removing operation libraries. In addition, it allows multiple applications to execute concurrently on a node. The architecture has been validated by implementing a proof-of-concept prototype on micaZ hardware. Although the full implementation of the current prototype is slightly too large to fit on micaZ hardware, it shows that DAViM's memory requirements are modest enough to fit on more re-

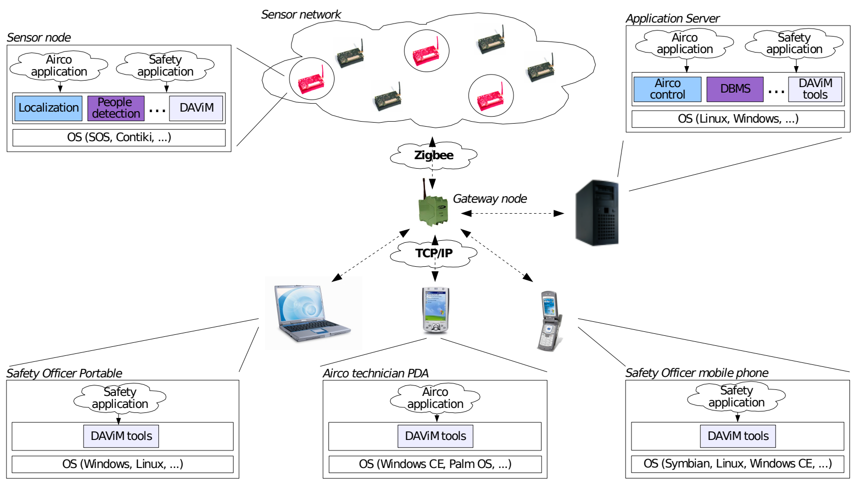

Figure 1: A sensor network as a lightweight service platform on which service providers can install a wide variety of applications, e.g. for airco control, localization and people detection. Application components are distributed over user terminals (e.g. a portable, PDA or a mobile phone), the sensor network and the gateway between the two. DAViM is installed on the sensor network. The DAViM tools on the user terminals support functionality for code injection and operation library management.

cent hardware.

The remainder of the paper is structured as follows. Section 2 summarizes the key requirements that DAViM must support. Section 3 describes related work in the context of dynamically reconfigurable sensor systems. Section 4 presents the DAViM architecture through exploring a representative use case and discusses its main contributions. Section 5 describes the proof-of-concept implementation of the architecture. Section 6 formulates the main conclusions and sketches directions for future work.

2. REQUIREMENTS SUMMARY

When considering a sensor network as a lightweight service platform, we can identify two key roles of the DAViM architecture. On the one hand, DAViM must enable service providers to use and extend the sensor network by installing various applications on it and retrieving data from its sensors. On the other hand, DAViM must support network administrators to manage the sensor network by distributing, storing and handling applications.

To implement this segregation successfully and to enable complex applications to execute on the sensor network, DAViM must offer support to as many applications as possible by extending the available set of operations if needed and by allowing applications to execute in parallel.

We summarize the key requirements of the system from both perspectives: customizability of sensor behavior, extendability of sensor functionality, and concurrency. The first requirement offers support to service providers, the others help manage the sensor network as a service platform.

2.1 Customizability of sensor behavior

First of all, sensor behavior must be customizable by dynamically injecting application code in the sensor network. This code is stored in every node of the network and its execution makes each sensor behave as preferred.

Secondly, the system must offer functionality that is specific for a particular application domain. This functionality is offered as operation libraries that can be called by applications running on the node. By consequence, applications can be very lightweight since the core functionality is already available in the libraries on the node.

2.2 Extendability of sensor functionality

Deciding what functionality will be installed on every sensor node is complex since requirements can vary during the life-time of an application, network conditions are highly unpredictable, and system resources are very scarce and variable. For example, when a more efficient sensor localization algorithm becomes available, it can be much more efficient to dynamically install this building block as needed.

