A General Architecture for Wireless Sensor Networks: First Steps

Bernhard Hurler, Hans-Joachim Hof, Martina Zitterbart
[hurler | hof | zit @ tm.uka.de]
Institute of Telematics
University of Karlsruhe, Germany

Abstract

Wireless sensor networks have become a very attractive research topic in recent years. Many academic and professional research groups made efforts to construct operative hardware devices and sophisticated software to meet the special conditions in their projects. But still there has been little done to create a general structure for smart sensors to cooperate and to offer their services to human or software clients. In this paper we present first results of our investigations in this topic. As a test scenario and source of inspiration we set up a sensor network prototype in an office situation, where the physical environment should be measured and adjusted according to specific conditions. In particular the light and humidity state of potted plants within an office should be autonomously adjusted to the plants' special needs as most research associates in our lab forget to care for their plants on a regular basis. On the basis of this prolific scenario we introduce a first stage middleware system architecture providing service distribution and accomplishment within wireless sensor networks. Core components of the architecture have been implemented in hardware and software to show the feasibility and abilities of our approach.

Introduction and Scenario

In recent years the field of wireless sensor networks has attracted considerable interest among numerous research groups all over the world. Efforts has been made to create small and power efficient hardware (e. g. SmartDust [WLLP01]) to allow small battery powered devices enhanced with sensor capabilities to communicate wirelessly and give information of the physical world. There are proposals as well for specialized software running on devices with limited power, memory and computation resources. TinyOS [LC02] is an example for an attempt to provide basic functions of an operating system on small devices.

The various approaches are quite successful in executing the specific tasks or example scenarios they are designed to. Nevertheless they tend to have a monolithic implementation and do not provide a generic architecture to implement new tasks or change ongoing tasks in a simple and well structured fashion. Thus, they constrain interoperability of components from different research groups.

Therefore we introduce in this paper a generic system architecture for sensor networks, which could function as a basis for a middleware component allowing easy and flexible access to the functions of a sensor network.

A sensor networks consists out of several small devices, which are autonomous in their communication and computation capabilities. All of them can be equipped with sensors and actuators to measure and alter physical values in their environment. To have a sensible and prolific test case for our middleware approach we designed a sensor network scenario with a couple of sensors and actuators. We constructed a flower pot capable of sensing humidity, a correspondent actuator granting the right amount of water for the flower, and a jalousie regulating the light conditions in cooperation with a light sensor (see fig. 1). All sensor (actuator) devices act autonomously and are expected to fulfil their respective primitive tasks (i.e. measuring or altering physical values). To accomplish the common goal of cultivating the flower they have to communicate with each other and react to static preconditions or dynamic change of the environment. Prior to any interaction, a binding of the actuator and some sensors is needed.

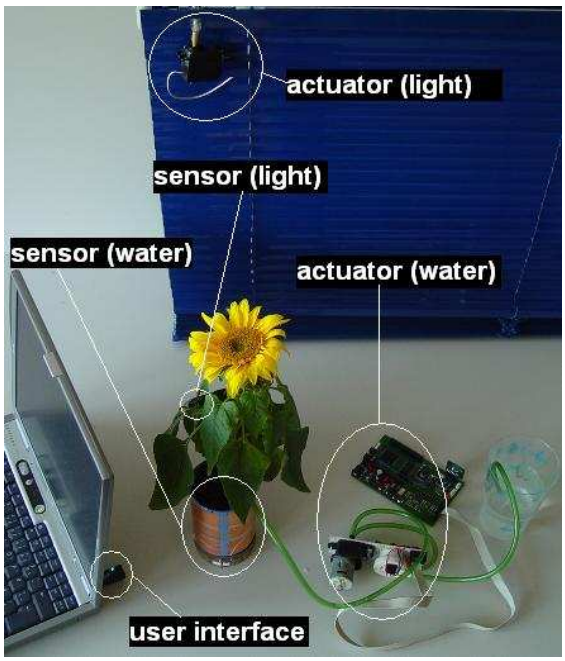

Fig. 1: Autonomously cultivated flower (with automatic water and light adjustment by a sensor network)

This scenario has most of the components and characteristics with which middleware systems have to deal. Multiple tasks have to be fulfilled whereas several small tasks are necessary to accomplish a common goal. The tasks may change over time and have to be altered or replaced either by machine or human interaction. The user issues a command ("cultivate this flower") to the sensor network and the sensor network self-organizes to fulfil this task.

