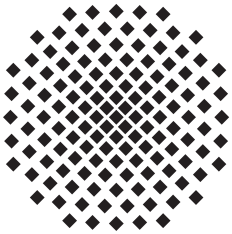


Universität Stuttgart

5. GI/ITG KuVS Fachgespräch „Drahtlose Sensornetze“

Stuttgart
17. und 18. Juli 2006

Pedro José Marrón (Herausgeber)

Technischer Bericht 2006/07
Fakultät 5: Informatik, Elektrotechnik und Informationstechnik
Institut für Parallele und Verteilte Systeme

Vorwort

Drahtlose Sensornetze bestehen aus einer großen Zahl von kleinen Rechnersystemen (Sensorknoten), die in der Regel mittels einer drahtlosen Kommunikationstechnologie miteinander vernetzt sind. Jeder Sensorknoten ist dabei mit einem eigenen Prozessor, Speicher und anwendungsabhängigen Sensoren ausgestattet. Die Eigenschaften dieser Sensorknoten und Netze implizieren eine Vielzahl von neuartigen Herausforderungen, die sich in einer regen Forschungsaktivität widerspiegeln.
Die GI/ITG-Fachgruppe „Kommunikation und Verteilte Systeme“ (KuVS) hat deshalb das Fachgespräch Sensornetze ins Leben gerufen, um Wissenschaftlerinnen und Wissenschaftler besonders aus dem deutschsprachigen Raum eine Gelegenheit zum Gedankenaustausch zu bieten. Der Fokus auf Diskussionen und der eher informelle Charakter dieser Veranstaltung eröffnen zusätzliche Möglichkeiten der Kooperation in diesem noch weitgehend US-amerikanisch geprägten Forschungsbereich. Diese Reihe wird durch das Leitungsgremium bestehend aus Dr. Thomas Fuhrmann (Universität Karlsruhe), Prof. Dr. Kurt Geihs (Universität Kassel), Prof. Dr. Holger Karl (Universität Paderborn), Prof. Dr.-Ing. Friedemann Mattern (ETH Zürich) und Dr. Hartmut Ritter (FU Berlin) organisiert.
Das bereits fünfte Treffen dieser Reihe findet in Stuttgart statt. Mit einem Programm, das einen Bogen von der Hardware über die Themen Routing, Middleware und Lokalisierung bis hin zu Programmierabstraktionen und Modellierung spannt, ist es gelungen, einen großen Bereich an Fragestellungen abzudecken. Wir bedanken uns deshalb ganz besonders bei den zahlreichen Autoren und Vortragenden aber auch den Diskussionsteilnehmern, die eine interessante Veranstaltung ermöglichen.

Stuttgart, im Juli 2006
Pedro José Marrón

Inhaltsverzeichnis

Middleware

Programmierabstraktionen

Daten-Frameworks und Modellierung

Hardware und Topologie

Routing und Lokalisierung

Supporting WSN Application Development Using Data type-centric Middleware Synthesis

Dennis Pfisterer, Carsten Buschmann, Horst Hellbrück, and Stefan Fischer

Institute of Telemactics, University of Lübeck, Germany
{pfisterer, buschmann, hellbrueck, fischer}@itm.uni-luebeck.de,
http://www.itm.uni-luebeck.de

Abstract. Writing applications for wireless sensor networks presents a complex domain that requires expertise from various fields. Existing middleware frameworks alleviate this problem by hiding low-level networking aspects but are neither comprehensive nor flexible enough. Our system called FABRIC addresses this problem by proposing a data type-centric middleware synthesis framework for heterogeneous devices. While hiding networking aspects and complexity of distributed systems, it still offers the required flexibility because data handling can be differentiated on a per-type basis. This approach especially addresses the unique requirements of resource constrained devices. In addition, FABRIC supports a clear separation between application and framework development, thus supporting a less error-prone development process.

Introduction

Developing applications for distributed systems challenges the developer with a number of peculiar problems. This is especially true for large, complex distributed systems. Hence, hiding these intricate communication aspects from the developer is beneficial in order to reduce the overall design complexity of applications.

This led to the development of a large number of function or service oriented middleware concepts like CORBA. They often rely on the client-server paradigm and a stable communication infrastructure. With the advent of ad-hoc networks, these assumptions were no longer valid and especially for wireless sensor networks (WSNs) data-centrism has proven to be advantageous [1]. As a result different middleware concepts were adapted to the needs of WSNs. However, as we discuss in [2], these are either not comprehensive or not flexible enough.

The authors believe that especially on resource-constrained devices like wireless sensor nodes, the handling of data including its communication should be varied according to its semantics, i.e. its meaning to the application. However, semantics are difficult to describe, vary between applications and after all are only known to the developer.

Consequently, we here propose differentiated data handling based on the type of data. The underlying idea is that data of certain semantics is of a particular,

usually complex type. The developer can now annotate the corresponding data type definitions with treatment *aspects* like reliable transport or confidentiality.

Conceptually we distinguish so called *domains* that embrace related aspects. Apart from the aforementioned (reliability and security), arbitrary other annotation domains can be supported. Examples are data aggregation or routing mechanisms. The annotations of the type definitions, which we favor to be written in the W3C Schema definition language, determines the way data is treated. For each annotatable aspect, a number of *modules* provide the corresponding functionality. Modules of the same aspect may target different hardware platforms, programming languages or be of different complexity. For a given annotated type definition, the framework picks the *best* modules based on each modules self-description. A code generation processor reads the schema, and passes both type definitions and annotation attributes to a set of selected modules. These then generate source code for the functionality they are in charge of, e.g. reliable messaging or security.

The key advantage of the generation of custom middleware code for each application is that it results in lean code featuring exactly the required annotated functionalities for the defined data types. This is especially desirable for resource constrained devices like wireless sensor nodes because their program memory is much too small to contain the full extent of most comprehensive static middleware solutions.

The remainder of this paper is structured as follows. Section 1 presents our approach to a type-centric middleware synthesis framework we call Fabric. Section 2 shows how Fabric supports the development of sensor network applications. The paper is concluded by a summary.

1 Architecture

Fabric significantly eases the development process of sensor network applications by relieving the application developer from dealing with low-level networking issues without sacrificing data type specific treatment. Instead, high-level data management operations are available for transmitting and receiving application data structures. Unlike other middleware systems, the Fabric framework is capable of customizing the generated middleware code on the basis of individual type definitions. A high-level data type description language is complemented with annotations (a concept we call *type annotation*) to parameterize the synthesis process.

Figure 1 shows an overview of the Fabric architecture. It has been designed to support multiple hardware architectures, heterogeneous environments and the interconnection of them by synthesizing application-layer gateway functionality. We distinguish between two roles: an application developer and a framework developer. An application developer's primary interest is an easy to use, flexible system. The framework developer customizes the generic Fabric-system and implements *modules* that provide the functionality available to the application developer.

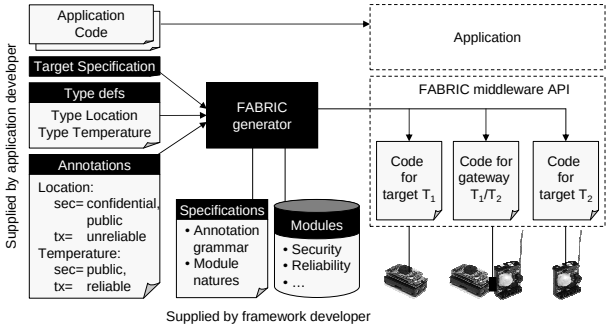


Fig. 1. The FABRIC architecture

A code generation step transforms the type annotations into customized middleware source code such that the resulting code complies to the given target specification. The generation process consists of two major steps: first the appropriate modules for each data type are selected and second they are invoked to actually generate source code. In this paper we focus on the application developer's role, FABRIC's internal mode of operation and the tasks of the framework developer are presented in detail in [2].

2 Data type-centric application development

From the application development view the system needs the following input: the annotated type definitions and a target specification describing the target hardware platform. Next, the FABRIC-generator synthesizes the middleware functionality including the API functions that can be compiled, linked together with application code, library functions as well as the OS or the firmware of sensor nodes. We will have a look at these input sources in more detail.

As mentioned above, annotation aspects are grouped into domains. Data type definitions are annotated with aspects that then influence the treatment of each individual data type by the synthesized middleware. Figure 2 depicts this process with the two type definitions with exemplary aspect annotations: "confidential" and "public" from the domain "security" and "(un)reliable" from "tx". The set of annotatable aspects is completely customizable, yet for a concrete FABRIC-system it is given by the framework developer.

The above example is representative for two distinct annotation cases. The *Temp* data type has at most one annotation per domain (domain security: public, domain tx:reliable) while the *Location* data type has more than one annotation for one domain (domain security: confidential and public). The first case specifies unambiguously how this data type is treated by the generated middleware and no run-time decisions are necessary. Yet, in second case, run-time decisions on data treatment are possible. The generated middleware will contain code for both security schemes and options for selecting the active one. It is then up to

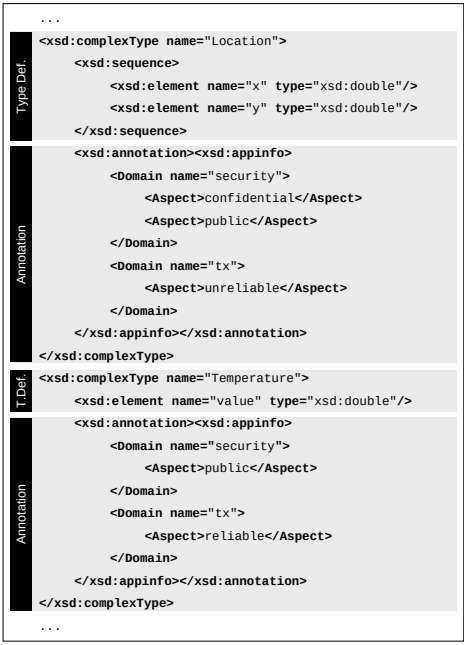


Fig. 2. XML-Schema annotation

the application developer to select the appropriate method depending on the context.

The type annotation concept requires a data type definition language and the ability to annotate each data type. Our current implementation uses XML and XML Schema technologies for the type definitions, the annotation specification, the annotations and the target specification. XML Schema is a widely accepted standard that allows for a convenient type definition by the developer and a coherent realization of the type annotation concept. *Complex type* definitions specify the application data types, which may be arbitrarily structured, though the current implementation does not yet feature local complex type definitions and nested annotations.

XML Schema permits an annotation of nearly all schema-elements with user-defined tags. We exploit this feature by attaching XML documents to each data type definition containing the annotation for each data type. To achieve extensibility and to ensure the presence of required annotations, the annotation grammar is given by an XML Schema supplied by the framework developer.

An example how the application developer specifies the annotated data types is depicted in figure 2: the data types “Location” with x and y coordinates and “Temp” with a single value are defined using complex-type definitions. These additionally incorporate an XML document with annotations per data type. Note that the annotation data is only used for code generation at compile-time and does not have to be stored on the devices at run-time.

In addition to the annotated data types, the generator needs a target specification in order to produce the adequate code for the target. It consists of so called *natures* (e.g. the platform, programming language, etc.). From these and the annotations the generator determines the best fitting modules from the set of available ones (for details see [2]).

Finally the application developer writes the code for his target application and benefits from the type-specific API of the middleware. Without the need to change a single line of application code, the treatment of a data type can be altered by simply changing its annotation. A set of basic functions for sending and receiving instances of the individual data types, selecting the run-time options described above and possibly parametrization methods for some aspects is available in the generated API. This allows for easy application development where the developer is relieved from dealing with error-prone issues like serialization, security or packetization of the data type. The concrete generated API may look different depending on the target specification and the features available in the corresponding programming language, hence we do not present the interface for a specific language here.

3 Conclusions and Future Work

In this paper we present how the development of applications for wireless sensor networks can be supported by Fabric, a framework that provides data type-centric middleware synthesis for heterogeneous devices. While hiding networking aspects and complexity arising from distribution, it still offers the required flexibility because data handling can be differentiated on a per-type basis. Starting from an XML Schema data type definitions augmented with annotations, application scenario specific code is generated.

This incorporation of application knowledge results in lean, custom tailored code. It also speeds up the development process by allowing high level data modeling. Fabric supports a clear separation between application and framework development. The application developer defines the application data types and the annotations in a high-level language, while the framework developer backs this information with target platform specific modules. By supporting several targets, application developers benefit from platform independent development because his code can remain virtually unchanged.

We are currently finalizing the core implementation of Fabric and plan to support platforms such as the Mica2 and ESB wireless sensor nodes in near future.

References

1. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. SIGMOBILE Mob. Comput. Commun. Rev. **6**(4) (2002) 59–61
2. Pfisterer, D., Buschmann, C., Hellbrück, H., Fischer, S.: Fabric: Towards data type-centric middleware synthesis. In: Proceedings of the Fifth Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2006). (2006)

Designing Industry-grade Middleware Services for Wireless Sensor Networks

Martin Fenne, Christian Scholz, and Thomas Wieland

University of Applied Sciences Coburg
Department of Electrical Engineering and Computer Science
96450 Coburg, Germany
{fenne,scholz,wielandt}@fh-coburg.de

Abstract. Wireless sensor networks could be used for sensing tasks in industrial automation and process control. For becoming attractive to industrial developers, however, more abstract and more powerful programming models are required to provide standardized and portable system abstractions. This programming model is best realized as a middleware layer shielding the operating system basics and offering advanced services to the application. In this paper, we first identify several decisive requirements for such a middleware in an industrial context. These requirements are then compared with numerous existing middleware approaches that we have analyzed. Finally we propose key design elements based on these considerations that could be guidelines for future investigations on the way to an industrial adoption of sensor networks.

1 Introduction

Today wireless sensor networks (WSNs) are in a transition state from research prototypes to practical and commercial usage. A lot of research has been conducted to make those networks more flexible and reliable. The WSN paradigm assumes that a WSN consists of a dozen to hundreds of individual nodes (called "motes"), connected wirelessly, sharing their data in a multi-hop manner.
At the same time there is the huge market of automation and industrial production where a plethora of different sensor types is currently in use. Most of these sensors are wired to a machine or factory floor network; some of them are already connected wirelessly. Of course, the wireless sensors in automation do not follow the WSN paradigm at the moment. Industrial sensors can reside in nodes with own processors and memory, but are typically connected in star or bus topology. Each of them has one designated task, whereas the power of a wireless sensor network is only unleashed in mesh networks combining measurements from several nodes.
The contemporary research activities in WSN focus on gathering and forwarding data, usually allowing just one application for the entire network. In these cases, the protocols and applications are built by the same team, following a common and thus closely coupled design idea. The fundamental services that the TinyOS [1] operating system provides, for

example, are just enough to set up a very simple network. Any more sophisticated routing procedure or energy-saving strategy must be selected individually and added as a module. However, if wireless sensor networks should be attractive for industrial users, an out-of-the-box readiness has to be achieved. We believe that the practical and wide-spread adoption of sensor networks need powerful and robust software services running on the nodes. This software should offer a much more abstract programming model such that the industrial developer can concentrate on his own application, without needing to know about all the hardware and protocol details of the mote.

