*K. Aberer, G. Alonso, G. Barrenetxea, J. Beutel, J. Bovay, H. Dubois-Ferrière, D. Kossmann, M. Parlange, L. Thiele, and M. Vetterli*

# Infrastructures for a Smart Earth – The Swiss NCCR-MICS initiative –

*Karl Aberer* received his Ph.D. in mathematics in 1991 from the ETH Zurich. From 1991 to 1992 he was postdoctoral fellow at the International Computer Science Institute (ICSI) at the University of California, Berkeley. In 1992 he joined the Integrated Publication and Information Systems institute (IPSI) of GMD in Germany, where he was leading the research division Open Adaptive Information Management Systems. In 2000 he joined EPFL as full professor. Since 2005 he is the director of the Swiss National Research Center for Mobile Information and Communication Systems (NCCR-MICS). His main research interests are on distributed information management, P2P computing and semantic interoperability.

*Gustavo Alonso* is a full professor in the Department of Computer Science at the Swiss Federal Institute of Technology in Zurich (ETHZ). He holds degrees in Telecommunications Engineering from the Madrid Technical University (1989) and in computer science (M.S. 1992, Ph.D. 1994) from the University of California at Santa Barbara. Before joining ETH Zurich, he was a visiting scientist in the IBM Almaden Research Laboratory in San Jose, California.

*Guillermo Barrenetxea* received the B.Sc. and the M.Sc. in telecommunication engineering, in 2000, from the Public University of Navarre (UPNA), Spain. He received his Ph.D. degree in communication systems at the Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland, in 2005. From September to May 2000, he was a research assistant at Institut Eurecom, Sophia-Antipolis, France. From June to October 2000, he was an intern at British Telecom (BT), Adastral Park, Ipswich, UK. His research interests include network routing, sensor networks, and source coding.

*Jan Beutel* received his MSc and PhD in Electrical Engineering from the Swiss Federal Institute of Technology (ETH), Zurich in 2000 and 2005 respectively. He has been with u-blox AG, Zurich and spent time as a visiting researcher at the Berkeley Wireless Research Center. He is currently working as a senior researcher with Lothar Thiele at the Computer Engineering and Networks Lab (TIK) at ETH Zurich. His research interests lie in the development, deployment and fast prototyping of sensor network applications, integrated application protocols for ad-hoc networks and local positioning algorithms. He has been lead architect of the BTnode project, a platform for fast-prototyping of ad hoc and sensor networks.

*Jacques Bovay* received his Dipl. Eng. degree in Electrical Engineering from EPFL in 1972. He then held several positions in industry and administration, the latest one in 1992 as Deputy Head of the Telecom Division at the Swiss Federal Office for Communications. In 1996, he joined EPFL as scientific advisor. He was one of the initiators of the Swiss National Center of Competence in Research on Mobile Information and Communication Systems, of which he is now a member of the management committee.

*Henri Dubois-Ferrière* received the MS degree in Communications Systems from EPFL in 2000. He then worked for FastForward Networks in San Francisco for two years. From 2002 he was a research assistant at EPFL, working in the area of wireless sensor networks. He received the PhD degree in 2006 for his thesis on Anypath Routing.

*Donald Kossmann* is a professor for Computer Science at ETH Zurich. He received his MS in 1991 from the University of Karlsruhe and completed his PhD in 1995 at the Technical University of Aachen. After that, he held positions at the University of Maryland, the IBM Almaden Research Center, the University of Passau, the Technical University of Munich, and the University of Heidelberg. He is a co-founder of i-TV-T, a German company that develops SRM applications.

*Marc Parlange* received his BS in Applied Mathematics from Griffith University, Brisbane, in 1984, his MS in Agricultural Engineering from Cornell University in 1987, and his Ph.D. in Civil and Environmental Engineering from Cornell University in 1990. His teaching career started at the University of California-Davis in 1990 as an assistant, then later as an associate professor. He has been a full professor at Johns Hopkins University in Baltimore from 1996 to 2005, and is now a full professor at EPFL. His research interests are hydrology and fluid mechanics in the environment.