In the following section we present our approach to a generic system architecture for sensor networks middleware. Then we give insights into the hardware and software implementation of the architecture and the flower pot scenario. The last section we give an overview of the current stage of our research efforts and our mid-term plans.

Architecture

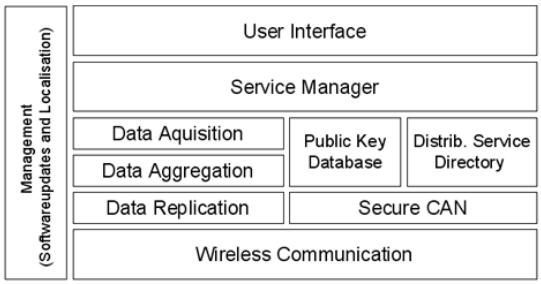

Fig. 2: Architecture for sensor networks

In fig. 2 we present a common architecture for sensor networks. In this short paper we concentrate on two main topics, which are most crucial for a functional sensor network: Service Manager and Distributed Service Directory.

The Service Manager is responsible for receiving and accomplishing services. It accomplishes the binding of actuators and sensors.

The Distributed Service Directory offers service lookup and is therefore necessary for the process of pairing sensors and actuators in a self-organizing environment. As the Distributed Service Directory is crucial for the whole Architecture, great importance has been attached to robustness and security issues.

Implementation details

The whole "flower" scenario consists out of four individual smart devices. Each of them is equipped with an Atmega128L microcontroller, a Bluetooth module (class 2), and device dependent sensors or actuators (see fig. 3). The microcontroller is responsible for the internal communication with its respective sensor or actuator and with the Bluetooth module. Furthermore it provides the smart part of the whole device, which is respectively the implementation of the architecture presented in the last section. In particular it provides the (external) communication with the sensor network via Bluetooth and the distribution and execution of primitive or complex services.

Fig. 3: "Saucer" for the flower pot with microcontroller and Bluetooth module

The services itself are written in a newly developed language, which allows a nested description of simple and complex tasks to be accomplished by single devices or by the whole sensor network. All service descriptions consist out of four basic service types. There are two primitive types (*query* and *order*) and two complex types (*conditional* and *repetitive*).

All devices provide at least one so-called primitive service, which is a simple query of physical data on sensor devices and a plain order to change a physical value on actuator devices. The *primitive query service* allows requesting a sensor device to send data with certain accuracy and maximum age to an interested network partner. The *primitive order service* allows requesting an actuator device to change a physical value with a certain priority to a given value, where it remains for the specified time period. In order to react reasonably to a changing environment a *complex conditional service* is available. It is based on one or two query services and dependent on the result of the query or the comparison of the two queries it causes the service manager to execute a list of service IDs (pointing to primitive or complex services). Finally there is a *complex repetitive service* to implement recurrent tasks of the sensor network. This kind of service repeats the execution of arbitrary (both primitive and complex) services given in a list (with service IDs) for a certain amount of time with a specific frequency.

All kinds of services reside in the Distributed Service Directory, where they can be stored by the user and retrieved by the smart devices, which have to perform certain services.

The DSD is based on S-CAN, an enhancement of Content Addressable Networks (CAN) presented in [RAT01]. CAN realises a distributed hash table. It has been enriched by a protocol for secure construction. The protocol only uses symmetric cryptography and establishes a shared secret between

neighbours in the Content Addressable Network. The protocol ensures, that takeover of some nodes only has minor impact on the whole distributed hash table and that no attacker can overtake a distinguished part of the hash table. It also ensures integrity by the use of redundancy similar to [CLA00] and [Kub00]. The basic idea of S-CAN is the existence of a so called Master Device which is used to bring new devices into the sensor network. The Master Device does not store any information about the sensor network, thus it is stateless and can easily be replaced. All necessary symmetric keys for communication between the Master Device and a member of the S-CAN are derived on the fly by the Master Device. A joining device gets a kind of join-ticket from the Master Device which enables the joining device to overtake a specific part of the distributed hash table and thereby join the S-CAN. Later on, the Master Device checks, if the joining device really joined at the specified place. The Master Device also assists in building trust between the joining device and its new S-CAN neighbours.