In order to handle such changing circumstances, first of all, the system must be extendable by adding missing functionality when needed. The set of operations should be modularized in order to allow fine-grained extensions. At all times, the system should offer the most appropriate set of operations without pre-installing unneeded functionality.

Secondly, the system must be able to support the execution of added functionality, i.e. reacting to new types of events and handling these events correctly. Consequently, the system must be extendable by adding specific event handlers and corresponding state machines. For example, when functionality is added to communicate with a previously unused hardware sensor, the system must be extended to handle new types of hardware events.

Finally, the system must provide each application with an execution environment that offers the minimal but sufficient (sub)set of operations. Different applications may have highly diverse requirements, resulting in different amounts of system support needed. Some operations may be used by multiple applications, some operations may be application specific. Consequently, common operations must be decoupled from the application and concentrated in libraries that dynamically offer operation sets with respect to the system.

2.3 Concurrency

The system should allow multiple applications to use the set of operation libraries in parallel. Concurrency should be transparent: each application should execute as if it were the only one on the system and the only application using a particular operation library. The system must ensure safe execution and prevent race conditions and deadlocks; it must protect shared variables and schedule operations from different applications.

3. RELATED WORK

Related research in the domain of sensor reconfigurability typically focuses on either operating system reconfigurability or virtual machine support.

Research in dynamically reconfigurable sensor operating systems has shown that it is feasible to dynamically extend system services in a sensor network. The cost of extending sensor functionality mainly depends on the granularity of the code to be transported in the network. TinyOS/Deluge [6], for instance, replaces a complete system image which implies considerable network overhead to transport the update to all nodes in the network. Several approaches [12, 14, 7] have been proposed to reduce the code that has to be transported to be able to reconstruct the updated image. SOS [5] modularizes the operating system in a static kernel and multiple service modules for routing, localization, timing, etc. Modules are position independent code, have a limited size (4KB) and can be dynamically downloaded to update the system. Contiki [3, 2] allows to dynamically load binary modules in the standard ELF file format and a variant called CELF (compact ELF).

Research in the domain of sensor virtual machines shows that the update cost can be decreased considerably by installing on every node a virtual machine that offers a set of high-level operations related to a specific application domain. In order to customize sensor behavior, application scripts are downloaded to every sensor node. Such applications can use high-level operations that are available in the VM of every node, which makes them very lightweight. By consequence, sensor updates can be very efficient on condition that the operation library that is available on the node offers enough functionality to cover the life-time of the sensor network.

Deciding what functionality will be offered in the VM is complex given the highly dynamic circumstances in which a sensor network operates. ASVM [10], based on the work in Maté [9], allows customization of the available operations at deployment-time. VM* [8] provides a continuous update model in which the functionality of a sensor virtual machine can be incrementally extended as needed. VM* only includes the operations that are needed by the application that is running on the node, and extends the virtual machine when applications with extra requirements are installed.

The two related research domains, operating systems and virtual machines for sensor networks, focus on two requirements that were described in Section 2: customizability of sensor behavior and extendability of sensor functionality. The research for this paper can be positioned between the two state-of-the-art approaches for dynamic sensor updating. Virtual machine approaches enable highly efficient updates but are limited to a pre-defined application domain. Modular operating systems enable any service to be updated, but imply a higher cost since downloaded modules are more coarse-grained compared to a virtual machine application. The DAViM architecture offers a best-of-both-worlds solution by extending the idea of ASVM [10] to allow customization of available operations at runtime through dynamically loadable operation libraries. In addition, DAViM can be reused for various application domains by enabling concurrent execution of different VMs.

Dunkels e.a. have investigated the energy efficiency of various reprogramming techniques for wireless sensor networks [2]. They confirm that the combination of virtual machine code and native code using dynamically loadable modules, which is the approach taken in this paper, is beneficial for energy efficiency.