There are some of the specific characteristics of sensor networks that are less important in industrial contexts. One of them is *mobility*. As production equipment is generally stationary, the sensors are not supposed to move around in a large radius. They may be mounted on a moving part of a machine, but these normally have just a range of a few meters. In some cases, however, like item tracking between production lines, mobility can be an issue.

Factories are well-guarded buildings or sites to which only authorized personnel has access. So *data access control* and encryption on the air link is also of minor importance. Transmissions of automation data must usually not fear hacker interference, apart from acts of sabotage.

Moreover, the production equipment needs lots of electrical power. Hence it is in most cases no problem to provide the sensor nodes with a power supply. The classical concern of mote design, *energy efficiency*, is thus uncritical for most industrial applications.

2 Requirements on Middleware Systems

As discussed earlier [2]-[4], middleware systems for wireless sensor networks must meet several functional and non-functional requirements. Most of them are not really specific to WSNs, but apply to any highly distributed computing system. Some applications may do without one or another condition, but as a middleware should as generic as possible, an all-purpose view is desirable.

We list only the most important requirements here, dividing them in three groups. The first group comprises general requirements that a distributed system should fulfill to make it usable in professional environments:

- **Performance**: In general, the middleware should introduce a delay to the overall communication chain that is in the same order of magnitude as the operational performance without this software layer.
- **Adaptability**: The middleware system should be adaptable to changes in applications' and users' requirements or changes within the network (e.g. a different transport protocol).
- **Maintainability**: The middleware shall make it possible to improve it and correct errors or software failures as easy as possible.
- **Scalability**: The system should be scalable to the number of motes and number of users/traffic.

The second group lists some requirements that lead to additional functionality, but are only considered as optional:

- **Location awareness**: It means the ability of the middleware (and the programs running on a particular mote in general) to determine its location. This can be done topologically, i.e. relative to the other nodes of the network, or geographically, i.e. in terms of an absolute geographical position.
- **Addressing**: The middleware should be aware of the addressing scheme employed in the network. If there is not enough support by the underlying base functions or the employed concept is insufficient, the middleware may introduce its own alternative addressing scheme and use it consistently on all participating motes.

Moreover, there are some requirements that are usually not brought in connection with WSNs, but which we regard as crucial to gain industrial acceptance.

- **Monitoring/Logging**: The middleware should provide services that enable an effective monitoring of the application. This surveillance should be possible in real-time, i.e. during operation, and a posteriori, i.e. after operation by creating and analyzing log files. The monitoring should not be limited to any individual node nor should it require to investigate each node separately; it should instead allow to look at the operation of network as a whole.
- **Updatability**: Installation and replacement of application software over the air, taking the available applications and versions into account. The update should be able to be initiated from a gateway node and to be resumed if the connection to a mote has been interrupted.
- **Task sharing/network partitioning**: The motes in the network should be able to switch from one application to another if necessary. After deployment the network could work as a whole, but should also support partitioning in the sense that some parts of the network perform different tasks than the others.
- **Support of role concept**: Various roles can be assigned to the nodes, either for establishing hierarchies for a more efficient node management and networking (vertical role assignment) or for supporting different tasks of the motes in terms of sensor equipment or software components (horizontal role assignment).

3 Middleware Classification

The research in wireless sensor networks led to various approaches of middleware software, but only a few are really applicable. We analyzed more than 50 middleware systems from nearly ten years of research. Most works were only for educational or research purposes, hence they are only publicly available in paper form. We classified the different approaches in four main categories (figure 1).

The family of **low-level** middleware only capsules certain functionalities of the underlying operating system or plays the role of an operating system. These middleware programs simply try to make common functionalities like routing, energy awareness, data storage, as well as aggregation

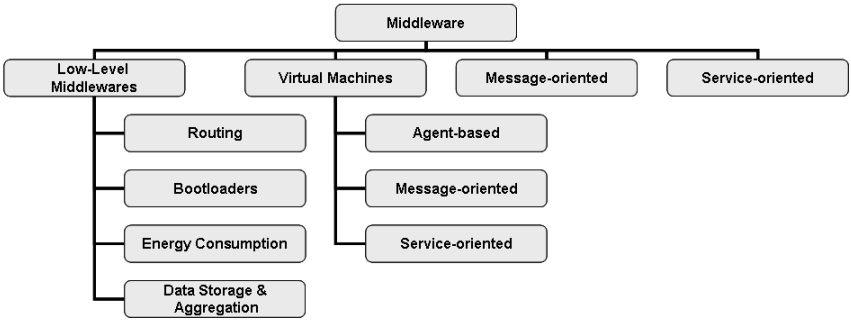


Fig. 1. Categorization of middleware systems.

easier and offer well-defined interfaces. They can be used in nearly any system, but they are very hardware specific. Since they are designed on a low level, the functionality of a higher middleware (e.g. abstraction of objects) never can be reached. Such middleware can also be the lowest tier of a bigger middleware system. Current projects of low-level middleware are for example *Deluge* [5], *CoCo/Mo* [6] and *MoteWorks* from Crossbow Inc.

Virtual Machines give the user the ability to replace code on the motes and therefore they are predestined for **agent-based** approaches. Nevertheless message- or service-oriented applications are also possible. The main benefit is the use of an interpreted language, so the code can be improved on a running system. This is a big difference to the low-level approach of the bootloader, where only compiled images can be updated. As mentioned before, an agent-based middleware takes most advantage of this behavior. This architecture is most applicable in homogeneous, but unstable networks with moving nodes, supporting more than one task. By the distribution of the agents, applications can move to the subnetworks where they are needed. *Maté* [7] and its TinyOS port *Bombilla*, developed at UC Berkeley, can be seen as the standard virtual machine for wireless sensor networks. *Agilla* [8] is the agent-based approach of a middleware built upon Bombilla and Deluge. It is a very active project with first application examples.

The classic **message-oriented** approach is also represented. Here the middleware provides interfaces for the underlying system and message structures. It also can help to do some automated actions like changing routing algorithms. This approach bears some problems within highly distributed or strongly collaborating networks. Here the vast amount of messages that need to be sent can lead quickly to communication blockings. Examples of message-oriented middleware are *Hood* [9] and *Dynamo* [10].

The last family consists of the **service-oriented** middleware. Here the middleware provides functions to publish and subscribe to services. Sensor readings, data storage and aggregation can then be implemented in services to which any mote in need can subscribe. This is a quite con-

servative way of a middleware which has formerly been developed for wired distributed systems. Therefore this approach is well known, but it does not take the whole advantages of a WSN. Static heterogeneous networks would be the best environments for such a middleware. The service-oriented paradigm is followed by *Spatial Views* [11], *TinyLime* [12], *TinyCubus* [13], and the *GREEN* middleware [14] of the RUNES project. Most commercial projects also use this strategy (*Octavex* by Octave Technology, *PLASMA* by IHP, and *eZeeNet* by Meshnetics).
There are also some hybrid middleware programs which belong to more than one class providing more functionality and flexibility.
Summarizing it can be said, that most of the work is done for research and educational reasons. Only a very very small part has reached a commercial state yet.

4 Design Challenges

To design and implement middleware services that fulfill all or most of the requirements stated in sec. 2 is still a lot of work – even when starting with one of the open available systems mentioned above. We sketch some concrete issues here that are crucial for architecture and design (see [3] for related aspects).

- **Partition control and role concept**: To support a flexible network architecture with dynamic task sharing, distributed self-configuring cluster mechanisms are necessary. Each node should be able to switch between several roles, depending on the current network configuration. Protocols for cluster forming have to developed that activate sleeping nodes, distribute the characteristics of the cluster, and perform "load balancing" in terms of task sharing.
- **Versioning of applications**: For dynamic updates and multiple applications, formal descriptions have to be specified representing the capabilities and status of an individual node.
- **Resource allocation and management**: As each application needs different resources, an automatic way of detecting and reserving available resources like processing power, energy, or memory has to found. In addition, the partitions/clusters must be able to coordinate each other, as they ultimately might use the same communication resources on their connections to gateways and other central units.
- **Monitoring**: For a precise monitoring, different levels of on-line and off-line surveillance have to be supported on each node. The forwarding of monitoring data must interfere the network in such an extent that the operation is completely different to the normal case. Moreover, the memory on the nodes is very limited, so the efficiency of the logging has to be very high. Both issues require a lot of theoretical work on protocols and operating system scheduling.

5 Conclusion

For leveraging the usage of wireless sensor networks in industrial automation and process control, the development of respective middleware

systems is absolutely necessary. But the requirements arising from this field call for much more abstract, flexible, and adaptive services than available today. The virtual machine model, followed by systems like Bombilla, is a very promising approach for application switching and role models. The challenges in designing industry-grade middleware as mentioned above still mean significant work in solving all conceptual and technical issues before a commercial roll-out of the WSN technology can commence.

Acknowledgements

The authors want to thank Christoph Niedermeier and his team at Siemens Corporate Technology for financial support and fruitful discussions leading to the results reported here.

References

1. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D.E., Pister, K.S.J.: System architecture directions for networked sensors. In: ASPLOS, Cambridge, MA (2000) 93–104
2. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. ACM Mobile Computing and Communication Review **6**(4) (2002) 59–61
3. Yu, Y., Krishnamachari, B., Prasanna, V.K.: Issues in designing middleware for wireless sensor networks. IEEE Network Magazine **18**(1) (2004) 15–21
4. Mascolo, C., Hailes, S., Lymberopoulos, L., Picco, G.P., Costa, P., Blair, G., Okanda, P., Sivaharan, T., Fritsche, W., Karl, M., Rónai, M.A., Fodor, K., Boulis, A.: Survey of middleware for networked embedded systems. Technical report, EU project 'RUNES', IST-004536-RUNES (2005)
5. Hui, J.W., Culler, D.: 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, ACM Press (2004) 81–94
6. Winter, R.: Coco/mo (communication cooperation/mobility). Technical report, FU Berlin (2006) http://www.inf.fu-berlin.de/inst/ag-tech/projects/SPP1140/index.htm.
7. Levis, P., Culler, D.: Maté: A tiny virtual machine for sensor networks. In: ASPLOS. (2002) 85–95
8. Fok, C.L., Roman, G.C., Lu, C.: Mobile agent middleware for sensor networks: An application case study. In: Proc. of the 4th Int. Conf. on Information Processing in Sensor Networks (IPSN'05), IEEE (2005) 382–387
9. Welsh, M., Mainland, G.: Programming sensor networks using abstract regions. In: Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI '04), San Francisco, CA, USA (2004) 29–42

10. Mohapatra, S.: DYNAMO: Power Aware Middleware for Distributed Mobile Computing. PhD thesis, Donald Bren School of Computer Sciences, University of California, Irvine (2006)
11. Ni, Y., Kremer, U., Iftode, L.: Spatial views: Space-aware programming for networks of embedded systems. In: 16th International Workshop on Languages and Compilers for Parallel Computing, TAMU, College Station (2003)
12. Curino, C., Giani, M., Giorgetta, M., Giusti, A., Murphy, A.L., Picco, G.P.: Tinylime: Bridging mobile and sensor networks through middleware. In: 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005), IEEE Computer Society (2005) 61–72
13. Marrón, P.J., Minder, D., Lachenmann, A., Rothermel, K.: Tinycubus: An adaptive cross-layer framework for sensor networks. Information Technology **47**(2) (2005) 87–97
14. Sivaharan, T., Blair, G.S., Coulson, G.: Green: A configurable and re-configurable publish-subscribe middleware for pervasive computing. In: OTM Conferences. (2005) 732–749

Middleware support for the "Internet of Things"*

Karl Aberer, Manfred Hauswirth, Ali Salehi

School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

Abstract. The "Internet of Things" intends to enhance items of our everyday life with information and/or processing and interconnect them, so that computers can sense, integrate, present, and react on all kinds of aspects of the physical world. As this implies that enormous numbers of data sources need to be connected and related to each other, flexible and dynamic middleware support essentially providing zero-programming deployment is a key requirement to cope with the sheer scale of this task and the heterogeneity of available technologies. In this paper we briefly overview our Global Sensor Networks (GSN) middleware which supports these tasks and offers a flexible, zero-programming deployment and integration infrastructure. GSN's central concept is the virtual sensor abstraction which enables the user to declaratively specify XML-based deployment descriptors in combination with the possibility to integrate sensor network data through plain SQL queries over local and remote sensor data sources. The GSN implementation is available from http://globalsn.sourceforge.net/.

1 Introduction

In the sensor network domain most of the ongoing work focuses on energy efficient routing, aggregation, and data management algorithms inside a single sensor network. The deployment, application development, and standardization aspects received little attention so far. However, as the price of sensors diminishes rapidly we can soon expect to see very large numbers of heterogeneous sensor devices and sensor networks being deployed to support the vision of the "Internet of Things" [1]. Major challenges in this "Sensor Internet" environment are (1) minimizing the development and deployment efforts which are key cost factor in large-scale systems and (2) the data-oriented integration of very large numbers of data sources. Despite the heterogeneity of the available platforms, their requirements for processing, storing, querying and publishing data, however, are similar and the main differences among various software platforms are the different abstractions for the sensors and sensor networks.

Our Global Sensor Networks (GSN) middleware addresses these problems and 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 (a minimal set of powerful abstractions which can be easily configured and adopted), adaptivity (adding new types of sensor networks and dynamic (re-) configuration of

* 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 and was (partly) carried out in the framework of the EPFL Center for Global Computing.

data sources has to be supported during run-time), scalability (peer-to-peer architecture), and light-weight implementation (small memory foot-print, low hardware and bandwidth requirements, web-based management tools).

In the following, we will give a concise overview of GSN and how it provides an infrastructure to support the "Internet of Things". Section 2 discusses the virtual sensor abstraction, Section 3 highlights GSN's data stream processing functionalities, and Section 4 describes its architecture. For a detailed description of these issues we refer the reader to [2]. Section 5 then discusses GSN's dynamic plug-and-play functionalities which rely on simple semantic descriptions based on the IEEE 1451 standard [3] to identify sensors and dynamically integrate them into GSN. This is a key enabling technology in GSN supporting fully dynamic and zero-programming integration of data sources in GSN.