*Lothar Thiele* received his Dipl.-Ing. and Dr.-Ing. degrees in Electrical Engineering from the Technical University Munich in 1981 and 1985 respectively. After completing his habilitation thesis from the Institute of Network Theory and Circuit Design of the Technical University Munich, he joined the Information Systems Laboratory at Stanford University in 1987. In 1988, he took up the chair of microelectronics at the Faculty of Engineering, University of Saarland. He joined ETH Zurich as a full Professor of Computer Engineering, in 1994. His research interests include models, methods and software tools for the design of embedded systems, embedded software and bio-inspired optimization techniques.

*Martin Vetterli* received his Engineering degree from ETH Zurich, his MS from Stanford and his Ph.D. from EPFL in Lausanne. In 1986, he joined Columbia University in New York, first with the Center for Telecommunications Research and then with the Department of Electrical Engineering where he was an Associate Professor. In 1993, he joined the University of California at Berkeley, were he was Full Professor until 1997. Since 1995, he has been a Professor at EPFL. From 2001 to 2004 he directed the National Competence Center in Research on mo-

bile information and communication systems. He is also Vice-President for International Affairs at EPFL since October 2004. His research interests are in the areas of applied mathematics, signal processing, and communications.

**ABSTRACT**

The Swiss National Competence Center for Research in Mobile Information and Communication Systems (NCCR MICS or MICS) is a research initiative sponsored by the Swiss National Science Foundation to promote long term research projects in areas of vital strategic importance for Switzerland. The NCCR MICS covers a wide spectrum of topics in the area of mobile information and communication systems, ranging from information theory related to ad-hoc sensor networks to business models for pervasive computing, including network and routing issues, software and application development, and actual deployments of sensor networks. In this paper, we briefly present MICS as a whole, describe a major application in the area of environmental monitoring and discuss in some detail the hardware and software platforms developed in the Center.

## 1 SMART EARTH

Wireless sensor networks are changing the way we use information technology: information becomes embedded into our physical environment by means of miniature devices and computers, providing dense sensing close to physical phenomena. This information is distributed, processed, stored, and fed into software applications that act on the information provided. The physical environment becomes thus intertwined with the Internet information space, evolving into what we call the *Smart Earth*.

### 1.1 The NCCR MICS

In the recent years much progress has been made in theory, algorithms, and systems for sensor networks. These developments lie at the core of the Smart Earth idea. In spite of these advances, still much remains to be done to make sensor networks a practical reality. In this paper, we describe a large research initiative that aims at realizing the Smart Earth concept: the Swiss National Competence Center for Research in Mobile Information and Communication Systems (NCCR MICS, www.mics.org). The NCCR MICS is tackling all technical aspects of wireless sensor networks, from the study of fundamental principles (network structures, distributed algorithms, information and communication theory) to the development of platforms (wireless sensor technology, ad-hoc networks, in-network information processing, software verification).

### 1.2 Structure and Organization

The NCCR MICS is a nation-wide research center encompassing more than 40 faculty members across different Swiss universities and more than 90 Ph.D. students. MICS is currently in its second 4-year phase (from 2005 to 2009) after finishing a successful first phase from 2001 to 2005.

MICS is run by a Management Committee that acts as link between the project participants and the Swiss National Science Foundation. The management committee is assisted in this task by an external scientific board of 10 internationally renowned researchers that actively review MICS activities once a year. MICS is officially reviewed once a year by the Swiss National Science Foundation through a panel of international experts. MICS also counts on an Advisory Board that consults on higher level strategic issues and is composed of five people with ample management experience in university, research and/or business administration. An open scientific conference is organized every 6 months at different locations in Switzerland, where the work of all participants is presented in two or three intense days of keynotes, Ph.D. students' talks, posters, demonstrations, panels, and strategic meetings. Readers interested in attending or participating in these meetings are encouraged to contact any of the authors. These conferences coincide alternatively with the reviews of the scientific board and the Swiss National Science Foundation. There is also an annual summer school where external speakers are invited to give one week courses on a variety of topics related to the research areas covered in MICS.

MICS is organized into 4 clusters; each cluster encompasses several research projects and one or more application projects.

