Data Management for a Smart Earth - The Swiss NCCR-MICS initiative -

Karl Aberer
School for Computer and Communication Science
EPF Lausanne
1015 Lausanne, Switzerland
+41 21 693 4679
karl.aberer@epfl.ch

Gustavo Alonso Donald Kossmann
Department of Computer Science
ETH Zurich
8092 Zurich, Switzerland
+41 44 632 (7306) / (2940)
{alonso, kossmann}@inf.ethz.ch

ABSTRACT

The Swiss National Competence Center for Research in mobile Information and Communication Systems (NCCR-MICS or MICS) is one of several research initiatives sponsored by the Swiss National Science Foundation to promote long term research projects in areas of vital strategic importance for the evolution of science in Switzerland, for the country's economy and for Swiss society. 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 (from architecture to geology). In this paper, we briefly present MICS as a whole and discuss in some detail two ambitious projects in the area of data management. The first project, XTream, addresses the whole life cycle of sensor based applications from the acquisition by sensors, aggregation and integration in gateways, storage in databases, generation of events that are relevant to users and applications, up to the subscription and consumption of events in a distributed architecture. The second project is Global Sensor Networks (GSN), which aims at enabling the rapid and efficient publication, sharing and interoperability of heterogeneous sensor data sources over large networks such as the Internet and P2P overlays.

1. INTRODUCTION

Wireless sensor networks are increasingly being used to monitor a wide range of natural phenomena and human activities. For instance, monitoring the watershed from glaciers to river mouths allows scientists to better understand the mechanisms governing the circulation of water and, thus, to improve prediction and management of this valuable resource. Similarly, wireless sensors and actuators in the walls of a building can be used to more efficiently control energy consumption while increasing the comfort of the users by creating individual microclimates. 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*.

Advances in technology are making it possible to acquire and store an ever increasing amount of data. Text and video are clear examples nowadays with individuals making vast amounts of information available and even producing it themselves now that the mechanisms exist to make it accessible to a broad audience. The key premise of our work is that sensor networks and sensor data will soon follow this trend. Hence, our goal is to provide dynamic mechanisms to cope with the resulting data deluge: rather than building large repositories with limited processing capacity, we focus on the publication, distribution, dissemination, and processing of sensor data.

This work is part of a large research initiative: the Swiss National Competence Center for Research in Information and Communication Systems (NCCR-MICS, *www.mics.ch*). The NCCR-MICS is tackling all technical aspects of 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), and their deployment in applications.

In this paper, we briefly present the NCCR-MICS, its structure and the broad research goals of its four clusters. Then, we concentrate on two concrete data management projects, *XTream* and *Global Sensor Networks* (GSN), and how they are addressing the data issues raised by the large scale deployment of sensor networks.

2. NCCR MICS

2.1 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, mainly from Computer Science and Electrical Engineering but also with students in mechanical engineering, architecture, and business departments. MICS is currently on 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 10 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, demonstration, panels, and strategic meetings*. 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 research is organized into 4 clusters:

- Theory of Self-organized, Distributed Communication and Information
- Mobile Communication and Processing Platforms
- Networked Software Systems
- In-network Information Management

Each cluster encompasses several research projects and one or more application projects. The application projects focus on real deployments of the technology developed within MICS.

2.2 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. This cluster aims at establishing the basis for the work in all other clusters in that it develops the necessary fundamental understanding of the problems associated with sensor networks. 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, which aims at obtaining sensor measurements for supplying boundary conditions for complex physical models of environmental phenomena such as wind and water flow (*sensorscope.epfl.ch*).

2.3 MICS Cluster 2: Mobile Communication and Processing Platforms

Cluster 2 deals with technology that is 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. Scientists involved in this cluster are also exploring how energy reduction can be achieved by using a more systematic cross-layer optimization, including additional upper and lower layers.

* Readers interested in attending or participating in these meetings are encouraged to contact any of the authors.

2.4 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 (*cn.cs.unibas.ch/projects/permasense/*) focuses on the permafrost region in the Swiss alps (including placement of sensors on vertical mountain walls), the other project's focus is on water and humidity monitoring and management in an arid part of the Indian subcontinent.