2 Virtual sensors

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
- an SQL-based specification of the stream processing performed in a virtual sensor
- 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. Figure 1 shows a fragment of a virtual sensor definition which defines a sensor returning an averaged temperature from a remote virtual sensor (`wrapper="remote"`) which is accessed via the Internet through another GSN instance (GSN instances cooperate in a peer-to-peer fashion).

```
 ...
<life-cycle pool-size="10" />
<output-structure>
  <field name="TEMPERATURE" type="integer"/>
</output-structure>
<storage  permanent-storage="true" size="10s" />
<input-stream name="dummy" rate="100" >
  <stream-source alias="src1" sampling-rate="1" storage-size="1h">
     <address wrapper="remote">
       <predicate key="type" val="temperature" />
       <predicate key="location" val="bc143" />
     </address>
     <query>select avg(temperature) from WRAPPER</query>
   </stream-source>
  <query>select * from src1</query>
</input-stream>
 ...
```

Fig. 1. Virtual sensor definition (fragment)

To specify the processing of the input streams we use SQL queries which refer to the input streams by the reserved keyword `WRAPPER`. The `<input-stream>` element

provides all definitions required for identifying and processing an input stream of the virtual sensor. The `<life-cycle>` element defines deployment aspects such as the number of threads available for processing, the `<storage>` element controls how stream data is stored persistently (among other attributes this controls the temporal processing), and `<output-structure>` defines the structure of the produced output stream. A detailed description of virtual sensors is provided in [2].

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: (1) a local clock at each GSN container, (2) implicit management of a timestamp attribute, (3) implicit timestamping of tuples upon arrival at the GSN container (reception time), and (4) a windowing mechanism which allows the user to define count- or time-based windows on data streams. Multiple time attributes can be associated with data streams and can be manipulated through SQL queries. 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:

1. The new data stream element is timestamped using the local clock.
2. 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.
3. The input stream queries are evaluated and stored into temporary relations.
4. The output query for producing the output stream element is executed based on the temporary relations.
5. The result is permanently stored if required and all consumers of the virtual sensor are notified of the new stream element.

GSN's query processing approach is related to 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 with the full range of operations allowed by the standard syntax.

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 which enables on-demand use and combination. Virtual sensor descriptions hold 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, i.e., this provides a simple model of identification and discovery of virtual sensors through metadata. GSN nodes communicate among each other in a peer-to-peer fashion. Figure 2 depicts the internal architecture of a GSN node.

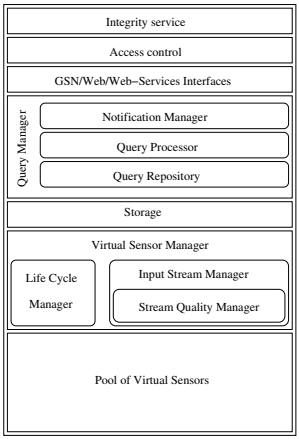


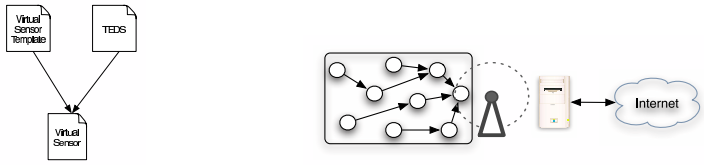
Fig. 2. GSN container architecture

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 done 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 query repository manages all registered queries (subscriptions) and defines and maintains the set of currently active queries for the query processor. The notification manager deals with the delivery of events and query results to the registered clients. The notification manager has an extensible architecture which allows the user to customize it to any required notification channel. The top three layers deal with access mechanisms, access control, and integrity and security.

5 Zero-programming deployment

As described before, GSN reduces deployment to writing an XML file describing the sensor-dependent properties. While having a single XML file (30-50 lines) describing a virtual sensor is much simpler than low-level device-dependent programming, human intervention is still required. To overcome this step, GSN uses the IEEE 1451 standard [3] for automatic detection, configuration and calibration. An IEEE 1451-compliant sensor provides a Transducer Electronic Data Sheet (TEDS) which is stored inside the sensor. The TEDS provides a simple semantic description of the sensor, i.e., it describes the sensor's properties and measurement characteristics such as type of measurement, scaling, and calibration information. A large number of sensors is already compliant to IEEE 1451 and this number is growing steadily.

To support truly zero-programming, GSN uses the TEDS self-description feature for the dynamic generation of virtual sensor descriptions by using a virtual sensor description template and deriving the sensor-dependent fields of the template from the data extracted from the TEDS as shown in Figure 3(a).



(a) Automatic generation (b) Setup

Fig. 3. Zero-programming deployment

The node setup to support plug-and-play deployment is shown in Figure 3(b). At least one base station capable of interacting with sensing devices (e.g., mica2, mica2dot, BTnode) is needed. When a sensor node enters a detection area (depicted by an antenna in the figure), GSN detects it, requests its TEDS, and dynamically instantiates a new virtual sensor based on the TEDS. This means that by using TEDS GSN can detect, identify, and automatically deploy sensors without human intervention. The generated virtual sensor description is immediately included into the detecting GSN node's repository and all processing dependent on the new sensor is executed. This is done on-the-fly while GSN is running. At the moment TEDS provides only that information on a sensor which enables interaction with it. Thus for some parts of the generated virtual sensor description, e.g., security requirements, storage and resource management, etc., we use default values.

The deployment of a sensor (network) in GSN implies making the produced data accessible through the Internet via web services and a web interfaces. Thus also all remote processing dependencies at any other node in the network of GSN nodes are triggered. This enables truly global integration of mobile sensors. GSN periodically asks for TEDS from all sensors to ensure that they are alive. If no response is provided by a sensor GSN removes the previously created virtual sensor, frees the associated resources and notifies dependent query and nodes.

6 Conclusions

GSN provides a flexible middleware for deployment of sensor networks meeting the challenges that arise in real-world environments. Its plug-and-play deployment functionality makes it specifically interesting as an infrastructure for the "Internet of Things," enabling completely new types of applications. For example, users or things could be equipped with RFID tags not only holding static data but also specifying data-processing tasks (queries), which GSN can recognize and include into its stream processing. This provides data processing dynamicity besides physical mobility without additional efforts for deployment. The GSN implementation is available from http://globalsn.sourceforge.net/ and a detailed description of GSN in provided in [2].

References

1. Gershenfeld, N., Krikorian, R., Cohen, D.: The Internet of Things. Scientific American (2004)
2. Aberer, K., Hauswirth, M., Salehi, A.: The Global Sensor Networks middleware for efficient and flexible deployment and interconnection of sensor networks. Technical Report LSIR-REPORT-2006-006, Ecole Polytechnique Fédérale de Lausanne (2006)
3. NIST: IEEE1451 (2006) http://ieee1451.nist.gov/.

Genetic Programming Techniques for Sensor Networks

Thomas Weise, Kurt Geihs

University of Kassel, Wilhelmshöher Allee 73, Kassel, Germany
{weise|geihs}@vs.uni-kassel.de

Abstract. In this paper we present an approach to automated program code generation for sensor nodes and other small devices. Using Genetic Programming, we are able to discover algorithms that solve certain problems. Furthermore, non-functional properties like code size, memory usage, and communication frequency can be optimized using multiobjective search techniques. The evolution of algorithms requires program testing, which we perform using a customized simulation environment for sensor networks. The simulation model takes into account characteristic features of sensor nodes, such as unreliable communication and resource constraints. An application example is presented that demonstrates the feasibility of our approach and its potential to create robust and adaptive code for sensor network applications.

1 Introduction

Today we experience a growing demand for distributed systems consisting of sensor nodes [1]. As explained in [2], such devices are restricted in resources like memory size, processing speed, and battery power. The communication among them is costly in terms of energy and not reliable either. Furthermore, the topology of sensor networks is volatile and usually cannot be determined a priori, enforcing self organization. Algorithms and protocols normally applied to distributed systems are therefore often insufficient and need to be replaced with specialized counterparts.

These requirements make programming sensor nodes a demanding task for software developers because the design of sensor network applications must pay attention to aspects that are not directly related to the application functionality itself. We claim that Genetic Programming techniques are an effective means to automatically discover powerful programming solutions for sensor networks. Clearly, Genetic Programming is not suited to produce very large programs for general application tasks. However, we view sensor nodes as ideal targets for Genetic Programming since these nodes can only perform a limited functionality due to their resource constraints.

In this paper we introduce DGPF, the Distributed Genetic Programming Framework [3], which allows us to automatically discover distributed algorithms for given problems. Additionally, such algorithms can be optimized in various ways, taking energy consumption into account as well as memory usage or code size. In the next section we describe how Genetic Programming can be applied to sensor networks followed by an example in the third section. At the end of the paper, a short

summary of related work is given along with future work and a conclusion.

2 Genetic Programming and Sensor Networks

For a long time, Genetic Algorithms have been used in science to derive solutions for any type of problems, from construction of wind turbines to pattern-recognition systems. The application of Genetic Algorithms with the goal to evolve computer programs is called Genetic Programming. This section will give a brief overview on how Genetic Algorithms work in common and how they can be applied to sensor networks.

2.1 Genetic Algorithms

As shown in Fig. 1, Genetic Algorithms start with an initial population of random solution candidates called individuals. In our case, the individuals are small programs that can be executed on sensor nodes. As in nature, the population will be refined step by step in a cycle of computing the fitness of its individuals, selecting the best individuals and creating a new generation derived from these. If a reasonable good solution has evolved, the algorithm will terminate.

Randomized optimization algorithms are called "multiobjective" if they permit the specification of more than one optimization objective. Using our DGPF, several fitness functions can be defined, allowing us to optimize programs not only for functionality but also for nonfunctional requirements like energy consumption and communication frequency. Furthermore, different search algorithms like randomized Hill Climbing and Genetic Algorithms can be combined to speed up the optimization process.

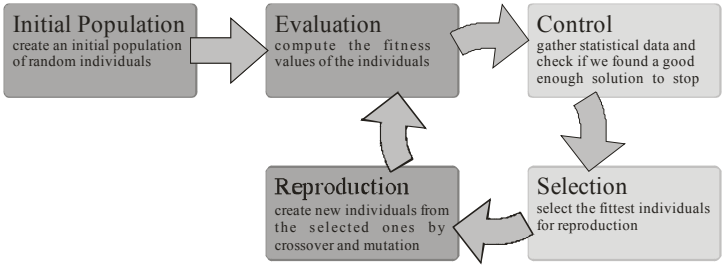


Fig. 1. Common cycle of Genetic Algorithms.

In our case the individuals that are being evolved are algorithms in the form of small programs. The functionality and effectiveness of such an algorithm can be determined by simulating it on a virtual hardware representing a single sensor node. In terms of distributed algorithms, it is not sufficient to simulate only one sensor node. Thus multiple instances of the program will be executed in a network simulator in parallel. The following two subsections describe the virtual hardware and the network model used in our simulations.

2.2 Virtual Hardware

A sensor node is modeled as an automaton that consists of a virtual hardware holding its execution state and a program running on that hardware. Unlike most other approaches in Genetic Programming which grow stateless functions, we have developed an architecture with a fixed-sized memory. The instruction set can be reduced to the one introduced by Teller [4], granting the Turing completeness needed to model real distributed algorithms or network protocols. Like real microprocessors, direct and indirect memory access and a collection of arithmetic expressions are included in the instruction set as well as conditional jumps. Communication is also modeled with primitive directives which allow storing memory words in an output buffer and transmitting the buffered data. A single message can be received into the input buffer and will be processed by reading the received words sequentially. Newly incoming messages get lost if the input buffer is already occupied.
The example in the next section displays some of the available instructions (see Fig. 2).

2.3 Simulation

The network simulation provides additional statistical data for each automaton, holding information on the number of messages sent, lost, and successfully processed.

As in real networks, many automata run asynchronously at approximately the same speed which, however, might differ from node to node and cannot be regarded as constant either. To grant realistic simulations, the network model has the following properties:

1. The links between the nodes are randomly created, yet it will be ensured that there are no network partitions.
2. Messages are simple word sequences with no predefined structure.
3. Messages cannot be sent directly. Like radio broadcasts they will be received by any node in transmission distance. Finding out which message is of concern will be in the responsibility of each node.
4. Messages can get lost without special cause.
5. Transmissions may take a random time until they reach their target.
6. The collision of two transmissions underway leads to the loss of both messages.

3 Example Algorithm

One example problem that we have solved with the DGPF is the so-called election problem. This problem is well known in the area of distributed computing and therefore we have used it to validate our approach. Election means to select one node out of a group of nodes, for instance to act as a communication relay.

Each node owns a unique identifier and all nodes must know the ID of the elected node after the election algorithm has terminated. One method to perform such an

election would be to select the node with the maximum ID. We therefore introduce a functional fitness function that evaluates how many nodes know the maximum ID after a given amount of time. Additionally, we enter three non-functional fitness functions into the evolution: parsimony pressures for minimum code size, minimum memory size and a minimum transmission count. Each automaton is initialized with its own ID in its memory cell.

It took several hours to obtain the solution depicted in Fig. 2 using an older version of our framework on a 3 GHz Pentium 4 PC. Most time was spent on the network simulation. Due to many optimizations, the current version of the DGPF runs significantly faster and also incorporates various new distribution schemes [3] for Genetic Algorithms resulting in further speedup, depending on the number of available computers.

The discovered algorithm seems to be simple and quite efficient for the problem definition: The nodes initially send the contents of their first memory cell (their ID) to all neighbors. If and only if a node then receives a greater ID, it stores it in the first memory cell and starts the cycle again. Otherwise, no message is sent which pays respect to the parsimony pressure for minimum transmission count. The node waits instead for incoming transmissions.

```
@0:
  Send mem[0]
@1:
  mem[1]=Receive
  If(mem[1]<=mem[0]) Goto @1
@2:
  mem[0]=mem[1]
  Goto @0
```

Fig. 2. The genetically evolved election algorithm.

4 Related Work

Although Genetic Programming was invented almost 20 years ago by Koza [5] , it is a novel idea to employ it for finding distributed algorithms, in particular for sensor networks. In protocol engineering, Genetic Protocol design has already proven to be a useful and promising technology in several independent research projects [6], [7], [8]. In contrast to these approaches, our framework is not just bound to communication protocol synthesis but is able to evolve functional programs together with a dedicated protocol.

The idea of multiobjective optimization using Genetic Algorithms reaches back to the 1960s but is generally contributed to Schaffner. His VEGA-algorithm [9], although not very efficient, was the first real evolutionary algorithm which considered multiple optimization criteria in a non-trivial manner. A few years later, Zitzler et. al. showed that their advanced multiobjective evolutionary algorithm SPEA2 outperforms singleobjective GA in code size reduction by far [10]. Extending this idea, we optimize not only for code size, but also for other non-functional criteria.

5 Future Work and Conclusion

In the near future, a graphical user interface will be provided to ease the work with the DGP Framework. The quality of the results and the speed of the evolutionary process will be increased by integrating additional optimization techniques. In terms of sensor network programming, we will implement a port to the MSP430 instruction set. This will allow us to test the evolved programs on real sensor nodes like the ESBs of the ScatterWeb platform [11].

In this paper we have presented a method and a framework for automated creation of efficient algorithms for sensor networks based on Genetic Programming. We have shown the viability of the approach for a simple example. Our new framework and all results are provided as open source to the research community under the LGPL license. More information on our research as well as the fully documented Java source code of the DGPF can be found at http://dgpf.sourceforge.net.