### 1.3 MICS Cluster 1: Theory of self organized, distributed communication and information

Cluster 1 addresses the basic theoretical aspects of sensor and wireless networks in four areas: information theory, network theory, distributed signal processing and distributed algorithms. The projects within this cluster explore a wide variety of issues from the trade-offs between data rates, reliability, bandwidth and energy consumption to mechanisms for signal reconstruction that can be used to accurately describe real phenomena using the data captured by a collection of sensors. This cluster also encompasses an application project in environmental monitoring, SensorScope (see below).

### 1.4 MICS Cluster 2: Mobile Communication and Processing Platforms

Cluster 2 deals with the technology required to cope with the challenges linked to the implementation and deployment of wireless ad-hoc networks: from basic routing to new technologies such as ultra-wide band communication. This cluster also explores two interesting applications. The first application involves using sensors to study the dynamics of rapid gravity-driven flows, such as avalanches and earth mass movements. The second application project builds distributed, self-organized, networked robotic olfactory systems for chemical plume mapping and odor source localization.

### 1.5 MICS Cluster 3: Networked Software Systems

Cluster 3 is devoted to the basic support necessary to develop applications that rely on sensor networks. The work in this cluster includes, among others, techniques to check the properties of software modules, using a combination of compile-time (off-line) and run-time (dynamic) analysis; the analysis of multi-threaded programs to detect errors or to issue warnings; exploring alternative architectures for sensor networks to facilitate deployment, monitoring and debugging; and proving communication protocols to be correct and secure. This cluster includes two application projects which deploy sensors in difficult-to-reach regions: the PermaSense project focuses on the perma-

radio and a microcontroller. The BTnode rev3 features an additional low-power radio, generic IO peripherals and switchable power conversion and distribution systems. The low-power radio is the same as on the Berkeley Mica2 Motes, making the BTnode rev3 a twin of both the Mica2 Mote and the older BTnode rev2. Both radios can be operated simultaneously or be independently powered off completely when not in use, thus considerably reducing the idle power consumption of the device. Being the only dual-radio platform for sensor networks available today, the BTnode rev3 is ideally suited for versatile and flexible functional prototyping of a broad range of applications with the tradeoff possibility of the two radios and the flexibility offered by ample memory resources (see Fig. 2). The dual-radio approach provides opportunities to create tiered architectures with high-bandwidth nodes bridging ultra-low-power devices like the Berkeley Motes to Bluetooth-enabled gateway appliances, or to investigate duty-cycled multi-front-end devices with wake-up radios or bandwidth-power-latency trade-offs.

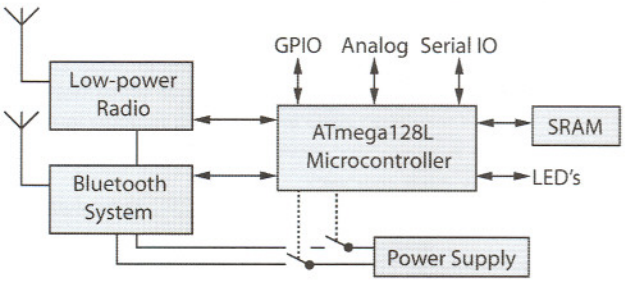


*Fig. 2 BTnode rev3 System Architecture*

The C-based BTnut system software is built on top of a threaded OS core for embedded systems, the open source Ethernut Nut/OS. The basic support of this OS core are primitives for scheduling multiple threads, basic memory management, events, synchronization, streaming I/O and device drivers that allows an extremely fast jump-start, even on complex applications. Compared to the popular TinyOS operating system [3], the BTnut system software does not require installing and learning new languages and tools (nesC) but uses plain C-based programming and is based on standard operating systems concepts that are familiar to most developers. In combination with a developer kit and accompanying tutorial as well as community support through a Wiki-based project web page and an archived mailing list, this ensures a quick jump-start and accelerated learning curve.

### 3.1.2 Education Outreach

The first steps in BTnode development are simplified by an in-depth tutorial with chapters on embedded programming, threaded execution models, debugging and Bluetooth communication. The tutorial is used in conjunction with a standard developer kit to jump-start students on the technology for individual research projects, course-work or labs (e.g. an annual graduate lab in embedded systems design with 120 participants) in only a few hours (www.btnode.ethz.ch).