2.5 MICS Cluster 4: In-Network Management Systems

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. The two projects that are described in the next two sections (XTream and GSN) are part of Cluster 4. Additional projects in this cluster address topics such as algorithms and mechanisms for the on-line detection of events in sensor networks (as opposed to just gathering data), protocols for the efficient dissemination of information across sensor networks, efficient mechanisms for concurrent programming in embedded systems, and studies on the commercial aspects of sensor networks. This cluster also has its own application project led by a group of architects that are trying to exploit sensor networks in buildings as a way to minimize construction and renovation costs as well as optimizing energy consumption in buildings.

3. XTREAM

3.1 Overall Goals

The XTream project is a collaborative effort within the Department of Computer Science of ETH Zurich. Its main goal is to look at the complete acquisition, distribution, delivery, and processing chain of data from sensor networks, understanding sensors in the widest possible sense. XTream targets all levels of the system from the software support at the sensor platforms, the middleware at the gateway of a sensor network, event generation and subscription, data stream processing, and declarative generation of data processing in distributed applications that must process diverse and heterogeneous data streams.

In its first phase, the XTream project tackled the limitations of existing programming platforms and languages to adequate them to the needs of sensor data processing. Again, these limitations appear at all levels and a significant effort has been made to define a coherent architecture with the proper abstractions and support at each level. In what follows, we present the results obtained so far.

3.2 SwissQM

One of the biggest limitations of existing sensor networks today is the lack of an adequate software platform to program the sensors themselves. What is currently available is either too low level (a stripped down operating system) and, thus, difficult to program, or too high level (query based systems) and, thus, not flexible enough. We ran into this problem at the very beginning when we started experimenting with simple sensor networks. The learning curve was very steap to program sensors, the resulting software was difficult to maintain, and high level systems simply did not support any of the things we wanted to do.

A first key contribution of XTream was SwissQM [4]: 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).

SwissQM is a key element in the XTream architecture because it abstracts the sensor hardware behind a simple bytecode language. Now we are in a position to develop front ends that provide either bridges to high level programming languages such as Java, XQuery and SQL or links to optimizers residing at higher layers in the architecture.

3.3 Gateway

A central component in the XTream architecture is a gateway server. The gateway server is more powerful than sensor nodes; it typically has high (wired) communication to the Internet and less limitations in terms of energy consumptions, main memory, and processing. The gateway collects data from the sensors and it compensates for functionality that cannot be implemented efficiently by the sensors directly. Furthermore, the gateway compiles SwissQM bytecode based on high-level declarative specifications, thereby carrying out optimizations and all planing that would be impossible to carry out by the individual sensors.

The gateway of XTream is the main processing engine of the sensor network. As with SwissQM, developing the gateway involved developing first adequate software support. The gateway is being implemented in Java on top of Concierge (*www.flowsgi.inf.ethz.ch/concierge.html*, now available as open source), 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 [3].

3.4 SLETs and channels

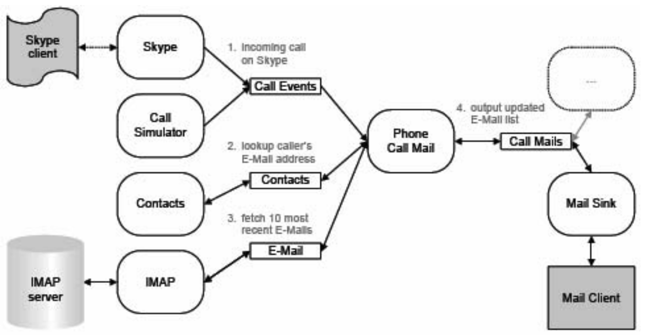

Figure 1. A first implemented XTream prototype combining streams from Skype and from an IMAP e-mail server

The ultimate goal of XTream is to create a development platform that will allow non specialists to create applications based on data streams. The concept is similar to mashups and built on similar primitives such as Web services. The architecture of XTream is divided into two components: Slets and channels (Figure 1). Slets are intended as the processing components that receive data, potentially from many channels, and produce data, potentially for many channels. Slets can be written in any programming language (e.g., Java); a declarative way to specify Slets is described in the next subsection. Channels are physical entities that abstract from the specifics of the underlying network, transport protocols, and data sources, offering a typed, XML based, query driven interface to mitigate the (potential) impedance mismatch between the programming language at the Slets and the data representation of the streams. They will also hide aspects such as access to remote data, refresh rates, access to multiple sources, etc.