6 References

[1] Jürgen Bohn, Vlad Coroama, Marc Langheinrich, Friedemann Mattern, Michael Rohs: "Disappearing Computers Everywhere - Living in a World of Smart Everyday Objects", New Media, Technology and Everyday Life in Europe Conference, 2003, London, UK

[2] Holger Karl, Andreas Willig: "A Short Survey Of Wireless Sensor Networks", Technical Report, Telecommunication Networks Group, Technische Universität Berlin, 2003

[3] Thomas Weise, Kurt Geihs: "DGPF - An Adaptable Framework for Distributed Multi-Objective Search Algorithms Applied to the Genetic Programming of Sensor Networks", submitted to BIOMA 2006, The 2nd International Conference on Bioinspired Optimization Methods and their Applications 9 - 10 October 2006, Ljubljana, Slovenia

[4] Astro Teller, "Turing completeness in the language of genetic programming with indexed memory", Proceedings of the 1994 {IEEE} World Congress on Computational Intelligence Volume 1, IEEE Press, 1994

[5] John R Koza: "Non-Linear Genetic Algorithms for Solving Problems". United States Patent 4,935,877. Filed May 20, 1988. Issued June 19, 1990.

[6] Khaled El-Fakihy, Hirozumi Yamaguchiz, Gregor v. Bochmann: "A Method and a Genetic Algorithm for Deriving Protocols for Distributed Applications with Minimum Communication Cost", Proceedings of Eleventh IASTED International Conference on Parallel and Distributed Computing and Systems, November 3-6, 1999, Boston, USA

[7] Lidia Yamamoto, Christian Tschudin, "Genetic Evolution of Protocol Implementations and Configurations", IFIP/IEEE International workshop on Self-Managed Systems and Services (SelfMan 2005), Nice, France

[8] F. Comellas, G. Gimenez: "Genetic Programming to Design Communication Algorithms for Parallel Architectures", in Parallel Processing Letters, 1998

[9] J. David Schaffner: "Multiple objective optimization with vector evaluated genetic algorithms", in Genetic Algorithms and their Applications: Proceedings of the First International Conference on Genetic Algorithms, pp.31-100, Lawrence Erlbaum, 1985

[10] Bleuler, S., Brack, M., Thiele, L. Zitzler, E.: "Multiobjecive Genetic Programming: Reducing Bloat using SPEA2", In Proceedings of the 2001 Congress on Evolutionary Computation CEC2001. IEEE Press (2001) 536-543

[11] ScatterWeb, hard- and software platform for sensor networks, including Embedded Sensor Board with TI MSP430 controller, see http://www.inf.fu-berlin.de/inst/ag-tech/scatterweb_net/

Efficient Flash-based Virtual Memory for Sensor Networks

Andreas Lachenmann, Pedro José Marrón, Kurt Rothermel

Universität Stuttgart, Germany
{lachenmann,marron,rothermel}@ipvs.uni-stuttgart.de

Abstract. In this paper we present a virtual memory system for sensor networks that uses flash-memory to extend the amount of RAM available on each node. By analyzing access traces from simulation tools it creates an efficient memory layout that makes virtual memory viable despite the resource constraints typical for sensor networks.

1 Introduction

Most sensor nodes are equipped with just a few kilobytes of RAM. Therefore, main memory is usually a very scarce resource when developing sensor network applications. In fact, several applications already require more memory than available on current sensor nodes. For instance, TinyDB [1] requires the user to select at compile-time which functionality to include in the code image.

As applications for sensor networks increase in complexity, RAM limitations will continue to cause difficulties for developers. In traditional computing systems, virtual memory [2, 3] has been widely used to address this problem. With virtual memory, parts of the contents of RAM are written to secondary storage when they are not needed. This mechanism is easy to use, since the system takes care of managing the memory pages. However, current operating systems for sensor networks (e.g., [4, 5]) do not include support virtual memory.

Sensor nodes are equipped with flash memory as secondary storage, which is much larger than main memory (between 512 kB and 1 MB). It is organized in pages of several hundred bytes that have to be written *en bloc*. Accessing it is much more expensive than accessing RAM: it takes several milliseconds to read or write a flash page whereas variables in RAM can be accessed in a few processor cycles. In addition, accesses to flash memory are comparatively energy expensive. Nevertheless, this type of memory is appropriate for the implementation of virtual memory on sensor nodes.

In this paper we present *ViMem*, a virtual memory system for TinyOS-based sensor networks that uses flash memory to extend the size of RAM available to the application. Since energy is a limited resource in sensor networks, *ViMem* tries to minimize the number of flash memory operations. It uses variable access traces obtained from simulations to rearrange variables in virtual memory at compile-time so that only a small number of flash accesses is necessary.

2 Design Overview

ViMem consists of two main parts: a compiler extension and a runtime component, which are described in this section.

Application developers should be able to use variables in virtual memory just like those in RAM. However, since sensor network hardware does not directly support virtual memory, all access to data in virtual memory must be redirected to *ViMem*'s runtime component. Our system accomplishes this by using a pre-compiler that modifies all such variable accesses. This pre-compiler changes source code written in nesC [6], the programming language used by TinyOS [4]. We have selected nesC and TinyOS because of their active research community that has developed a large number of application and system components.

The developer maintains full control of which variables are kept in RAM and which ones are stored in virtual memory. Only those tagged with a special attribute are put into virtual memory. This way, variables that are used in interrupt handlers and other performance-critical functions can always be kept in RAM. The pre-compiler executes the memory layout algorithm described in Section 3 to place variables on pages in virtual memory.

ViMem's runtime system is responsible for the management of memory pages kept in RAM and for the provisioning of data to the application. The challenge here is to determine which memory page has to be replaced when another one is loaded from flash memory. Therefore, the algorithm has to predict which pages are most likely used again in the future. In addition, it has to consider the costs for replacing a page (writing modified pages is more expensive than just reading).

This algorithm was not the main focus of our research. Therefore, for *ViMem*'s replacement policy we have adapted the Enhanced Second-Chance Algorithm [3], which approximates a least-recently used (LRU) page replacement strategy.

3 Memory Layout Algorithm

This section describes our memory layout algorithm that determines the placement of variables in virtual memory. It is the core part of our approach to reduce the number of flash accesses and, thus, improve on efficiency. The algorithm has two main goals: First, it aims to reduce the overall number of page replacements. Second, it puts special effort in decreasing the number of write accesses to flash memory.

3.1 Use of Variable Access Traces

In general, finding an optimal memory layout is not possible since the exact order in which variables are accessed at runtime depends on many factors. For example, in sensor networks data gathering requests from users as well as sensory input and packets received from other nodes may influence the application flow. Therefore, our memory layout algorithm can only provide a heuristic that does not necessarily find the best solution for each execution path.

Although the specific order of data accesses is not predictable, there are typically patterns that recur. Our algorithm uses simulation traces to determine such patterns for variables stored in virtual memory. Gathering information about variable accesses using simulation does not introduce any overhead at runtime and, thus, does not alter the behavior of the application itself.

If no simulation data is available (e.g., when building a new application), the *ViMem* pre-compiler uses the variable references in the source code to estimate the number of accesses. Obviously, this information can be inaccurate because it is unclear how often a function is called or which branch of an if-statement is selected at runtime, for example.

The pre-compiler splits up complex variables, such as arrays and structs, and examines each part individually. For example, the first elements of an array might be accessed more frequently than the last ones. Therefore, instead of recording the access just for complex variables as a whole, all data accesses are associated with individual data elements. We define as such a data element an atomic part of a complex variable with a simple data type like "int".

3.2 Grouping of Data Elements

Having gathered information about accesses to data elements, the memory layout algorithm forms groups of those that are often accessed together. When reading an access trace the pre-compiler calculates the weights of a fully-connected graph $G = (V, E, f, g)$, where the nodes V are the data elements and the edges E represent the relationship between the data elements. In this graph both the nodes and edges are weighted: The weight of a node, given by $f : V \to \mathbb{R}$, indicates how often the corresponding data element has been accessed, and the weight of an edge, defined by $g : E \to \mathbb{R}$, gives information about the proximity of two data elements.

For each sensor node in the network, the pre-compiler maintains an ordered list of data elements that have been accessed recently. Each data element appears in this list at most once with only its most recent access being present. The sum of the sizes of all these elements may not exceed the size of a flash page. Thus, these elements represent those that should be preferably in RAM when the new element is accessed. When the *ViMem* pre-compiler adds a data access from the trace, it increments both the access count in the data element's node and the proximity value of all data elements accessed previously.

Fig. 1 shows an example of how an access trace is processed. The figure displays parts of an access trace for one node, the graph G, and the list of recently accessed elements. For simplicity, it assumes that the size of a memory page is just 8 bytes. The figure shows the simple variables "a", "b", and "c" as well as struct "s". As described above, the algorithm splits up the parts of the struct and examines each field individually. In the example the last line of the access trace has been processed, which leads to the following changes. First, the element is added to the list of recently accessed data elements (I). Since the total size of the elements in this list is greater than the page size, the algorithm

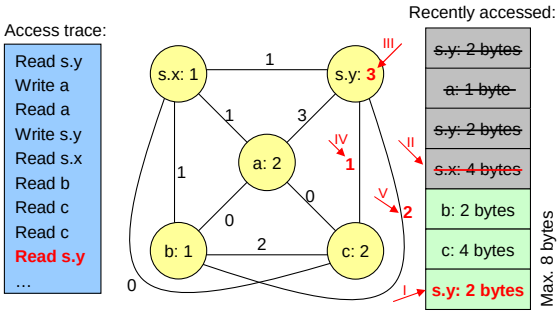


Fig. 1. Example for processing an access trace

removes the oldest element ("s.x", II). Then it increments the weights of "s.y" (III) and of all its edges to elements in the list (IV and V).

After reading the complete access trace, *ViMem*'s pre-compiler traverses the graph G using a breadth-first search that takes into account the proximity values. It begins with one of the nodes from which the edge with the maximum proximity value starts. Being at node v, the search follows an edge e only if $g(e)/f(v) > k$, where k is a threshold that indicates which data elements should be definitely put on one page of flash memory. The search is aborted if the elements reached no longer fit on a single page. All data elements that are reached using this search mechanism are grouped in order to be placed on the same memory page.

3.3 Data Placement

After determining the groups of data elements used together, the memory layout algorithm places them on actual memory pages. This part of the algorithm processes the elements in the order of their access frequencies and places them with a first-fit strategy. This way data elements that are accessed often are placed on the same memory page, which can probably stay in RAM for most of the time.

The algorithm uses two sets of pages: one with elements that are mostly read and one with those that are modified more often. If a "mostly-read" page has to be removed from RAM, this approach makes it more likely that it does not have to be written back to flash memory.

4 Simulation Results

We modified TinyDB to make use of *ViMem* and obtained simulation results with the Mica2 simulator Avrora [7]. In Fig. 2 we show the number of page faults, i.e., the number of read accesses from secondary storage. In many cases *ViMem*'s memory layout algorithm greatly reduces the number of accesses to flash memory. Using only the data references from the source code it is already able to decrease the percentage of accesses leading to page faults by more than 60%. Nevertheless, memory layouts that have been optimized for a given scenario can reduce the percentage of page faults even further by additional 80%.

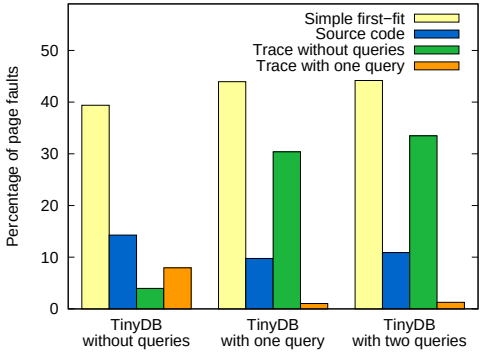


Fig. 2. Number of page faults

5 Summary

In this paper we have described *ViMem*, our virtual memory system for TinyOS-based sensor nodes. It uses a compile-time approach to create an efficient memory layout based on data access traces obtained from simulation. We have presented results that show that this algorithm reduces the overhead of virtual memory significantly. The remaining overhead does not hinder the implementation of complex applications for sensor networks. Therefore, using *ViMem* the memory limitations of sensor nodes are not as strict as before.

References

1. Madden, S., Franklin, M.J., Hellerstein, J.M., Hong, W.: The design of an acquisitional query processor for sensor networks. In: Proc. of the Int'l Conf. on Management of Data. (2003) 491–502
2. Denning, P.J.: Virtual memory. ACM Comput. Surv. **2**(3) (1970) 153–189
3. Silberschatz, A., Galvin, P.B., Gagne, G.: Operating System Concepts. 6th edn. John Wiley & Sons (2002)
4. Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., Pister, K.: System architecture directions for networked sensors. In: Proc. of the 9th Intl. Conf. on Architectural Support for Programming Languages and Operating Systems. (2000) 93–104
5. Dunkels, A., Grönvall, B., Voigt, T.: Contiki – a lightweight and flexible operating system for tiny networked sensors. In: Proc. of the First Workshop on Embedded Networked Sensors. (2004)
6. Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., Culler, D.: The nesC language: A holistic approach to networked embedded systems. In: Proc. of the Conf. on Programming Language Design and Implementation. (2003) 1–11
7. Titzer, B., Lee, D., Palsberg, J.: Avrora: Scalable sensor network simulation with precise timing. In: Proc. of the Fourth Int'l Conf. on Information Processing in Sensor Networks. (2005) 477–482

When Modularity Matters

Olaf Landsiedel, Klaus Wehrle

Distributed Systems Group
RWTH Aachen, Germany
firstname.lastname@rwth-aachen.de

Abstract. In an attempt to employ sensor network technology for animal observation, in particular of wild rats, we identified several restrictive shortcomings in existing sensor network research. In this paper, we present modular and flexible communication protocols as an efficient substrate to address these shortcomings. Their modularity allows recomposing a protocol dynamically at runtime and adapting it to the changing needs of a deployed sensor network.

1 Introduction

One of the core motivations for the research in sensor networks is the vision of deploying sensor networks in nature to observe environmental phenomena. Typically, environmental phenomena are observed to study them. The knowledge about them is limited and the researcher hopes to reveal some unknowns, i.e. during the observation the researcher learns about the specifics of the phenomena. Thus, during the deployment the measurement and data aggregation schemes of the sensor nodes need to be adapted to changing application needs.

However, once the sensor network is deployed it is generally very challenging to regain physical access to the sensor nodes and to reprogram these according to the changed application needs. For example, we are currently equipping rats with standard sensor nodes (mica2dot) and a sensor suite consisting of light, audio, and acceleration sensors. A sensor is attached to a lab rat via a special leather jacket, which has a pocket fitted for this equipment. This jacket has openings for the front legs and wraps around the rib cage and the back. Rats live in underground burrows. Thus, once these sensor nodes are deployed on the rats regaining access involves a major digging afford.