The Distributed Service Directory uses the S-CAN to store service descriptions. Insertion in the distributed hash table is done by simply hashing the service name and using the hash as an address in the virtual overlay space constituted by S-CAN. The same procedure is used to lookup services. Queries can be issued which include necessary parameter ranges etc. To avoid an unbalanced load on the nodes of the distributed hash table, the service names may be spread, for example by adding some location information to the service name before hashing or by building hierarchies which are encoded in the service string which will be hashed. This also makes lookups more efficient.

In the flower pot example there are two main services, which are executed at the same time. One service makes sure that the light is appropriate for the flower; the other one takes care of water provision. The first one is a *repetitive service* which continuously reruns two *conditional services*. Both of them evaluate a *query service* on the light sensor attached at the flower. Assuming that there is to much light for the flower the first conditional service makes the jalousie to decrease the solarisation; the second one does vice versa in the case of too little light. The second *repetitive service* responsible for appropriate humidity in the flower pot is similarly implemented. If changes in the service description occur, e.g. a new plant with different needs is bought, it could be propagated with very little effort to the sensor network. It then reacts immediately to the modifications without the need for “wired” contact to the smart devices.

To store new services in the Service Directory or the smart devices itself Bluetooth is used as communication technology. The devices are communicating via Bluetooth directly or via a dedicated central desktop computer. We decided to build up our flower scenario with the centralised approach. This allows observing the communication activities in the sensor network and gives the possibility to change network characteristics (like accessibility and communication speed) and evaluate the effects easily. Bluetooth is used for our test bed, because off-the-shelf hardware is cheap and available. Nevertheless the proposed architecture itself is independent of the underlying networking layer.

Current stage and mid-term goals

At the current stage we are able to distribute services via Bluetooth and make the devices to accomplish their common task autonomously. Both single primitive services and complex services can be changed or replaced easily and propagated wireless to the sensor network. The Distributed Hash Table is not integrated in the flower scenario yet. So the services are sent directly to the accordant devices. A simulation of the DSD-protocol is work-in-progress.

Still there is work to be done to define properly the communication protocol of the Service Manager responsible for exchanging services and user data between the different smart devices. The prototype implementation of the protocols serves as a proof of concept but has to be refined in the future.

The partial implementation of our general architecture and its application to the flower pot scenario gave promising results. Based on the knowledge we could obtain, while dealing with our prototype, we are confident to improve and complete the current prototype to a functional and useful middleware for wireless sensor networks.

References

[CLA00]: Ian Clarke, Oskar Sandberg, Brandon Wiley and Theodore W. Hong: “Freenet: A Distributed Anonymous Information Storage and Retrieval System”, in Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability, Berkeley, USA, 2000

[Kub00]: J. Kubiatowicz, et al: “OceanStore: An Architecture for Global-Scale Persistent Storage”, Proceedings of ACM ASPLOS, December 2000

[LC02]: Phil Levis and David Culler, “Maté : a Virtual Machine for Tiny Networked Sensors” , ASPLOS, December 2002

[RAT01]: Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp and Scott Shenker, „A Scalable Content-Addressable Network“, In Proceedings of ACM SIGCOMM 2001, August 2001

[VCCC+02]: P. Veríssimo, V. Cahill, A. Casimiro, K. Cheverst, A. Friday and J. Kaiser. “CORTEX: Towards Supporting Autonomous and Cooperating Sentient Entities”. Proceedings of European Wireless 2002, Florence, Italy, February 2002

[WLLP01]: B. Warneke, M. Last, B. Leibowitz, K.S.J. Pister, "Smart Dust: Communicating with a Cubic-Millimeter Computer", Computer Magazine, Jan. 2001. IEEE, Piscataway, NJ. pp. 44-51