Balani e.a. present DVM [1], which takes a similar approach for dynamically updating sensor VMs. DAViM, however, is positioned in a broader and more open perspective by approaching sensor networks as lightweight service platforms. In view of this, DAViM also enables multiple applications to be executed in parallel. Furthermore, the DAViM architecture is independent from the underlying sensor operating system.

4. DAVIM ARCHITECTURE

This section presents the main components of the DAViM architecture (see Figure 2). The three key aspects that differentiate the DAViM architecture from related sensor virtual machines are: (1) a network coordinator that is responsible for distributing application code, operation libraries, as well as VM descriptions, (2) a scheduler that schedules operations over multiple VMs, and (3) a library manager that allows dynamic updates of the used instruction set of a VM.

The architecture is explained by exploring how it supports a representative use case that incorporates the requirements outlined in Section 2. We consider a sensor network deployed in a building of a chemical factory used for storage of chemicals (see Figure 1). The use case is presented incrementally by using three scenarios: the first scenario illustrates the basic VM support of DAViM, the second scenario shows how DAViM enables a VM to be dynamically updated and the third scenario describes DAViM support for executing multiple VMs in parallel. It should be noted that the DAViM architecture assumes that the underlying operating system provides dynamic memory and dynamic modules.

4.1 Basic virtual machine support

The sensor network is used by the air conditioning system to detect when to switch the system on or off. The sensor network informs the air conditioning system whenever an observed temperature rises above or falls below certain thresholds. These thresholds may change during the lifetime of the sensor network. The nodes in the sensor network may also change due to addition or removal of containers that have a node attached.

To support this basic application, the system must first of all include the application specific functionality that is needed by the air conditioning system. Secondly, it must be possible to dynamically customize the application to change the thresholds. Thirdly, new nodes that enter the network must be brought up to date to enable them to be used in the application.

This scenario is supported by deploying the DAViM sys-

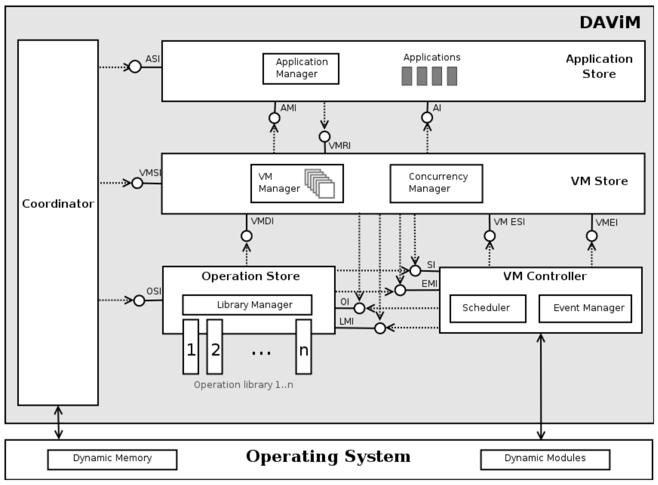

Figure 2: Overview of the DAViM architecture.

tem on the sensor nodes along with one virtual machine with an instruction set capable of sensing temperature and communicating with the air conditioning system. The application itself is implemented in the instruction set of this virtual machine.

To change the thresholds, the air conditioning system can install a new version of (a part of) the application on the gateway node. The Coordinator component on the gateway will inform the neighbors of the update and distributes the new code to them. Each node that receives the update via its Coordinator component passes the code to the Application Manager, which installs the newly arrived update and informs the VM Store of this change. The Concurrency Manager analyzes the new code to determine which shared variables are used. This information is stored to ensure safe execution of the different parts of an application. Afterwards the VM Manager reboots the virtual machine with the new application.

Whenever a new container with a sensor node attached enters the sensor network, the Coordinator component will cooperate with the Coordinator components of neighboring nodes to synchronize the state of the DAViM system. In case the entering node misses an application loaded on the other nodes, neighboring nodes will send it to the new node.