In this paper we discuss the case of reconfigurable communication protocols for sensor networks and our ongoing work in this area. Furthermore, we argue that other components of the sensor network operating system do not require such an amount of flexibility. Thus, compared to the SOS [3] operating system we do not propose a modular scheduler or memory management.

The remainder is structured as follows: Section 2 discusses the case of modular communication protocols. Next, section 3 introduces our fine grained modules for protocol building. Section 4 discusses related work and section 5 concludes.

2 The Case for Modular Protocols in Sensor Networks

In this section we motivate the use of modular and reconfigurable communication protocols in sensor networks, see fig. 1. Commonly, sensor network deployment and main-

tenance consists of several steps: (1) The sensor network is deployed: Via flooding a sensor node determines its position in the network and announces its existence to the surrounding nodes. (2) Based on its position in the network, various tasks are assigned to a node: while data collection and forwarding are commonly assigned to a huge number of nodes, selected nodes take care of data aggregation [4, 6] or act as routing beacons [1, 2]. (3) During deployment the conditions change. Due to node failure or other environmental influences – such as changing radio propagation – nodes take over tasks from other nodes. Furthermore, nodes are re-tasked based on data sampled in previous measurements to adapt their functionality to new upcoming needs, as discussed in section 1.

Today's sensor node operating systems [5] and their applications are statically linked at compile time. This approach allows to use code optimization and resource facilitation analysis. However, all functionality that might be used during deployment needs to be compiled into the binary at compile time. Furthermore, updates while a sensor network is deployed become very costly, as a whole binary needs to be redistributed. Thus, modular communication protocols can be of high benefit for sensor networks.

3 The Modules

In this section we discuss the modules that are used to compose a communication protocol. When analyzing various communication protocols we identified the following key properties: (1) A module shall present protocol independent functionality, for example to set certain bytes in a packet. (2) A configuration string at runtime or compile time specifies the exact functionality, making a component protocol dependent. (3) All component interfaces (in- and out-ports) are standardized to ensure that components can be combined arbitrarily.

Based on these above described properties, we designed our modular communication protocols. The components can be grouped into five main groups: Source, sink, operational, validation and de-multiplex. In the source group are all components that emit packets into the protocol, i.e. the incoming network interface, the application, and timers which omit packets at certain intervals. Similar, the sink class represents outgoing network interfaces, the application and packet droppers. The operational components change the packet's header, payload or additional options like the outgoing device. Validation components check certain parts of a packet, based on the result they emit it to one of their outgoing ports. This class of components splits a packet flow into multiple flows. The de-multiplex component merges flows.

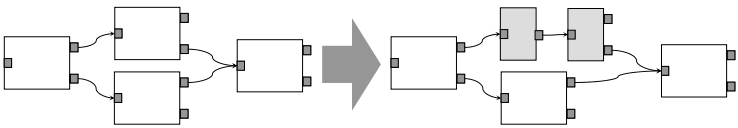


Fig. 1: Modular and reconfigurable protocols allow for dynamic changes.

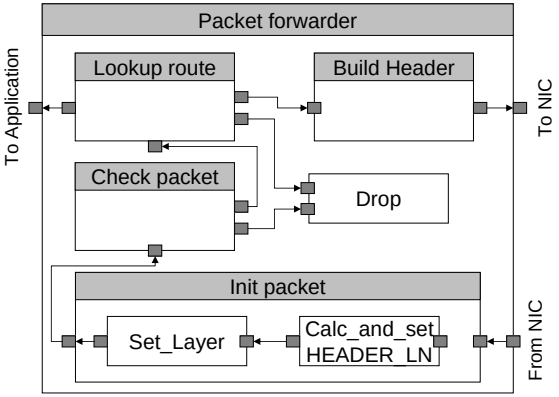


Fig. 2: A packet forwarder build from compound and simple components

From these component classes we derived the individual components. Thus, each component has dedicated functionality which ranges from setting bits or bytes and computing a checksum to storing the system state in a so called blackboard.

Compound modules are used to group components into functional and semantic groups. Compound modules are either protocol dependent or protocol independent. A protocol independent compound for example is a loop, which parses a packet for a certain bit or byte pattern. In the case of the IP protocol it can be used to parse for IP-options. Protocol dependent compounds are used to group functionality and make the protocol description more readable. A typical compound is a compound which builds a protocol header, see fig. 2.

To allow for easy component and protocol development, we have implemented a compiler and GUI which derive from a meta language the right modules to use and concatenate and configure these accordingly. Space limitations prevent us from discussing these features in more detail.

The work discussed in this paper is ongoing work. As today's sensor network protocols are kept quite simple, we derived our modules from the more complex Internet protocols, e.g. IP and TCP. Currently we use the modules derived from the Internet protocols to build sensor network protocols ranging from tree-based routing to data aggregation.

4 Related Work

In this section we discuss the existing works on modular communication protocols and compare our work to them. Modular protocols have been previously presented for the use in the internet. Click [8] is a modular software router. However, most of the Click modules present IP specific functionality. In our approach modules are protocol independent, a configuration at run- or compile-time makes their behavior protocol specific. As result, our modules can be reused for various protocols. KIDS [9] provides a mod-

ular QoS system, similar to Click it focuses on a certain protocol type, in this case on QoS functionality.

The Sensor Operating System (SOS) [3] and FlexCup [7] introduce a modular operating system for sensor nodes. It allows to change major parts of the OS dynamically at run-time. However, from our point of view, OS components like the scheduler do not need to be changed at runtime. Thus, we propose a more lightweight approach. Furthermore, as our approach focuses on protocols only, it provides a more fine grained approach.

5 Conclusion

In this paper we presented our ongoing work on modular communication protocols for sensor networks. We introduced our fine grained approach to protocol independent modules which makes it applicable to wide range of different communication protocols.

Next to the implementation of various sensor network protocols our ongoing implementation efforts focus on two topics. First, it might be interesting to implement the modules on various platforms and even add a platform abstraction layer. Thus, the modules can be run on various systems. As a result, one can test a protocol by using modules implemented for a simulator and then use the same already evaluated and tested configuration for a sensor network. Furthermore, we consider it highly interesting to evaluate how protocol verification techniques can be applied to the meta language describing component configuration and concatenation.

References

1. Q. Cao and T. Abdelzaher. A scalable logical coordinates framework for routing in wireless sensor networks. In *Proc. of Real-Time Systems Symposium (RTSS)*, 2004.
2. R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica. Beacon Vector Routing: Scalable Point-to-Point in Wireless Sensornets. In *Proc. of Networked Systems Design and Implementation (NSDI)*, 2005.
3. C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proc. of International Conference on Mobile Systems, Applications, and Services (MobiSys)*, June 2005.
4. C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proc. of Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2000.
5. P. Levis et al. The Emergence of Networking Abstractions and Techniques in TinyOS. In *Proc. of Networked Systems Design and Implementation (NSDI)*, March 2004.
6. S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TAG: a Tiny AGgregation Service for Ad-Hoc Sensor Networks. In *Proc. of Operating System Design and Implementation (OSDI)*, Dec. 2002.
7. 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 *Proc. of the Third European Workshop on Wireless Sensor Networks (EWSN 2006)*, 2006.
8. R. Morris, E. Kohler, J. Jannotti, and M. F. Kaashoek. The Click modular router. In *Proc. of Symposium on Operating Systems Principles (SOSP)*, 1999.
9. K. Wehrle. An Open Architecture for Evaluating Arbitrary Quality of Service Mechanisms in Software Routers. In *Proc. of International Conference on Networking (ICN)*, 2001.

Towards Predictable Wireless Sensor Networks – The Sensor Network Calculus

Jens B. Schmitt

disco | Distributed Computer Systems Lab, University of Kaiserslautern, Germany

Abstract. At the current state of affairs it is hard to obtain a predictable performance from wireless sensor networks, not to mention performance guarantees. In particular, a widely accepted and established methodology for modeling the performance of wireless sensor networks is missing. In the last two years we have tried to make a step into the direction of an analytical framework for the performance modeling of wireless sensor networks based on the theory of network calculus, which we customized towards a so-called *sensor network calculus* [1]. We believe the sensor network calculus to be especially useful for applications which have certain timing requirements. Examples for this class of applications are factory control, nuclear power plant control, medical applications, and any alerting systems. In general whenever the sensed input may necessitate immediate actions the sensor network calculus may be the way to go. In this paper we summarize these activities and discuss the open issues for such a an analytical framework to be widely accepted.

1 Introduction

Decisions in daily life are based on the accuracy and availability of information. Sensor networks can significantly improve the quality of information as well as the ways of gathering it. For example, sensor networks can help to get higher fidelity information, acquire information in real time, get hard-to-obtain information, and reduce the cost of obtaining information. Application areas for sensor networks might be production surveillance, traffic management, medical care, or military applications. In these areas it is crucial to ensure that the sensor network is functioning even in a worst case scenario. If a sensor network is used for example for production surveillance, it must be ensured that messages indicating a dangerous condition are not dropped. If functionality in worst case scenarios cannot be proven, people might be in danger and the production system might not be certified by authorities.

As it may be difficult or even impossible to produce the worst case in a real world scenario or in a simulation in a controlled fashion, an analytical framework is desirable that allows a worst case analysis in sensor networks. Network calculus [2] is a relatively new tool that allows worst case analysis of packet-switched communication networks. In [1] a framework for worst case analysis of wireless sensor networks based on network calculus is presented and called *sensor network calculus*. This framework has further been extended towards random

deployments [3] and the case of multiple sinks in [4]. The goal of this paper is to summarize these activities and show the usefulness of the sensor network calculus as well as opportunities for future work along this avenue.

2 Sensor Network Calculus: A Brief Walk-Through

In this section we use the notation and the basic results provided in [1], furthermore a single sink communication pattern is assumed. It is assumed that the routing protocol being used forms a tree in the sensor network. Hence N sensor nodes arranged in a directed acyclic graph are given.

Each sensor node i senses its environment and thus is exposed to an input function R_i corresponding to its sensed input traffic. If sensor node i is not a leaf node of the tree then it also receives sensed data from all of its child nodes $child(i,1),\ldots,child(i,n_i)$, where n_i is the number of child nodes of sensor node i. Sensor node i forwards/processes its input which results in an output function R_i^* from node i towards its parent node.

Now the basic network calculus components, arrival and service curve, have to be incorporated. First the arrival curve $\bar{\alpha}_i$ of each sensor node in the field has to be derived. The input of each sensor node in the field, taking into account its sensed input and its childrens' input, is:

$$\bar{R}_i = R_i + \sum_{j=1}^{n_i} R^*_{child(i,j)} \tag{1}$$

Thus, the arrival curve for the total input function for sensor node i is:

$$\bar{\alpha}_i = \alpha_i + \sum_{j=1}^{n_i} \alpha^*_{child(i,j)} \tag{2}$$

2.1 Maximum Sensing Rate Arrival Curve

The simplest option in bounding the sensing input at a given sensor node is based on its maximum sensing rate which is either due to the way the sensing unit is designed or limited to a certain value by the sensor network application's task in observing a certain phenomenon. For example, it might be known that in a temperature surveillance sensor system, the temperature does not have to be reported more than once per second at most. The arrival curve for a sensor node i corresponding to simply putting a bound on the maximum sensing rate is

$$\alpha_i(t) = p_i t = \gamma_{p_i,0}(t) \tag{3}$$

This arrival curve can be used in situations where all sensor nodes are set up to periodically report the condition in a sensor field. The set of sensible arrival curve candidates is certainly larger than the arrival curves described above. The

more knowledge on the sensing operation and its characteristics is incorporated into the arrival curve for the sensing input the better the performance bounds become.

2.2 Rate-Latency Service Curve

Next, the service curve has to be specified. The service curve depends on the way packets are scheduled in a sensor node which mainly depends on link layer characteristics. More specific, the service curve depends on how the duty cycle and therefore the energy-efficiency goals are set.

The service curve captures the characteristics with which sensor data is forwarded by the sensor nodes towards the sink. It abstracts from the specifics and idiosyncracies of the link layer and makes a statement on the minimum service that can be assumed even in the worst case. A typical and well-known example of a service curve from traditional traffic control in a packet-switched network is

$$\beta_{R,T}(t) = R\,[t-T]^+ \tag{4}$$

where the notation $[x]^+$ equals x if $x \geq 0$ and 0 otherwise. This is often also called a rate-latency service curve. The latency term nicely captures the characteristics induced by the application of a duty cycle concept. Whenever the duty cycle approach is applied there is the chance that sensed data or data to be forwarded arrives after the last duty cycle (of the next hop!) is just over and thus a fixed latency occurs until the forwarding capacity is available again. In a simple duty cycle scheme this latency would need to be accounted for for all data transfers. For the forwarding capacity it is assumed that it can be lower bounded by a fixed rate which depends on transceiver speed, the chosen link layer protocol and the duty cycle. So, with some new parameters the following service curve at sensor node i is obtained:

$$\beta_i(t) = \beta_{f_i,l_i}(t) = f_i[t-l_i]^+ \tag{5}$$

Here f_i and l_i denote the forwarding rate and forwarding latency for sensor node i.

2.3 Network Flow Analysis

Finally, the output of sensor node i, i.e. the traffic which it forwards to its parent in the tree, is constrained by the following arrival curve:

$$\alpha_i^* = \bar{\alpha}_i \oslash \beta_i = \left(\alpha_i + \sum_{j=1}^{n_i} \alpha^*_{child(i,j)}\right) \oslash \beta_i \tag{6}$$

In order to calculate a network-wide characteristic like the maximum information transfer delay or local buffer requirements especially at the most challenged sensor node just below the sink (which is called node 1 from now on) an iterative procedure to calculate the network internal flows is required:

1. Let us assume that arrival curves for the sensed input α_i and service curves β_i for sensor node i, $i = 1, \ldots, N$, are given.
2. For all leaf nodes the output bound α_i^* can be calculated according to (6). Each leaf node is now marked as "calculated".
3. For all nodes only having children which are marked "calculated" the output bound α_i^* can be calculated according to (6) and they can again be marked "calculated".
4. If node 1 is marked "calculated" the algorithm terminates, otherwise go to step 3.

After the network internal flows are computed according to this procedure, the local per node delay bounds D_i for each sensor node i can be calculated according to a basic network calculus result [2, chapter 1]:

$$D_i = h(\bar{\alpha}_i, \beta_i) = \sup_{s \geq 0}\{\inf\{\tau \geq 0 : \bar{\alpha}_i(s) \leq \beta_i(s+\tau)\}\} \tag{7}$$

To compute the total information transfer delay $\bar{D}_i$ for a given sensor node i the per node delay bounds on the path $P(i)$ to the sink need to be added:

$$\bar{D}_i = \sum_{i \in P(i)} D_i \tag{8}$$

The maximum information transfer delay in the sensor network can then obviously be calculated as $\bar{D} = \max_{i=1,\ldots,N} \bar{D}_i$. Note that this kind of analysis assumes FIFO scheduling at the sensor nodes which however should be the case in most practical cases.