Together, Slets and channels are used in order to compose complex applications from data streams and databases: Slets are the active components that encapsulate functionality and channels implement the dataflow between these active components. As a first approximation, the composition takes place in a workflow manner with a simple, yet powerful GUI that allows designers to plug channels into Slets and vice versa. These workflows are declarative and can be optimized in several different ways by the underlying runtime platform. An example of such an optimization is pushing down of operations from Slets into the gateway and possibly the sensor network; thanks to the design of SwissQM and the gateway such optimizations can indeed be implemented.

3.5 Programming Model

As mentioned in the previous subsection, XTream adopts a two-step programming model: Orchestration is carried out at the Slet/channel level and basic functionality is specified inside Slets, the gateway, and in the sensors themselves. For orchestration, any programming language can be used in order to program, say, Slets and any kind of data (e.g., comma separated values or RSS) can be shipped through channels. For best performance, however, XTream proposes extensions of XQuery in order to program Slets, the gateway, and even specify the operations that need to be carried out at the sensor level. XQuery is used for several reasons: First, XQuery makes it possible to process most data formats including, of course, comma-separated values and RSS. Second, the XQuery data model is based on sequences of items which is a perfect match for data streams. Third, XQuery is declarative, thereby enabling optimization and development tools (e.g., graphical editors and debuggers).

Unfortunately, the current XQuery standard is not powerful enough for XTream. Therefore, we are currently working on the following extensions, mostly in collaboration with Oracle and other industrial partners:

- Windows: This extension makes it possible to express tumbling windows, sliding windows, and landmark windows in XQuery in the same spirit as CQL (and other dialects) do for SQL. Window queries are particularly important at the gateway in order to carry out, e.g., data cleaning.
- Event generation: This extension makes it possible to compare two states of a database and notify users of critical state transitions (e.g., temperature dropped by more than ten degrees in ten minutes).
- Scripting: This extension adds error handling, variable assignment, and external function calls (e.g. Web service calls) to XQuery in order to allow more general programming [2].

3.6 Digital home scenario

As a proof of concept, and in collaboration with Siemens AG, we have developed a prototype that uses XTream to disseminate, process, control and display on a variety of hand-held devices information about a digital home (Figure 2).

The digital home scenario involves several house-related data streams (e.g., ringing of the door bell, storm warnings, status of home appliances) and personal data streams (e.g., E-Mails, information on phone calls, SMS, calendar events). All these data streams need to be processed in quasi-real-time, but with different requirements and priorities.

The prototype illustrates the goals and architecture of XTream. The different data sources are wrapped as slets that output data to channels. The channels are declaratively specified and the system forwards the data from slets to the relevant channels. Additional slets were developed in order to implement functionality such as merging streams, filtering out important events, or raising alarms when certain conditions occur. These slets forward their data to other channels that can be used again by new slets. User interfaces are hidden behind slets that read from channels. Certain slets control appliances; e.g., turn off the lights if all inhabitants have left the house. In this way, heterogeneity of both sources and sinks is hidden behind a uniform slet interface. The channels take care of the transport and distribution, as well as of storage of the data in transit. The Xtream software operates in a lab (a prototype house with real appliances). For testing purposes, Siemens provided a simulator, a configurable virtual digital home, as shown in Figure 2.