4.2 Dynamic adaptation of virtual machines

Since the energy cost to cool down the whole building can be quite high, and since only a fraction of the stored chemicals must be kept really cold, the management may decide to divide the building in several rooms that can be cooled down separately by the air conditioning system.

To support this new situation, the application on the sensor network needs to be able to not only detect when a treshold is exceeded, but also where this happens. This requires the instruction set of the virtual machine to be extended to with a localization instruction.

The instruction sets of the virtual machines are grouped in operation libraries of related instructions. These operation libraries can be loaded, changed or removed dynamically. To extend the virtual machine of our air conditioning system with a localization service, we implement this service as an operation library and load it into the DAViM system. The operation libraries are implemented as operating system modules. The Coordinator uses the mechanisms provided by the operating system to distribute and install the library in the sensor network. After the operation library is loaded, the Coordinator informs the Library Manager. The Library Manager is part of the Operation Store, to which also the operation libraries belong. Its task is to do the bookkeeping associated with the operation libraries. The most important task is to keep track of the mapping between the identifier of a library and the actual operating system module implementing it.

Before the localization library can be used, it has to be included in the instruction set of the virtual machine of the air conditioning application before it can be used. Again, this update is installed on the gateway node and distributed automatically by the Coordinator component of DAViM. The installation of the updated virtual machine is handled by the VM Manager.

4.3 Support for multiple virtual machines

To comply with new legislation, a safety officer, possibly externally hired, has to keep statistics of the presence of workers in the neighborhood of the most dangerous chemicals. To ease his work, the safety officer wants to deploy a service on the existing sensor network to monitor the presence of people in the building and to trigger an alarm when somebody stays in the neighborhood of a dangerous chemical for too long. In addition, it must be possible to monitor the distance between the chemicals to ensure that a safety perimeter around dangerous chemicals is preserved.

This new application is independent of the already deployed air conditioning application and, by consequence, they should be isolated. The new functionality needed for detecting people must be added without interfering with the already installed application. Furthermore, since the application also needs the functionality for localization, the system must support flexible and transparent sharing of this functionality in order to minimize memory requirements.

DAViM supports the isolation of applications by allowing multiple virtual machines to run concurrently. These virtual machines do not have to be loaded in advance, but can be loaded dynamically. This makes it possible to deploy

DAViM on a sensor network without predicting the future applications that will run on it. For the safety application, a new virtual machine is installed. To perform this installation, a description of the new virtual machine is loaded onto the gateway node. After arrival on a node, the addition of a virtual machine is handled similar to an update to an existing virtual machine.

The applications in both virtual machines, as well as the virtual machines itself can be updated without interfering with the application in the other virtual machine. The VM Controller ensures the running virtual machines operate smoothly. The Scheduler enforces a scheduling policy involving all loaded virtual machines. The Event Manager dispatches the events in the system (e.g. timer events, events generated by operation libraries, . . .) to the appropriate virtual machine.

In order to enable flexible and transparent sharing of operation libraries, the library identifier encoded in the byte code of an instruction set is decoupled form the actual library identifier known by the Library Manager. In addition, this enables dynamic updates of the operation libraries that are included in a virtual machine and loading of operation libraries without interference with running virtual machines.

4.4 Discussion

The DAViM architecture supports the requirements that were described in Section 2 as follows. First of all, sensor behavior can be customized by distributing applications in the network via the Coordinator component, and storing them in the Application Store that is available on every node. Secondly, the functionality offered by each sensor can be dynamically extended by distributing operation libraries (again via the Coordinator) and storing them in the Operation Store. The Library Manager decouples the operation libraries from the VM that is using them. In this way, an operation library can be transparently added to a VM, which allows to offer exactly those operations that are needed by the applications running on a node. Thirdly, the Concurrency Manager and the Scheduler enable multiple VMs to safely execute in parallel. In combination with the extension capabilities, each VM can be offered a customized operation library that can be used as if only one VM were present.