3 Advanced Sensor Network Calculus

After this brief walk-through the sensor network calculus basics, we will discuss some of the more advanced techniques we have developed to further customize network calculus to the wireless sensor network setting as well as some of the applications of the framework we have proposed.

We have seen in the previous section how the single sink communication pattern typically found in wireless sensor networks was used to iteratively work out the internal traffic flow bounds inside the network and use these to calculate delay bounds in an additive fashion. However, one of the strengths of network calculus is its powerful concatenation result which allows in general to achieve better bounds when a tandem of servers is first min-plus convoluted to a single system compared to an additive analysis of the separate servers. This concatenation result is not directly applicable in a wireless sensor network scenario even when only considering the simple single sink case. Therefore, we have generalized the concatenation result for general feedforward networks in [5], introducing a principle called "pay multiplexing only once" which makes optimal use of subpaths shared between flows and achieves improvements over the additive bounds which may be on the order of magnitudes depending on the scenario. A further

extension of the basic sensor network calculus which we also describe in [5] is the integration of maximum service curves into the sensor network calculus which allows to improve the bounds on the network-internal flows and thus in turn lowers the performance bounds, again often very considerably. All these techniques, among other general network calculus techniques, have been implemented in the so-called DISCO Network Calculator. As we believe that tool support is of great importance for a wide acceptance of the sensor network calculus we provide the DISCO Network Calculator in the public domain[1].

Apart from trying to push the sensor network calculus forward methodwise, we have also illustrated how to apply it for several design and control issues in wireless sensor networks. In [1] we have shown how a buffer dimensioning of the sensor nodes may be performed based on the worst case analyses of sensor network calculus such that no information is lost due to buffer overflow inside the network. Furthermore, we also discussed in [1] how different choices of duty cycles affect the information transfer delay. In [3], we considered the case of a randomly deployed sensor network and how this further dimension of uncertainty can be factored into the sensor network calculus. In particular we discussed how constraints from topology control may be used to improve the performance bounds from the sensor network calculus. Thus we proposed to guide topology control decisions based on the sensor network calculus models. In [4] we used the advanced sensor network calculus result discussed in the previous paragraph to investigate scenarios with multiple sinks. In particular we demonstrated how sensor network calculus can be used to dimension the number of sinks as well as their placement in the sensor field.

4 Open Issues and Future Work Items

While we believe the sensor network calculus to have potential, there are still many open issues and correspondingly opportunities for future work. One immediate issue arising from the use of a deterministic analytical framework is the question how to capture the inherently stochastic nature of wireless communications. Here, we plan to integrate lately upcoming stochastic extensions of network calculus [6], which however again need to be customized for the sensor network case. Another issue is how to take in-network processing as is frequently proposed for wireless sensor networks into account. In [7] we have proposed a network calculus that allows for the scaling of data flows. This development should enable modelling of typical in-network processing techniques as for example aggregation of information. Furthermore, it should also be possible to accomodate the mobility of sensor nodes and/or sinks. As in many scenarios this is a kind of controlled mobility there is hope to capture even this difficult characteristic of advanced wireless sensor network scenarios.

Apart from these fundamental issues for the sensor network calculus, it is also important to demonstrate its usefulness in further applications. At the moment

[1] See http://disco.informatik.uni-kl.de/content/Downloads.

we design a task admission control scheme based on sensor network calculus for sensor networks that may have several concurrent tasks. Another work item could be a scheme where sleeping nodes are activated such that certain performance bounds can still be satisfied. Apart from these issues the presented framework should also be validated by packet-level simulations in order to increase the fidelity in the predictive power of our models. Especially this last point deserves our immediate attention and is already currently under investigation.

References

1. Jens Schmitt and Utz Roedig. Sensor Network Calculus - A Framework for Worst Case Analysis. In Proc. of IEEE/ACM Int. Conf. on Distributed Computing in Sensor Systems (DCOSS'05), pages 141-154. Springer, LNCS 3560, June 2005.
2. J.-Y. Le Boudec and P. Thiran. Network Calculus - A Theory of Deterministic Queueing Systems for the Internet. Springer, LNCS 2050, 2001.
3. Jens Schmitt and Utz Roedig. Worst Case Dimensioning of Wireless Sensor Networks under Uncertain Topologies. In Proc. of 3rd International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'05), Workshop on Resource Allocation in Wireless Networks, Riva del Garda, Italy. IEEE, April 2005.
4. Jens B. Schmitt, Frank A. Zdarsky, and Utz Roedig. Sensor Network Calculus with Multiple Sinks. In Proc. of IFIP Networking 2006, Workshop on Performance Control in Wireless Sensor Networks, Coimbra, Portugal. Springer LNCS, May 2006.
5. Jens B. Schmitt, Frank A. Zdarsky, and Ivan Martinovic. Performance Bounds in Feed-Forward Networks under Blind Multiplexing. Technical Report 349/06. University of Kaiserslautern, Germany, April 2006.
6. Yuming Jiang. A Basic Stochastic Network Calculus. In Proc. ACM SIGCOMM 2006, Pisa, Italy, September 2006.
7. Markus Fidler and Jens B. Schmitt. On the Way to a Distributed Systems Calculus: An End-to-End Network Calculus with Data Scaling. In Proc. ACM SIGMETRICS/Performance 2006 (SIGMETRICS'06), St. Malo, France, June 2006.

Mining Event Patterns in Sensor Networks *

Kay Römer

Institute for Pervasive Computing
ETH Zurich, 8092 Zurich, Switzerland
roemer@inf.ethz.ch

Abstract. Many sensor network applications are concerned with discovering interesting patterns among observed real-world events. Often, only limited apriori knowledge exists about the patterns to be found eventually. Here, raw streams of sensor readings are collected at the sink for later offline analysis – resulting in a large communication overhead. In this paper, we explore the use of in-network data mining techniques to discover frequent event patterns and their spatial and temporal properties. With that approach, compact event patterns rather than raw data streams are sent to the sink. We also discuss issues with the implementation of our proposal and report our experience with preliminary experiments.

1 Introduction

Wireless sensor networks have been successfully applied for detailed observation of a variety of real-world phenomena. Many of these applications are of a highly exploratory nature, where only a very rough idea of the expected findings exists before the experiment. In this case, streams of raw sensor readings from every node are typically delivered to a central sink for later offline analysis in order to find interesting patterns in the data – resulting in a large data volume that has to be delivered through the network (e.g., [5]).

A similar approach is used in the context of monitoring and debugging sensor networks [4]. Experience has shown that subtle real-world influences and large scale are the cause of numerous bugs and indeterministic behavior of sensor networks. Again, since the nature of these problems is often unknown in advance, testbeds and deployment support networks have been proposed to deliver high-volume event logs from every sensor node to a central sink for offline analysis in order to identify patterns that lead to failure.

In the above cases, missing advance knowledge about the patterns to be found eventually in the data limits the applicability of sophisticated in-network data processing and reduction techniques. Rather, raw data streams are delivered to the sink for later analysis. The resulting large data volume is a serious obstacle for deploying long-lived and large-scale sensor networks.

In this position paper, we explore the use of distributed data mining techniques to discover potentially interesting data patterns in a sensor network for the above type

* The work presented in this paper was supported (in part) by the Swiss National Science Foundation under grant number 5005-67322 (NCCR-MICS).

of applications. Rather than transmitting raw data streams from every sensor to the sink, only compact patterns mined at sensor nodes are transmitted to the sink, thus contributing to long-lived and large-scale sensor network deployments.

2 Approach

With our approach, a user can pose a *mining query* to the sensor network. This query is executed by a distributed runtime system in the sensor network. As a result, the user will receive at regular (but long) intervals a set of discovered event patterns from each node. An event pattern is a frequently occurring set of events observed at different nodes in the network.

A mining query specifies the types of events a user is interested in. The notion of event refers to a user-specified state change at a sensor node (e.g., sudden drop of temperature as measured by a sensor). In the context of our work, an event is simply an identifier (e.g., "temperature-drop") plus a timestamp when this event occurred according to some global time scale. We assume that time is partitioned into equal-sized *epochs*.

In addition, a mining query contains a number of constraints on the event patterns the system should look for. These are needed to limit the huge search space of potential event patterns.

The distributed mining algorithm then proceeds as follows. Every node in the network continuously collects event notifications from nodes in a user-defined network *neighborhood* (the size of which is specified by the mining query) using in-network aggregation techniques. The events that have been collected in this way during a user-defined amount of time called *history* (the duration of which is specified by the mining query as an integral number of epochs) are then fed to a mining algorithm. This algorithm executed at node n mines patterns of the following form:

$$A_1 \wedge ... \wedge A_m \Rightarrow E\,[S, C] \tag{1}$$

meaning that an event of type E occurred at node n with support S and confidence C given that antecedents A_i all hold true. Every antecedent A_i is of the form

$$A_i = (E_i, D_i, T_i, N_i) \tag{2}$$

meaning that A_i is true iff a certain type of event E_i occurred N_i times at a distance D_i from node n and T_i time units before E. D_i, T_i, and N_i usually denote intervals such as T_i = *"more than five minutes ago"*, D_i = *"less than 20 meters away"*, or N_i = *"between one and five times"*.

Support S is a number between 0 and 1 indicating how often this pattern could be found over time. Confidence C is a number between 0 and 1 indicating how strong the implication $\Rightarrow$ is. Note that the above patterns are a specific instance of *association rules* [1].

An example pattern discovered by this approach would be:

$$(\text{tempdrop}, [0, 10\text{m}], [0, 5\text{min}], [3, \text{inf}]) \Rightarrow \text{tempdrop}[0.95, 0.8] \tag{3}$$

meaning that node n observed a "temperature drop" event with support 0.95 and confidence 0.8 if at least 3 nodes no more than 20 meters away from n did also observe a "temperature drop" event during the last 5 minutes. The following sections contain details on some selected aspects of the basic approach described above.

2.1 Mining Queries

A mining query contains the following information:

- **epochlen**: the duration of an epoch in seconds.
- **neighborhood**: the radius of the neighborhood around a node given in network hops or meters.
- **history**: the length of the history given in epochs.
- **minimum support**: a number between 0 and 1 indicating the minimum frequency required for reported patterns.
- **minimum confidence**: a number between 0 and 1 indicating the minimum confidence for the implication $\Rightarrow$ required for reported patterns.
- **distance quantization**: quantization of Euclidean distance between nodes into a small set of partitions. Each partition is assigned a name (e.g., near=(0m,5m], far=(5m,∞]).
- **time quantization**: quantization of time between events into a small set of partitions. Each partition is assigned a name (e.g., now=0, recent=[1, 5epochs], old=(5epochs, ∞]).
- **event frequency quantization**: quantization of number of events into a small set of partitions. Each partition is assigned a name (e.g., none=0, some=[1,∞]).

The purpose of quantization is to cut down the search space to be considered by the mining algorithm by binning event occurrences into a small number of "partitions". Note that quantization is a critical issue as it affects the patterns that will be found eventually.

2.2 Event Collection

As stated earlier, each node in the network collects event notifications from nodes in a neighborhood. This can be achieved by applying a framework that supports neighborhood abstractions such as Abstract Regions [6]. These tools allow a node in the network to define a *neighborhood* that consists of a set of nodes that fulfill certain conditions such as to be within a given distance of the node. Primitives for collecting data from the nodes in a neighborhood are provided. Using the quantization of distance discussed in the previous section, in-network data aggregation is applied to collect the frequency of each event for each distance partition in the neighborhood of the node. Using the example partitions from the previous section, we would count the occurrence of each event at distances "near" and "far" using in-network aggregation.

2.3 Mining Algorithm

The overall approach we apply is to transform the set of events collected from the neighborhood during history into an assignment of $\{\mathsf{TRUE}, \mathsf{FALSE}\}$ to a small set of binary variables. The values of these variables characterize the collected events in a compact way and serve as the input for the data mining algorithm.

For the transformation, we make use of the quantization of distances, time, and event frequencies specified by the mining query. Essentially, there is one binary variable $< E > . < D > . < T > . < N >$ for each possible combination of an event $< E >$, a distance partition $< D >$, a time partition $< T >$, and an event frequency partition $< N >$. For example, the variable `tempdrop.near.now.some` would be TRUE, iff event "temperature drop" was observed by at least one node (partition "some"), during this epoch (partition "now"), not more than 5 meters away (partition "near").

In addition, we include a binary variable $< E >$ for each possible event $< E >$ that is TRUE iff this event occurred at the node executing the mining algorithm in the current epoch. For example, the variable `tempdrop` would be TRUE iff the node executing the mining algorithm observed a "temperature drop" event in the current epoch.

By applying this transformation, we obtain the set of binary variables with value TRUE for each epoch. Over time, this results in a stream of sets of binary variables with value TRUE.

To discover event patterns matching the mining query, we need to find sets of the above binary variables that occur with a minimum frequency in the stream. This lower frequency bound is given by the minimum support value in the mining query. Among the resulting frequent candidate sets, the ones satisfying the minimum confidence requirement are selected. The remaining sets can then be easily transformed into patterns of the form given in Equation 1.

Discovering such frequent sets of binary variables from a stream is a standard data mining task. In the literature, this problem is referred to as *mining of frequent itemsets* (each binary variable can be considered as an item that is either present in the set or not). In the recent past, various algorithms have been proposed to solve this problem with constrained resources over streams of itemsets (e.g., [2, 3]). From the resulting frequent itemsets, the ones satisfying the minimum confidence requirement are selected as in [1].

3 Implementation Issues

Clearly the major challenge in implementing our proposal is that of fitting the mining approach described in the previous section into the constrained communication, computational, and memory resources of a sensor node. One of the most challenging aspects is to implement the mining algorithm within the constrained memory resources of a sensor node. The BTnode [7] platform, for example, offers 256 kB of bank-switched RAM.

The memory footprint of algorithms for mining frequent itemsets is largely a function of the number of frequent itemsets discovered from the data stream. To examine the number of such frequent itemsets for a practical application, we performed an experiment using sensor data collected during one month from 54 sensor nodes in the Intel

Research Lab Berkeley [8]. This dataset was collected with an epoch duration of about 30 seconds (resulting in a total of about 65000 epochs) and contains, among others, temperature and light readings.

In our experiment, we consider two types of events: *temperature* and *light* events. Each sensor node with a temperature reading > 23 degrees Celsius in an epoch emits a temperature event in this epoch. Every sensor node with a light reading > 300 Lux emits a light event. Otherwise, we use the quantization from the example given in Sect. 2.1 with a neighborhood size of 10 meters and a history duration of 10 epochs.