### 3.1.3 BTnode based Testbed and Deployment-Support

The Deployment-Support Network (DSN) is a new methodology for developing and testing wireless sensor networks in a realistic environment [2]. With an additional wireless backbone

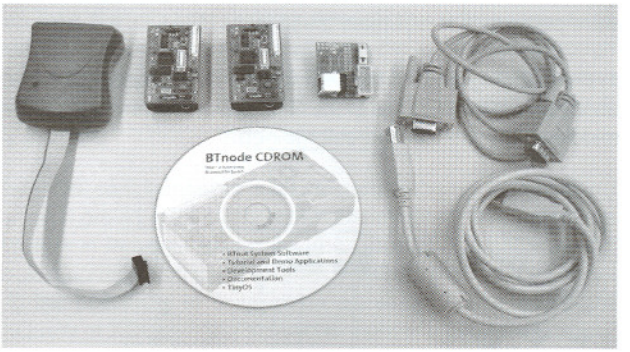


*Fig. 3 BTnode rev3 Developer Kit*

network, a deployed WSN can be observed, controlled, and completely reprogrammed over the air. The DSN provides minimal invasive visibility and control of a target network in a similar way as existing emulation testbeds, but overcomes the limitations of wired infrastructures [4]. As a result, development, test and experimentation can be conducted in realistic in- and outdoor deployments using the original hard- and software.

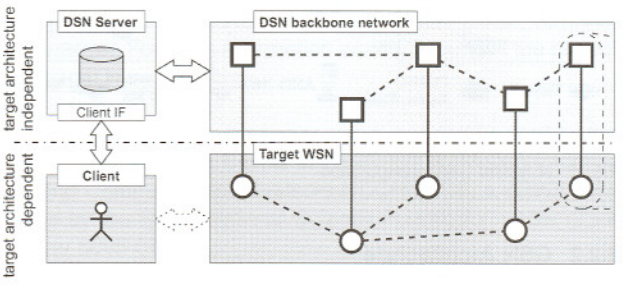


*Fig. 4 Deployment-Support Network Architecture*

Depending on the users' needs, a client can connect to a generic client interface on the DSN server (see Fig. 3) and execute commands or retrieve logged data from a network of DSN nodes. These nodes form an autonomous wireless network that is used to debug, reprogram, test and validate the operation of the "device under test", e.g. the target sensor network. Using this approach, automated testing and validation procedures become feasible while operating a set of nodes in a realistic scenario independently of fixed infrastructure such as Ethernet or power.

The architecture of the Deployment-Support Network is designed to cater to a number of different debugging and testing perspectives, ranging from detailed timing analysis at the interrupt level all the way to automated test case generation for conformance and reliability assessments at the application level. In a recent case-study done in cooperation with Siemens Building Technologies, the DSN has been successfully applied to the case of a fire sensor network [5].

## 3.2 Global Sensor Networks (GSN)

### 3.2.1 Overall Goals

Today we lack tools that would allow for rapid and efficient deployment of diverse sensor networks and for re-use and sharing of data generated by sensor networks at a global scale, despite of the similarity of the main tasks of processing, storing, querying and publishing data produced by a sensor network. The

frost region in the Swiss Alps, including the placement of sensors on vertical mountain walls (*cn.cs.unibas.ch/projects/permasense*); the focus of another project is on water and humidity monitoring and management in an arid part of the Indian subcontinent.

### 1.6 MICS Cluster 4: In-Network Information Management

Cluster 4 aims at supporting end-to-end data management for sensor and mobile networks covering all system layers and processing levels. The work in this cluster addresses the dire need for better tools (both actual as well as conceptual tools) to deal with the data generated by sensor networks. A common theme in this cluster is the use of a "declarative middleware" language and the interpretation of sensors as services. This cluster also takes a broader view on what a sensor is and generalizes sensors to any form of pull- or push-enabled data source. This cluster also has its own application project led by a group of architects that are using sensor networks in buildings as a way to minimize construction and renovation costs as well as optimizing energy consumption.