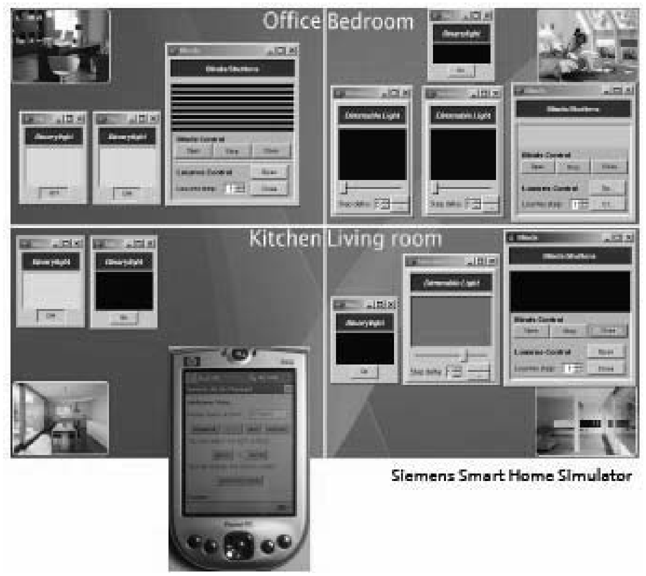

Figure 2. Simulator of a Digital Home controlled by the XTream prototype (in collaboration with Siemens)

4. GLOBAL SENSOR NETWORKS (GSN)

4.1 Overall Goals

As sensor network technology advances and the price of sensor networks rapidly diminishes, we can expect large numbers of sensor networks being deployed. This implies interesting opportunities and challenges for managing and sharing data produced by sensor networks at a global scale. Today we lack tools that would allow for rapid and efficient deployment of diverse sensor networks and for reuse 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 goal of Global Sensor Networks (GSN) is to provide a middleware platform that facilitates these tasks [1]. As a result, we expect to support developers of sensor networks in the rapid development of their applications and the simple publication of the data generated, and we expect to provide users an environment in which they can explore the sensor data space and potential applications in a way similar to the use of the current Internet.

In the following we provide an overview of the design considerations and features of a first system that has been developed recently and is being made available to the community as an open source release over Sourceforge (http://globalsn.sourceforge.net/)

The Global Sensor Networks (GSN) middleware provides a uniform platform for fast and flexible integration and deployment of heterogeneous sensor networks. The design of GSN follows four main design goals: Simplicity by using a minimal set of powerful abstractions which can be easily configured and adopted, adaptivity by enabling runtime reconfiguration when adding new types of sensor networks and data processing tasks, scalability and autonomy by using a peer-to-peer architecture, and light-weight implementation by ensuing a small memory foot-print, low hardware and bandwidth requirements, and web-based management tools.

4.2 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 they 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.

4.3 Data Stream Processing

In GSN a data stream is a sequence of timestamped tuples. The order of the data stream is derived from the ordering of the timestamps and the GSN container provides basic support to manage and manipulate the timestamps. These services essentially consist of the following components:

- a local clock at each GSN container
- implicit management of a timestamp attribute
- implicit timestamping of tuples upon arrival at the GSN container at reception time
- a windowing mechanism which allows the user to define count- or time-based windows on data streams.

In this way it is always possible to trace the temporal history of data stream elements throughout the processing history. Multiple time attributes can be associated with data streams and can be manipulated through SQL queries. In this way inherent properties of the observation process, such as network and processing delays, are made visible to applications for building their specific temporal abstractions on top of the available temporal data.

The production of a new output stream element of a virtual sensor is always triggered by the arrival of a data stream element from one of its input streams. Informally, the processing steps then are as follows:

- By default the new data stream element is timestamped using the local clock of the virtual sensor provided that the stream element had no timestamp.
- Based on the timestamps for each input stream the stream elements are selected according to the definition of the time window and the resulting sets of relations are unnested into flat relations.
- The input stream queries are evaluated and stored into temporary relations.
- The output query for producing the output stream element is executed based on the temporary relations.
- The result is permanently stored if required and all consumers of the virtual sensor are notified of the new stream element.

Additionally, GSN provides a number of possibilities to control the temporal processing of data streams, for example, bounding the rate of a data stream in order to avoid overloads of the system which might cause undesirable delays, sampling of data streams in order to reduce the data rate, and bounding the lifetime of a data stream in order to reserve resources only when they are needed.