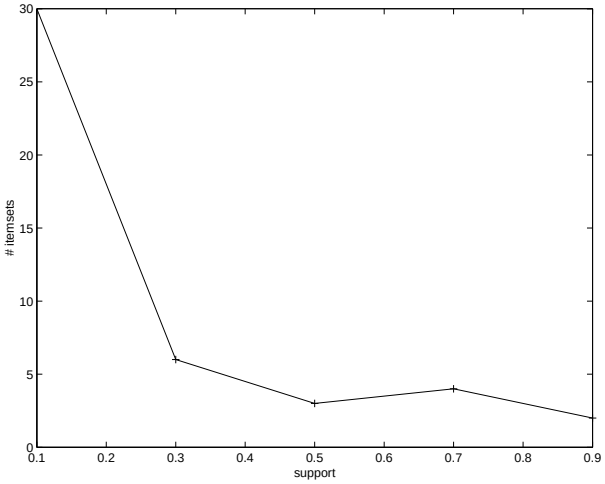


Fig. 1. Number of discovered maximal itemsets for different support values.

For our experiment, we consider the mining algorithm that would be executed on mote with ID 1 in the dataset. With the above settings, mote 1 generates a temperature event in about 23% of all epochs and a light event in about 14% of all epochs. We would expect a strong correlation of the occurrence of these events on mote 1 with the light and temperature events in the neighborhood of the mote.

Applying the method given in Sect. 2.3, we create a stream of itemsets which is then fed to an algorithm to discover maximal itemsets (a variant of [1]) for different values of minimum support. Here, a maximal itemset is a frequent itemset that has no proper supersets that are also frequent.

The results shown in Figure 3 are encouraging as the number of maximal itemsets is very small over the whole range of considered support values. Note that a single itemset in this experiment can be represented with 26 bits, since there are 26 different binary variables.

For a minimum support of 0.9 we obtain two maximal itemsets that result in the following patterns for node 1 in the format of Eq. 1:

$$(\mathrm{t}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{recent}, *, \mathrm{some}) \wedge (\mathrm{t}, \mathrm{old}, *, \mathrm{some}) \Rightarrow t\ [0.96, 0.38]$$

$$(\mathrm{t}, *, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \mathrm{now}, \mathrm{far}, \mathrm{some}) \wedge (\mathrm{l}, \{\mathrm{old}, \mathrm{recent}\}, *, \mathrm{some}) \Rightarrow l\ [0.92, 0.32]$$

Here, “t” and “l” refer to temperature and light events as defined above. The notations “{...,...}” and “*” mean that the enclosing antecedent is valid for the set of given partition identifiers or for all possible partition identifiers, respectively. The rules indicate that – as expected – the occurrence of temperature/light events at mote 1 is correlated with the occurrence of these events in the neighborhood of the node.

4 Conclusions and Future Work

We have examined the use of data mining techniques to discover frequent event patterns and their spatio-temporal relationships within a sensor network, such that compact patterns rather than raw data streams would have to be transmitted from nodes to the sink. In particular, such a system would be helpful to support exploratory settings, where only a rough idea of the actual findings exists before the experiment.

We have also discussed challenges in implementing this proposal on sensor nodes. In particular, we have identified the memory consumption of itemset discovery algorithms. We have performed an experiment with real-world data to motivate that an implementation is feasible. The work reported in this paper is a first step towards a distributed event-pattern-mining system for sensor networks.

References

1. R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules. In *VLDB 1994*, Santiago de Chile, Chile, September 1994.
2. H.-F. Li, S.-Y. Lee, and M.-K. Shan. An Efficient Algorithm for Mining Frequent Itemsets over the entire History of Data Streams. In *First Intl. Workshop on Knowledge Discovery in Data Streams 2004*, Pisa, Italy, September 2004.
3. G. S. Manku and R. Motwani. Approximate Frequency Counts over Data Streams. In *VLDB 2002*, Kong Kong, China, August 2002.
4. N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the Sensor Network Debugger. In *Sensys 2005*, San Diego, USA, November 2005.
5. R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a Sensor Network Expedition. In *EWSN 2004*, Berlin, Germany, January 2004.
6. M. Welsh and G. Mainland. Programming Sensor Networks Using Abstract Regions. In *NSDI 2004*, Boston, USA, March 2004.
7. BTnodes. www.btnode.ethz.ch.
8. Intel Lab Sensor Data. http://berkeley.intel-research.net/labdata/.

Statistical Modeling of Sensor Data and its Application to Outlier Detection

Christoph Heinz Bernhard Seeger

Department of Mathematics and Computer Science
Philipps University Marburg, Germany
{heinzch,seeger}@mathematik.uni-marburg.de

Abstract. Various applications rely on a continuous processing of data streams originating from a network of interconnected and collaborated sensors. The processing of those streams has turned out to be a difficult task as sensors only have limited resources and the data they produce is inherently uncertain and unreliable. In order to bridge the gap from raw, uncertain sensor readings to a meaningful model of the physical phenomenon observed, statistical modeling techniques have proved to be an adequate approach. By means of a statistical model, a wide range of sensor network related topics can be covered. In this work, we present an initial approach to tackle an important problem in sensor processing, namely the detection of outliers, with a statistical model.

1 Introduction

Recent advances in hardware technology combined with decreasing production cost for lightweight devices have facilitated the application of wireless sensor networks for monitoring entities in a plethora of real-world scenarios, e.g. environment monitoring, industry monitoring. In these scenarios, a large number of sensors is deployed and each one continuously monitors physical entities like temperature and air pressure. The raw sensor readings are typically transmitted to a central backend that provides an interface to query the sensor network.

The configuration of a sensor network comprises suitable settings for network topology, routing, and communication protocols [1]. Since the resources of a sensor are limited, the energy efficiency is a crucial factor in this configuration. Besides these low-level problems, a sensor network also has to cope with high-level problems: First, a sensor network produces huge amounts of data in form of transient streams. The time-critical nature of most applications combined with limited resources requires to give up the paradigm of exact answers and to use approximate answers instead [2]. Second, even if we could store all data, we must take the inherent uncertainty and unreliability of the data into account [3]. This uncertainty is due to the facts that a sensor can only provide discrete samples of a (continuous) physical phenomenon and that it is additionally subject to interfering effects like noise, hardware failures, inaccuracies. Hence, the querying of the raw sensor readings may produce misleading or even wrong answers.

Recently, the database research community has addressed those high-level problems. Especially research in data stream processing has come to the fore [2]. The processing of data streams is challenging as their requirements render the application of common database technologies unfeasible [4]. The processing of sensor data streams is even more challenging due to their inherent characteristics: data uncertainty, intra- and inter-stream correlations, sensitivity to energy consumption [5].

A promising approach that takes those characteristics into account is to incorporate statistical modeling techniques into sensor network processing. To convey a notion of the benefits of statistical models in sensor networks, let us sketch some recently proposed approaches: One, for example, is to equip query answers with probabilistic estimates of their validity to cope with imprecise sensor data [6]. In order to clean noisy sensors, [7] combines prior knowledge with noise characteristics of the sensor to obtain more accurate sensor readings. [3] also tackles the cleaning of sensor streams by exploiting temporal and spatial correlations of the sensor readings. The approach presented in [8] provides a statistical model for the complete sensor network that allows to query the network while acquiring new sensor readings only if necessary.

The above approaches share the property that they develop a statistical model for the distribution of the streams generated in a sensor network. The main assumption is that the sensor readings are samples of different physical phenomena under observation. If we describe the entirety of phenomena in terms of random variables, we have a variety of convenient statistical estimation techniques at our disposal. In this context, it is vital for further analysis to reveal the distribution of a random variable, more specifically its probability density function [9]. In [4], [10], we presented solutions to this problem for transient data streams. As these techniques particularly suit for the sensor stream scenario, we propose to use the statistical models they provide as point of origin for further analysis of the sensors. In this work, we present an initial approach to tackle outlier detection, an important task in almost every application on top of sensor networks, with the help of these statistical models. Before going into details, we will give a brief overview of the development of statistical models for sensor networks.

2 Statistical Modeling of Sensor Data

A sensor network acquires samples of physical phenomena with sensors located at the network nodes. We assume that each sensor measures at each time instant a single, real-valued attribute X_i, e.g. temperature. The sensor transmits the raw readings to a central basestation. With respect to the aforementioned unreliability of those readings, the basestation serves as an intermediate preprocessor. In the preprocessing step, the raw sensor readings are transformed into a meaningful statistical model of the complete sensor network. This model is continuously published to the query processing module. This module executes the posed queries with respect to the statistical model. For example, one possible

query is to determine all nodes whose reported temperatures are labeled as outliers with respect to the statistical model. The general architecture for the sensor stream processing is illustrated in Fig. 1. In the following, we will concentrate

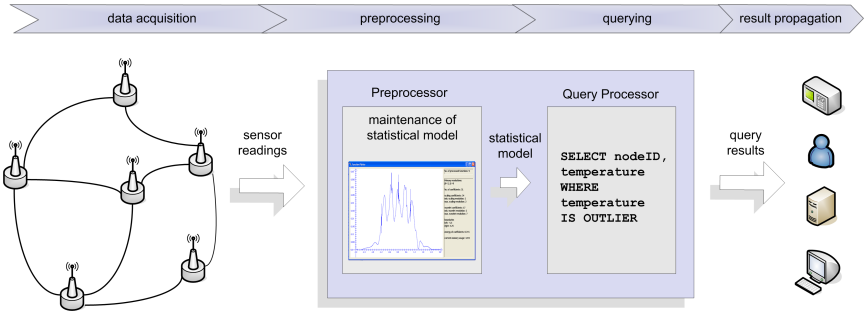


Fig. 1. processing architecture

on the computation of the statistical model within the preprocessor.

Point of origin is to model the set of attributes $X_1, ..., X_n$ as an n-dimensional random variable $X = (X_1, ..., X_n)$, i.e., we assume that the sensor readings are samples of the random variable X. Given the probability density function (pdf) $f(X_1, ..., X_n)$, we can determine the distribution of X in terms of the probabilities of arbitrary attribute constellations:

$$P(X_1 \in [a_1, b_1], ..., X_n \in [a_n, b_n]) = \int_{a_1}^{b_1} ... \int_{a_n}^{b_n} f(X_1, ..., X_n) dx_1 ... dx_n. \quad (1)$$

The knowledge of the pdf is crucial as it gives a comprehensive summary of the process described by the random variable. Not only can we determine the above probabilities, but also determine meaningful characteristics like mean, variance, quantiles, correlations.

Let us give an example: For two sensors that measure temperature and air pressure respectively, we could conclude with their pdf that the probability of a temperature above 25 degrees celcius and an air pressure lower than 1000 hPa is extremely low. We also could determine the mean temperature or the correlation, i.e. the degree of linear dependency, between temperature and air pressure.

However, the question remains how we determine the pdf for a given set of measured attributes? In real-world scenarios, we must assume that we have no prior knowledge of the sensor stream distributions; we only have the raw sensor readings. One approach is to assume that the unknown pdf belongs to an a priori known class of densities, e.g. Gaussians. Given a Gaussian distribution, it remains to estimate its mean and variance with the help of the sensor readings. Due to its simplicity, this is a practically relevant approach, e.g. [7]. However, if the random variable does not follow the preset distribution, the resulting model is likely to be useless.

On account of this, we have concentrated in our work on so-called assumption-free density estimation techniques provided by mathematical statistics [9]. Those techniques are very appealing as they let the data speak strictly for themselves without any assumptions. As the computational complexity of these techniques prevents their direct application to data streams, we developed adaptations [4], [10] based on kernels and wavelets respectively. Both techniques continuously provide - with computational low cost - suitable density estimates that keep pace with current trends in the stream. With those density estimates as statistical model of the streams in a sensor network, we can gain insight into the physical phenomena observed by this network. In the following, we will illustrate this with their application to outlier detection.

3 Outlier Detection for Sensor Data

The detection of outliers in a timely fashion is an important task for virtually all applications on top of sensor networks. In facility monitoring, for example, an alert shall be triggered in case of exceptionally high temperatures. Another example is the food industry where perishable items shall be timely detected.

Generally, outlier detection has a long history in statistics and database research and there are many ways to define an outlier. Intuitively, we expect an outlier to be unusual or unexpected in comparison to a given data set; its occurrence is 'improbable'. Given the terminology introduced above, we label a point as an outlier if it lies in a region with low probability. This informal definition includes two important aspects: First, we consider the region around the outlier, quasi its neighborhood, and found the decision on the probability of the region. Second, we set a threshold for this probability. This threshold determines the sensitivity of the outlier classification. We incorporate both aspects in the following formal definition of an outlier:

Definition 1 *Let f be a pdf with support $[a, b]$ and $\epsilon, \delta \in (0, 1)$. A point x is an **outlier**, if $P(X \in [x - \frac{\epsilon(b-a)}{2}, x + \frac{\epsilon(b-a)}{2}]) \leq \delta$ holds.*

The parameter ϵ determines the width of the region while δ determines the rate of false positives and false negatives. The higher δ is set, the more 'normal' points will be labeled as outliers. The lower it is set, the more outliers will not be detected. The explicit setting of the parameters depends on the concrete application. For illustrative purposes, Fig. 2 presents an example for the definition of outliers.

With the above definition, we can develop an online algorithm to detect outliers in sensor streams. The chief part of the algorithm is the maintenance of a density estimate with the techniques mentioned above. Based on this estimate, we label a new sensor reading either as outlier or not. Except the reading is definitely not possible, e.g. negative velocities, we incorporate it into the density estimate. This ensures that we do not label values becoming more frequent always as outliers. The more often they appear, the more this will be reflected in the density estimate, i.e., their probability increases and consequently the probability of being labeled as outlier decreases.

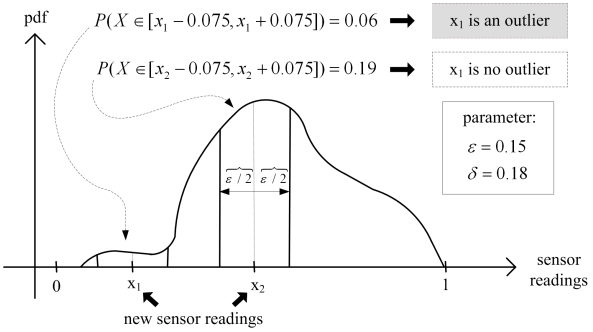


Fig. 2. detection of outliers

4 Conclusions

In this work, we investigated the augmentation of sensor network querying by meaningful statistical models. Instead of exploring the raw sensor readings, a statistical model offers a more reliable way to gain insight into the physical phenomena observed. A key ingredient of statistical models is the probability density function as it provides a comprehensive summary. Based on online computable estimates of the probability density function, we presented an initial approach to detect outliers in streaming sensor data. However, outlier detection is only one of many possibilities to enrich sensor querying with statistical modeling techniques. In fact, we expect this research direction to become highly relevant in future.