## 2 SENSORSCOPE

As an example of the type of applications pursued in MICS, we briefly describe SensorScope, a project that aims at better understanding the turbulent subgrid-scale physics in an urban environment. In order to provide fine-grained and continuous measurements to environmental scientists, SensorScope uses a Wireless Sensor Network (WSN) deployed at the university campus of EPFL. It measures key environmental quantities at high spatial resolution, for the purpose of modeling and understanding the energy exchange at the earth/atmosphere boundary. The WSN gathers data from a complete weather sensing unit specifically developed for the project. The design of the experimental setup considers the entire chain of requirements for a scientific atmospheric measurement campaign, including packaging, energy autonomy, sensor placement, and a diverse set of sensors.

The sensing unit is accessed through a wireless sensor node, a TinyNode module [6], consisting of a TI MSP430 microcontroller running TinyOS, and a Xemics XE1205 radio. Around this core module, we have designed a solar energy subsystem, giving energy autonomy to each station. The station also includes a sensor interface board accommodating seven external sensors, which makes the station capable of measuring nine different data inputs: air temperature and humidity, surface temperature, incoming solar radiation, wind speed and direction, precipitation, soil moisture and pressure at ground level. All sensors are placed on an aluminum skeleton which includes also a weatherproof housing containing the core module, solar energy board, and interface board. This weather station has been installed at over one hundred locations distributed around the EPFL campus. All the weather stations are periodically sampling their sensors and transmit the readings to a base-station through the network. The information generated in the network is stored in a database and shown in an interactive Web application based on Google Map (see Fig. 1), making the information available on-line in real-time (sensorscope.epfl.ch). The data are currently used in various atmospheric and environmental modeling projects. In the long term the aim of this work is to create a distributed sensing instrument that allows the generation of on-line available datasets in real-time, so that anybody can compute/analyze with.

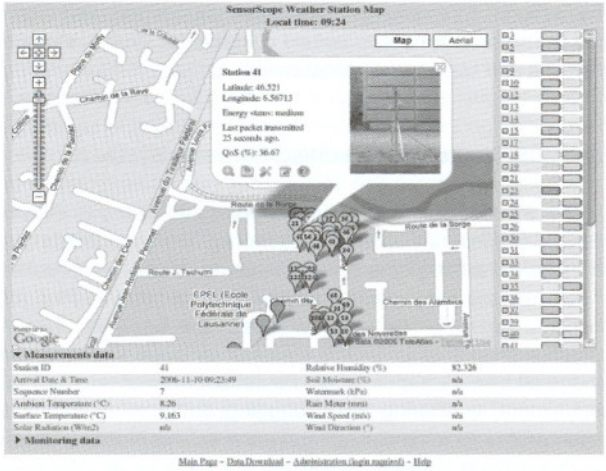


*Fig. 1 SensorScope Web Interface*

## 3 THE MICS SENSOR NETWORK PLATFORM KIT

To a large extent, the development of sensor networks and the associated applications is still ad-hoc. One goal of MICS is to develop a common platform that can be used across a wide range of applications such as SensorScope and that can allow scientist to quickly use the technology. The result of this effort is the Sensor Network Platform Kit (SNPK). The Sensor Network Platform Kit is intended to simplify deployment and use of sensor networks by providing a reliable and tested hardware and software platform for the sensor nodes along with an extensible and modular backend data management component. In contrast to related closed IP solutions (e.g. from industry) the SNPK builds on open technologies and targets a scientific audience.

The underlying infrastructure of the SNPK is a low-power wireless sensor node (MSP430-based) with TinyOS 2.0-based embedded software and tools, simple data gathering demo applications (duty-cycled multihop network), testbed and deployment support, a distributed data repository and data management tools, and support in form of tutorial, documentation and an installer CDROM (Fig. 3). In the following we describe three components of the SNPK: *BTnodes*, the testbed and deployment support; the distributed data repository and data management tools, *Global Sensor Networks* (GSN); and *SwissQM*, a currently developed virtual machine to facilitate advanced programming of sensor networks.