GSN's query processing approach is related to TelegraphCQ (*telegraph.cs.berkeley.edu/telegraphcq/*) as it separates the time-related constructs from the actual query. Temporal specifications, e.g., the window size, are provided in XML in the virtual sensor specification, while data processing is specified in SQL. At the moment GSN supports SQL queries with the full range of operations allowed by the standard syntax, i.e., joins, subqueries, ordering, grouping, unions, intersections, etc. The advantage of using SQL is that it is well-known and SQL query optimization and planning techniques can be directly applied.

4.4 GSN architecture

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. This paradigm enables on-demand use and combination of sensor networks. Virtual sensor descriptions are identified by user-definable key-value pairs which are published in a peer-to-peer directory so that virtual sensors can be discovered and accessed based on any combination of their properties, for example, geographical location and sensor type. GSN nodes communicate among each other in a peer-to-peer fashion. Figure 3 depicts the internal architecture of a GSN node.

The virtual sensor manager (VSM) is responsible for providing access to the virtual sensors, managing the delivery of sensor data, and providing the necessary administrative infrastructure. Its life-cycle manager (LCM) subcomponent provides and manages the resources provided to a virtual sensor and manages the interactions with a virtual sensor (sensor readings, etc.) while the input stream manager (ISM) manages the input streams and ensures stream quality (disconnections, unexpected delays, missing values, etc.). The data from/to the VSM passes through the storage layer which is in charge of providing and managing persistent storage for data streams. Query processing is controlled by the query manager (QM) which includes the query processor being in charge of SQL parsing, query planning, and execution of queries (using an adaptive query execution plan). The notification manager deals with the delivery of events and query results to the registered clients. The top three layers deal with access to the GSN container.

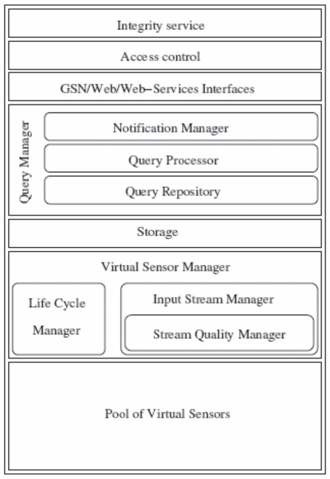

Figure 3. GSN architecture

4.5 Implementation

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 and is available from SourceForge. GSN is implemented to be highly modular 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 only has to specify an XML deployment descriptor as briefly outlined in Section 4.2 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., Texas Instruments).

The GSN implementation is highly performant. 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.

5. ACKNOWLEDGMENTS

NCCR MICS is a large collaboration effort in which the ideas of a large number of people have converged and been merged into a coherent whole. This paper reflects many of these shared ideas and, thus, we would like to thank all past and present participants of MICS for the collective and individual input. Regarding XTream, we would like to thank Irina Carbus, Michael Dueller, Tim Kraska, Rene Mueller, Christian Reichel, Jan Rellermeyer, Rokas Tamosevicius. Regarding GSN, we would like to thank Ali Salehi and Manfred Hauswirth. For information on the MICS projects, publications, pointers to all participants and an extensive list of publications and activities, 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.

6. REFERENCES

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

[2] Don Chamberlin, Michael Carey, Daniela Florescu, Donald Kossmann, Jonathan Robie. **XQueryP: Programming with XQuery.** *3rd International Workshop on XQuery Implementation, Experience, and Perspectives (XIME-P2006). Chicago, Illinois, USA. 30 Jun, 2006.*

[3] Rene Mueller, Gustavo Alonso. **Efficient Sharing of Sensor Networks.** *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS2007), Vancouver, Canada, 9-12 Oct 2006.*

[4] Rene Mueller, Gustavo Alonso, Donald Kossmann. **SwissQM: Next generation Data Processing in Sensor Networks.** *3rd Biennial Conference on Innovative Database Research (CIDR2007). Asilomar, California, USA. 7-10 Jan 07.*