As such, we argue that the DAViM architecture combines the advantages of both reconfigurable operating systems and virtual machines: as long as the needed operation libraries are present, lightweight applications can be downloaded to a DAViM VM to reconfigure the system's behavior. When multiple applications need to be deployed in parallel, or when extra sensor functionality is required by an application, operating system modules can be downloaded to reconfigure the VMs itself. Because such an update implies more code to be transported, it is less efficient. This is, however, acceptable, given the low frequency of performing such major updates and the considerable added value for employing sensor networks during a substantial period of time.

5. PROTOTYPE IMPLEMENTATION

The presented architecture has been evaluated by implementing a proof-of-concept prototype. The implementation offers support to download and install applications on a VM, extend the functionality in each node by downloading operation libraries, and activate multiple VMs on a single sensor node. The current implementation allows to activate up to eight VMs in parallel, each of which can use up to eight operation libraries. Due to memory limitations of micaZ hardware, however, only one VM can be deployed at the moment.

The architecture relies on an underlying operating system that provides dynamic memory and loadable modules. We have chosen the SOS operating system for our implementation because it provides this features and has good support for micaZ hardware. The core architecture and the operation libraries are implemented as SOS modules, what enables the operation libraries to be loaded dynamically. Code snippets for the basic operation libraries and some algorithms related to the Coordinator, the Concurrency Manager, and the Scheduler could be reused from an existing port of ASVM to the SOS operating system.[1] The architecture does not imply a specific algorithm for the Coordinator component. The current implementation uses the mechanism provided by SOS for distributing operation libraries (SOS uses a variant of MOAP [15]) and Trickle [11] for distributing applications and VM descriptions.

The limited amount of dynamic memory that could be reserved on micaZ hardware currently prevents realistic field tests. MicaZ sensor nodes have available 4KB of RAM of which SOS reserves 1,5 KB for dynamic memory allocation. Experiments on micaZ nodes have shown that more dynamic memory is needed to use more than one simultaneously activated VM for a realistic application. Yet, simulations show that DAViM needs less than twice the amount of dynamic memory available on micaZ hardware. This is a promising result that motivates us to implement the prototype on more recent hardware.[2] Our expectations are that more recent hardware will solve current memory limitations.

6. CONCLUSION & FUTURE WORK

This paper has presented the DAViM architecture, which offers a blueprint for dynamically adaptable sensor virtual machines. The architecture offers support for three key requirements that sensor networks have to deal with: customizability of sensor behavior, extendability of sensor functionality, and concurrent execution of multiple applications. The paper has discussed the main components. By way of evaluation of the architecture, we have implemented a proof-of-concept prototype which provides promising indications that it is feasible to implement the described functionality on state-of-the-art sensor hardware.

With respect to future work, we will first of all evaluate and refine the architecture in the context of an industrial case study. Our research will gradually extend its focus from an individual sensor node and a single sensor network towards an end-to-end view. With end-to-end sensor applications we mean applications that are distributed over an enterprise back-end and one or more sensor networks with possibly different types of sensor nodes.

Developing and deploying end-to-end sensor applications in a realistic business context remains very complex. This complexity is partly caused by the highly resource limited, dynamic and heterogeneous environments in which sensor

[1]Networked and Embedded Systems Lab at UCLA (http://cvs.nesl.ucla.edu/cvs/viewcvs.cgi/ASVM/).

[2]For example TelosB (10KB RAM), XYZ (32 KB RAM) or iMote2 (32 MB RAM)

applications must operate. These specific characteristics of sensor networks require that sensor middleware (1) enables to compose software that contains minimal but sufficient functionality to meet application requirements, and (2) hides system details and heterogeneity as much as possible, while still being open for fine-tuning or customizing the underlying system in a controlled way.

The DAViM architecture fits in this research picture by focussing on dynamic code injection based on application specific requirements.

7. ACKNOWLEDGEMENTS

The authors are very grateful to Ann Heylighen and Christophe Huygens for their valuable comments on the paper and for proof reading the text. Sam Michiels is a Postdoctoral Fellow of the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), Wouter Horré is a Research Assistant of the Research Foundation - Flanders (FWO).