### 3.1 The BTnode Platform

The work on the BTnode platform [2] originates in MICS phase 1 and before, where it has been specifically designed for functional prototyping of wireless networking applications at different layers. Over the years the platform has matured through 3 hardware revisions, a well-supported threaded operating system software package, as well as tutorial and other documentation resources.

#### 3.1.1 The BTnode rev3 Architecture

The BTnode is a versatile, lightweight, autonomous wireless communication and computing platform based on a Bluetooth

goal of Global Sensor Networks (GSN) is to provide a middleware platform that facilitates these tasks [1].

In the following we provide an overview of the design considerations and of the features of the Global Sensor Networks (GSN) middleware (globalsn.sourceforge.net).

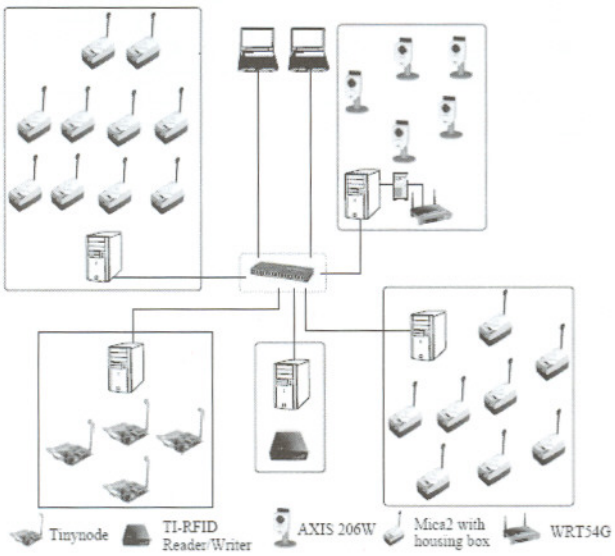


*Fig. 5 A typical GSN configuration*

### 3.2.2 GSN Architecture

GSN intends to facilitate in particular two tasks

1. The deployment of sensor networks and the publication of sensor data streams on the Internet
2. The integration of data streams from different and heterogeneous sensor networks over the Internet

In order to make use of GSN simple for application developers, it is based on a single minimal abstraction, the virtual sensor, specified in a fully declarative way. GSN nodes host virtual sensors and can be located anywhere on the Internet. GSN nodes are based on a container-based architecture, interpret the declarative specification provided for virtual sensors and allow runtime adaptation and reconfiguration of virtual sensors. GSN nodes are connected in a peer-to-peer data streaming network, where each node can be consumer and producer of data streams generated by virtual sensor hosted by the nodes. By this approach, publishing and re-using sensor data over the Internet can be performed with minimal effort, similarly as publishing and relating contents on the Web. GSN is based on a light-weight implementation with a small memory foot-print, low hardware and bandwidth requirements, and supported Web-based management tools. Fig. 5 shows a typical configuration of heterogeneous sensor networks connected through GSN.

### 3.2.3 Virtual sensors as Key Abstraction

A small set of powerful, easily combinable abstractions are key to successful middleware design. The key abstraction in GSN is the virtual sensor. Virtual sensors abstract from implementation details of access to sensor data and are the services provided and managed by GSN. A virtual sensor corresponds either to a data stream received directly from sensors or to a data stream derived from other virtual sensors. A virtual sensor can have any number of input streams and produces one output stream. The specification of a virtual sensor provides all necessary information required for deploying and using it, including metadata used for identification and discovery, the structure of the data streams which the virtual sensor consumes and produces, a declarative SQL-based specification of the data stream processing performed in a virtual sensor, and functional properties related to persistency, error handling, life-cycle management, and physical deployment. To support rapid deployment, these properties of virtual sensors are provided in a declarative deployment descriptor specified in XML. An example of a complete virtual sensor specification aggregating data from data sources from a setup as illustrated in Fig. 5 is given in Fig. 6.

```
<virtual-sensor name="room-monitor" priority="11">
   <addressing>
      <predicate key="geographical">BC143</predicate>
      <predicate key="usage">room monitoring</predicate>
   </addressing>
   <life-cycle pool-size="10" />
   <output-structure>
      <field name="image" type="binary:jpeg" />
      <field name="temp"  type="int" />
   </output-structure>
   <storage permanent="true" history-size="10h" />
   <input-streams>
      <input-stream name="cam">
         <stream-source alias="cam"  storage-size="1"
                        disconnect-buffer-size="10">
            <address wrapper="remote">
               <predicate key="geographical">BC143</predicate>
               <predicate key="type">Camera</predicate>
            </address>
            <query>select * from WRAPPER</query>
         </stream-source>
         <stream-source alias="temperature1" storage-size="1m"
                        disconnect-buffer-size="10">
            <address wrapper="remote">
               <predicate key="type">temperature</predicate>
               <predicate key="geographical">BC143-N</predicate>
            </address>
            <query>select AVG(temp1) as T1 from WRAPPER</query>
         </stream-source>
         <stream-source alias="temperature2"  storage-size="1m"
                        disconnect-buffer-size="10">
            <address wrapper="remote">
               <predicate key="type">temperature</predicate>
               <predicate key="geographical">BC143-S</predicate>
            </address>
            <query>select AVG(temp2) as T2 from WRAPPER</query>
         </stream-source>
         <query>
            select cam.picture as image, temperature.T1 as temp
            from   cam, temperature1
            where  temperature1.T1 > 30 AND
                   temperature1.T1 = temperature2.T2
         </query>
      </input-stream>
   </input-streams>
</virtual-sensor>
```

*Fig. 6 A XML-based virtual sensor specification*

### 3.2.4 Implementation

GSN follows a container-based architecture and each container can host and manage one or more virtual sensors concurrently. The container manages every aspect of the virtual sensors at runtime including remote access, interaction with the sensor network, security, persistence, data filtering, concurrency, and access to and pooling of resources.

The GSN implementation consists of the GSN-CORE, implemented in Java, and the platform-specific GSN-WRAPPERS, implemented in Java, C, and C++, depending on the available toolkits for accessing sensors. The implementation currently has approximately 20,000 lines of code. GSN is implemented in a highly modular way, in order to be deployable on various hardware platforms from workstations to small programmable PDAs, i.e., depending on the specific platforms only a subset of

modules may be used. GSN also includes visualization systems for plotting data and visualizing the network structure.

For deploying a virtual sensor, the user has only to specify an XML deployment descriptor as briefly outlined in Section 3.2.3 if GSN already includes software support for the concerned hardware and software. Adding a new type of sensor or sensor network can be done by supplying a Java wrapper conforming to the GSN API and interfacing the system to be included.

The effort to implement wrappers is quite low, i.e., typically around 100-200 lines of Java code. For example, the TinyOS wrapper required 150 lines of code. Our experience shows that new wrappers can be included usually in less than 1 day. Currently GSN includes already wrappers for the TinyOS family of motes (Mica, Mica2, Mica2Dot, TinyNodes, etc.), USB and wireless (HTTP-based) cameras (e.g., AXIS 206W camera), and several RFID readers (e.g., from Texas Instruments).

The GSN implementation is highly effective. As an indication, the processing time for one virtual sensor deployed on a GSN node is approximately 0.1ms on a standard workstation. Thus, in performance evaluations, we would typically host hundreds of virtual sensors on the same GSN node.

### 3.3 SwissQM

To facilitate even more the development of advanced functionality for sensor networks, the SNPK will in the future be extended with an additional module: SwissQM. SwissQM [8] is a flexible and extensible virtual machine that runs at the sensor nodes. SwissQM uses a bytecode language that is similar to Java bytecode with a few additions in order to be able to perform in-network data aggregation. SwissQM has been optimized for program size in order to minimize the overhead of distributing programs. It can concurrently run several programs, perform powerful data aggregation operations on-the-fly, and can be easily extended with user-defined functions. With SwissQM, it is trivial to, e.g., push down to the sensors data cleaning functionality like noise filters or window operators to minimize traffic and optimize the life time of the network. Being a programmable virtual machine, rather than, e.g., a SQL query engine, SwissQM is Turing complete and can be easily plugged into sophisticated software stacks that offer different interfaces to the outside world (including but not limited to SQL).