8. REFERENCES

[1] R. Balani, C.-C. Han, R. K. Rengaswamy, I. Tsigkogiannis, and M. Srivastava. Multi-level software reconfiguration for sensor networks. In *ACM Conference on Embedded Systems Software (EMSOFT)*, October 2006.

[2] A. Dunkels, N. Finne, J. Eriksson, and T. Voigt. Run-time dynamic linking for reprogramming wireless sensor networks. In *Proceedings of the Fourth ACM Conference on Embedded Networked Sensor Systems (SenSys 2006)*, Boulder, Colorado, USA, Nov. 2006.

[3] A. Dunkels, B. Gronvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *LCN '04: Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks (LCN'04)*, pages 455–462, Washington, DC, USA, 2004. IEEE Computer Society.

[4] J. Gehrke and L. Liu. Guest editors' introduction: Sensor-network applications. *IEEE Internet Computing*, 10(2):16–17, 2006.

[5] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the 3rd international conference on Mobile systems, applications, and services (MobiSys 2005)*, pages 163–176, Seattle, WA, USA, June 2005. ACM Press, New York, NY, USA.

[6] J. W. Hui and D. Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 81–94, Baltimore, MD, USA, Nov. 2004. ACM Press, New York, NY, USA.

[7] J. Jeong and D. Culler. Incremental network programming for wireless sensors. In *Proceedings of the First IEEE Communications Society Conference on Sensor and Ad-Hoc Communications and Networks (SECON)*, 2004.

[8] J. Koshy and R. Pandey. VM*: Synthesizing Scalable Runtime Environments for Sensor Networks. In *Proceedings of the Third International Conference on Embedded Networked Sensor Systems (SenSys'05)*, San Diego, CA, USA, Nov. 2005. ACM Press, New York, NY, USA.

[9] P. Levis and D. Culler. Maté: a tiny virtual machine for sensor networks. In *ASPLOS-X: Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95, New York, NY, USA, 2002. ACM Press.

[10] P. Levis, D. Gay, and D. Culler. Active sensor networks. In *Proceedings of the Second Symposium on Networked Systems Design and Implementation (NSDI 2005)*, pages 29–42, San Francisco, CA, USA, Mar. 2005. USENIX Association.

[11] P. Levis, N. Patel, D. E. Culler, and S. Shenker. Trickle: A Self-Regulating Algorithm for Code Propagation and Maintenance in Wireless Sensor Networks. In *Proceedings of the First Symposium on Networked Systems Design and Implementation (NSDI 2004)*, pages 15–28, San Francisco, CA, USA, Apr. 2004. USENIX Association.

[12] P. J. Marrón, M. Gauger, A. Lachenmann, D. Minder, O. Saukh, and K. Rothermel. Flexcup: A flexible and efficient code update mechanism for sensor networks. In *Proceedings of the Third European Workshop on Wireless Sensor Networks (EWSN 2006)*, pages 212–227, February 2006.

[13] J. Rabaey. Infrastructure as the Achilles Heel of Wireless Sensor Networks. Keynote at the Future of Autonomous Wireless Sensor Networks Symposium, Leuven, Belgium. http://bwrc.eecs.berkeley.edu/People/Faculty/jan/, June 2006.

[14] N. Reijers and K. Langendoen. Efficient code distribution in wireless sensor networks. In *Proceedings of the 2nd ACM internation conference on Wireless sensor networks and applications*, 2003.

[15] T. Stathopoulos, J. Heidemann, and D. Estrin. A remote code update mechanism for wireless sensor networks. Technical Report 30, Center for Embedded Networked Sensing (CENS), UCLA, 2003.

[16] Q. Wang, Y. Zhu, and L. Cheng. Reprogramming wireless sensor networks: challenges and approaches. *IEEE Network*, 20(3):48–55, 2006.