To facilitate the use of SwissQM, a gateway server is also being developed. The gateway collects data from the sensors and it compensates for functionality that cannot be implemented efficiently at the sensor. Furthermore, the gateway compiles SwissQM bytecode based on high-level declarative specifications, thereby carrying out optimizations and all planning that would be impossible to carry out by the individual sensors. The gateway of SwissQM is the main processing engine of the sensor network. The gateway is being implemented in Java on top of Concierge (www.flowsgi.inf.ethz.ch/concierge.html), an implementation of the OSGI specification tailored and optimized for small devices. Concierge allows the addition and removal of software modules at run time, a property that we will use to provide run-time extensibility. Through the gateway, we are building an SQL module, an XQuery module, and a web service module. What makes the gateway interesting is the fact that it can take the requests arriving through the different modules and apply multi-query optimization to the whole set. Through the gateway, we have demonstrated the ability to run more than a hundred concurrent user queries on a single sensor network by applying traditional optimization techniques [7].

SwissQM complements the efforts around the BTnode platform in that it provides an easily extensible target for integrating the sensors in larger IT infrastructures. Through the gateway, the project complements and extends the GSN platform by providing the possibility of using more advanced query languages (XQuery or XQueryP) on the sensor network, as well as providing extensibility.

## 4 CONCLUSIONS

This paper reflects many of the ideas developed over the years within MICS. We would like to thank all past and present participants of MICS for the collective and individual input. Regarding GSN, we would like to thank Manfred Hauswirth. Regarding BTnodes we would like to thank Matthias Dyer and the project team for tireless debugging and testing. Regarding SwissQM, we would like to tank Rene Mueller, Michael Duller, and Jan Rellermeyer. For more information on MICS, consult the project web page: www.mics.org

The work presented in this paper was supported (in part) by the National Competence Center in Research on Mobile Information and Communication Systems NCCR-MICS, a center supported by the Swiss National Science Foundation under grant number 5005-67322.

## 5 REFERENCES

[1] Karl Aberer; Manfred Hauswirth; Ali Salehi: A middleware for fast and flexible sensor network deployment. In: Proc. 32nd International Conference on Very Large Data Bases (VLDB2006), Seoul, Korea, 12-15 Sep 06.

[2] J. Beutel; M. Dyer; M. Hinz; L. Meier; and M. Ringwald: Next-generation prototyping of sensor networks. In: Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys 2004), pp. 291-292. ACM Press, New York, November 2004.

[3] P. Levis; S. Madden; J. Polastre; R. Szewczyk; K. Whitehouse; A. Woo; D. Gay; J. Hill; M. Welsh; E. Brewer; and D. Culler: Ambient Intelligence, chapter TinyOS: An Operating System for Sensor Networks, pp. 115-148. Springer, Berlin, 2005.

[4] G. Werner-Allen; P. Swieskowski; and M. Welsh: MoteLab: A wireless sensor network testbed. In: Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05), pp. 483-488. IEEE, Piscataway, NJ, April 2005.

[5] M. Dyer; P. Oehen; Thomas Kalt; Jan Beutel; L. Thiele; K. Martin; and P. Blum: Deployment Support Network – A toolkit for the development of WSNs. In: Proc. 4th European Workshop on Wireless Sensor Networks (EWSN 2007), pages to appear, January 2007.

[6] H. Dubois-Ferrière; R. Meier; L. Fabre Metrailler: TinyNode: A Comprehensive Platform for Wireless Sensor Network Applications. In: Proc. 5th Information Processing in Sensor Networks (IPSN 2006), Nashville, 19-21 Apr 06.

[7] Rene Mueller: International Conference on Mobile Ad-hoc and Sensor Systems (MASS 2007), Vancouver, Canada, 9-12 Oct 2006.

[8] Rene Mueller; Gustavo Alonso; Donald Kossmann: SwissQM: Next generation Data Processing in Sensor Networks. In: Proc. 3rd Biennial Conference on Innovative Database Research (CIDR 2007). Asilomar, California, USA. 7-10 Jan 07.