References

1. Tubaishat, M., Madria, S.: Sensor networks: an overview. IEEE Potentials **22(2)** (2003)
2. Babcock, B., Babu, S., Datar, M., Motwani, R., Widom, J.: Models and Issues in Data Stream Systems. In: PODS. (2002)
3. Jeffery, S.R., Alonso, G., Franklin, M.J., Hong, W., Widom, J.: A Pipelined Framework for Online Cleaning of Sensor Data Streams. In: Proc. of ICDE. (2006)
4. Blohsfeld, B., Heinz, C., Seeger, B.: Maintaining Nonparametric Estimators over Data Streams. In: Proc. of BTW. (2005)
5. Liu, H., Hwang, S., Srivastava, J.: Probabilistic stream relational algebra: A data model for sensor data streams. Technical report, University of Minnesota (2004)
6. Cheng, R., Kalashnikov, D., Prabhakar, S.: Evaluating Probabilistic Queries over Imprecise Data. In: Proc. of ACM SIGMOD. (2003)
7. Elnahrawy, E., Nath, B.: Cleaning and Querying Noisy Sensors. In: Proc. of WSNA. (2003)
8. Deshpande, A., Guestrin, C., Hellerstein, J., Madden, S., Hong, W.: Model-Driven Data Acquisition in Sensor Networks. In: Proc. of VLDB. (2004)
9. Silverman, B.: Density Estimation for Statistics and Data Analysis. Chapman and Hall (1986)
10. Heinz, C., Seeger, B.: Wavelet Density Estimators over Data Streams. In: Proc. of SAC. (2005)

SNoW^5: A versatile ultra low power modular node for wireless ad hoc sensor networking

Reiner Kolla, Marcel Baunach, Clemens Mühlberger

University of Würzburg, Bavaria, Germany
{kolla,baunach,muehlberger}@informatik.uni-wuerzburg.de

Abstract This technical report presents the architecture of the ultra low power sensor node *SNoW*^{5} for ad hoc wireless sensor networking (WSN). *SNoW*^{5} was developed at the University of Wuerzburg and aims on WSN research, educational and commercial applications. An overview over the basic concept, its hardware design and a tabular comparison to other existing nodes will be given. We conclude this technical report with a survey of our research so far and a short look on future works.

1 Introduction

Powerful information technology is already an inherent part of everybody's daily life. Unfortunately, most systems are rather large in size and depend on static and inflexible infrastructures. Therefore, current research focuses on small-sized and mobile devices which will be deployable into almost any object to directly support humans as well as machines. Carefully designed networks of flexible autonomous devices will lead to increased convenience, performance, security, etc. However, one must always take into concern, that minimalistic computers have low performance due to very restrictive requirements like ultra low power consumption. Thus, the demand for wireless networks of small autonomous devices increased heavily within the last few years. These wireless sensor networks (WSN) use the combined power of many small devices to solve even complex problems under exceptional circumstances for which a single device was too weak. Nevertheless, the successful coordination and interaction of these sensor nodes is a hard problem, comprising research areas like communication, self-organization, fault-tolerance, distributed algorithms and low-power design in both hardware and software.

Several sensor nodes are already available for research and commercial applications but after careful examination of these nodes we decided to develop a new one from scratch to meet our requirements more exactly. The most detailed knowledge of all software and hardware concepts combined with the large amount of features allows us to precisely observe theoretical considerations under hard real-life conditions. Thus, this technical report describes which hardware components were selected to make *SNoW*^{5} as versatile as possible (see Figure 1).

2 *SNoW*^{5} specifications, features and extensions

This section describes the hardware components used for *SNoW*^{5} in detail and illuminates why each one was chosen (see Figure 2). As already mentioned, some other

sensor boards already exist. Table 1 shows a detailed summary over the specifications and features of *SNoW*^{5} and compares it to a few other nodes available. As you can see, the named sensor nodes are somewhat similar in some points. In some other points the differences are rather large.

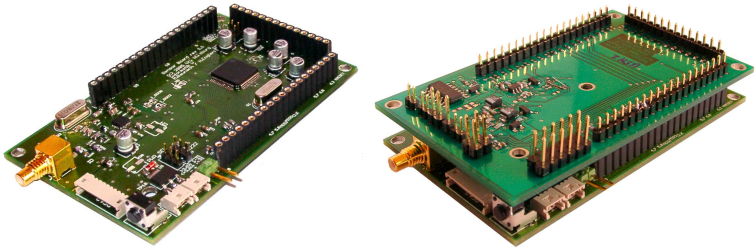


Figure 1. The *SNoW*^{5} main board (left) and with one stacked daughter board (right)

As *energy efficiency* is an important aspect, power consumption was minimized to make *SNoW*^{5} work for a long time even with limited power supply like a simple battery. The usage of renewable power supplies via solar panels and Piezo elements is planned for the near future. Therefore not only ultra low power components but also extended support for energy saving modes was required in both hardware and software. Special power saving concepts were implemented within our operating system and the communication protocols for example.

A *powerful core MCU* was another important issue. After careful comparison of quite a number of architectures, we selected TI's 16 bit ultra low power MCU family MSP430x16xx [1]. The device we prefer is the MSP430F1611 with 48 kB of flash memory, 10 kB of RAM and five operation modes with different power consumptions from 0.2 µA up to 9 µA at 8 MHz (software adjustable). In addition, various on chip peripherals are provided by the MSP430 and can be used on board or through the node's pin headers: 8 × 12 bit ADC, 2 × 12 bit DAC, 2 × 16 bit timer with capture/compare, brownout detector, hardware multiplier, 6 × 8 bit general purpose digital I/O ports and integrated bus protocols for serial communication like SPI, I^{2}C and RS232. These features allow comfortable usage of external analog and digital components like sensors and actors.

Another goal was *modularity* and *customizability*. As the support for a large variety of additional digital and analog devices is very important for our research, we decided to pursue a stackable design where sensors and actors can be easily attached to each node. Therefore we made all relevant digital and analog signals of the onboard devices available on pin headers. In this way it is not only possible to plug several expansion boards on a node at the same time to expand its capabilities as required by the specific application, this concept also allows us to avoid placing any sensors directly on the main board as this would restrict the versatility of *SNoW*^{5} due to preassigned I/O signals that could be used wiser within some other applications. This concept even permits buses for connecting devices on different expansion boards. We preserve the option to mount sensors on the case of the node where they are in direct contact with the environment.

Quite a number of expansion boards are already available or under development providing the possibility to attach resistive sensors for example. An ultrasonic transceiver

for distance measurement is supported as well as an acoustic amplifier for audio applications. In the near future special communication boards with IR, USB, WLAN, GPRS and extensions for orientation and movement detection can be expected.

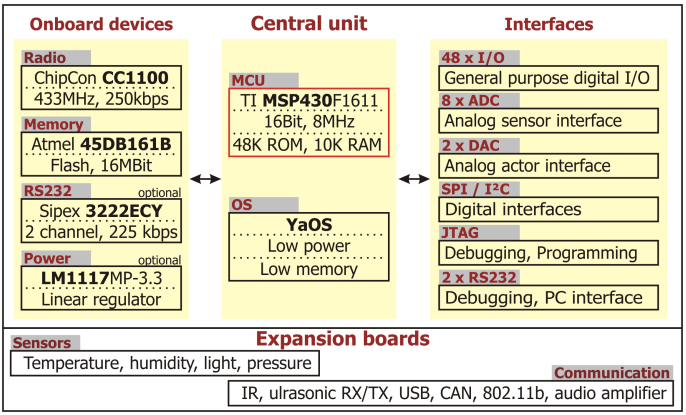


Figure 2. *SNoW*^{5} – a rough overview

A *compact design* was possible by careful selection of small devices. Nevertheless we rated convenient debugging and adaptability more important than small dimensions. For the benefit of easy and rapid prototyping of expansion boards we decided to use a 2.54 mm grid for the headers. Though this increases the overall size of the nodes we support standard bread boards and allow comfortable debugging via scopes and logic analyzers. For specific applications, the node's dimension can be easily reduced to about 50% of size by omitting debugging headers.

The next focus was on *flexible communication*. As long distance wireless communication is essential within any WSN we decided to use the highly configurable multi channel radio transceiver ChipCon CC1100 [2] with adjustable base frequency (see Table 1). This is of special interest for researching various wireless communication protocols. Some special features of the CC1100 are SPI interface, Wake-on-Radio RX (≥ 1.8 µA), integrated hardware address check and its digital RSSI/LQI output. The latter can be used to adjust TX power for dynamic cell sizes or even to roughly localize a node relative to its neighbour nodes as proposed in [3]. Its data rate ranges from $1.2 - 500$ kbit/s and the two 64 byte RX/TX buffers allow MCU-independent reception and transmission. For increased versatility in communication and application debugging we added two serial ports which can be individually enabled on demand and extended via USB. The reason for actually providing a RS232 interface was not only the additional debug/communication channel but the possibility to easily attach ready-made devices like GPS modules to *SNoW*^{5} (not yet available as expansion boards). Consequently it is no problem to use a single node as gateway from a PC to the wireless network. A JTAG port allows convenient programming and in-circuit-debugging of the applied MSP430 MCU.

Although cooperation between the nodes of a WSN accounts for its overall performance, we decided to facilitate *complete autonomous and network independent operation*. *SNoW*^{5} is able to compute or collect environmental data over a long period without

communicating with other nodes. To store even a large amount of information (logs, measured values, etc.) the non-volatile data flash Atmel AT45DB161B [4] is available on board. This byte addressable 16 Mbit memory also holds the node's basic configuration like its unique ID and communication parameters. The reason for selecting this very device were the two 528 byte data buffers allowing a very smart implementation of an embedded file system [5] that saves valuable RAM within the MSP430 MCU and has some other interesting features.

Finally, the *small but powerful operating system YaOS* was designed to handle all attached devices and to provide a simple to use interface for the application running.

Sensor Node	**Mica2 MPR410CB** [6]	**BTnode** [7]	**ESB ScatterWeb** [8]	**EYES** [9]	**Telos** [10]	***SNoW*^{5}**
Developer	Crossbow	ETH Zurich	FU Berlin	Univ. of Twente	UC Berkeley	Univ. of Wuerzburg
Date	2002	2004	2005	2003	2004	2005
Microcontroller unit						
IC	ATMega128L	ATMega128L	MSP430F149	MSP430F149	MSP430F1611	MSP430F1611 / F16xx
Speed (MHz)	7.37	7.37	?	5	0.4 - 8	0.4 - 8
Architecture	8 bit RISC	8 bit RISC	16 bit RISC	16 bit RISC	16 bit RISC	16 bit RISC
Flash ROM / RAM (kB)	128 / 4	128 / 4	60 / 2	60 / 2	48 / 10	48 / 10
Power, active(mA) / sleep(µA)	8 / 15	8 / 15	3.2 / 1.6	3.2 / 1.6	4 / 2	4 / 2
Wakeup time (µs)	180	180	6	6	6	6
Onboard memory						
IC	AT45DB041B	62S2048U	MC 24LC64	ST M25P40	ST M25P80	AT45DB161B
Type / Interface	Flash / SPI	SRAM / ?	EEPROM / I^{2}C	Flash / SPI	Flash / SPI	Flash / SPI
Non-volatile	yes	no	yes	yes	yes	yes
Size (kB)	512	240	64	512	1024	2048
Power, idle (µW)	5	?	0.03	150	150	5
Power, read / write (mW)	10 / 37.5	? / ?	0.15 / 0.3	12 / 45	12 / 45	10 / 37.5
Primary wireless communication						
IC	CC1000	CC1000	TR1001	TR1001	CC2420	CC1100
Interface	SPI	SPI	non-SPI	non-SPI	SPI	SPI
Data rate (kbit/s)	38.4	38.4	19.2	57.6	250	500
Modulation	FSK	FSK	OOK,ASK	OOK,ASK	O-QPSK	2FSK,GFSK,ASK,OOK,MSK,QPSK
Frequency (MHz)	433	433-915	868	868	2400	315, 433, 868, 915
HW addr. check, dig. RSSI/LQI	no	no	no	no	yes	yes
RX / TX @ 0 dBm (mA)	7.4 / 10.4	7.4 / 10.4	3.8 / 12	3.8 / 12	18.8 / 17.4	14.5 / 16.1
Low power RX / sleep (µA)	74 / 0.2	74 / 0.2	1800 / 0.7	1800 / 0.7	– / 1	15 / 0.4
Interfaces / Sensors / Misc						
PC Communication	RS232	Bluetooth / JTAG	RS232 / JTAG	RS232 / JTAG	USB	RS232 / JTAG
Extension pins / DC ports	51 / 1	55 / 1	24 / 1	14 / 1	16 / 1	67 / 1+2 (free for expansion)
Digital I/O / ADC / DAC	? / ? / 0	21 / 2 / 0	8 / 0 / 0	8 / 8 / 0	13 / 6 / 2	41 / 8 / 2
Accessible buses	SPI, I^{2}C	SPI, I^{2}C	SPI	–	SPI, I^{2}C	SPI, I^{2}C
Overall DC / physical specifications						
Size (mm × mm)	32 × 58	32 × 58	≈ 45 × 54	≈ 32 × 92	32 × 65	50 × 85
Supported operation voltage (V)	2.7 - 3.3	3.3 or 3.8 - 5	3 - 26	3	1.8 - 3.6	1.8 - 20
Regulated supply	no	yes	yes	no	no	yes
Power, active mode (mA)	30	≈ 33	12	?	14	8

Table 1. Node comparison table

3 Applications

SNoW^5 pursues two basic concepts: due to its small, customizable and energy efficient design paired with various communication channels it is ideal for various demanding applications. In addition, modularity and easy debugging enables our node for research and education. So, recommended fields of application are role-based scenarios where differently equipped nodes cover distinct areas of a comprising complex task.

One example is the supervision of territories and buildings for security, informational and controlling aspects. This is even possible in dangerous and misanthropical environments where no communication infrastructure is available and ad hoc networking is mandatory. Another reason for using *SNoW*^{5} is its easy adaptability to the requirements given. This is especially interesting for task forces like firefighters employing a

node in a large variety of situations. Here it can be used for measuring gas concentrations, locating persons and finding escape routes.

Its profits for research and education is the easy accessibility via serial interfaces and JTAG from a workstation. This allows development and analysis of embedded software/middleware like distributed and low power algorithms, operating systems and communication protocols. Functionality of sensors and actors can also be explored.

4 Conclusion and future work

In this technical report we have initially defined our demands on nodes of a WSN. According to these requirements we developed the *SNoW*^{5} node, whose architecture and features were outlined. Differences to other nodes were shown as detailed table. Some examples of particularly suitable applications for *SNoW*^{5} in commercial and research areas close this technical report.

The successful establishment of a wireless sensor network using *SNoW*^{5} finally enables us to evaluate theoretical assumptions under hard real-world conditions. Thus we are currently researching on theoretical problems like self-organizing and fault-tolerant systems. Underlying aspects will be network protocols, routing, time synchronization, power saving concepts, localization and embedded systems software design. Future work will also lead to several daughter boards for sensors and actors in addition to those mentioned above. We will also look for hardware improvements and miniaturization of the *SNoW*^{5} main board. Our long-term objective is the specification of hardware and software requirements leading to a mass market SOC design for generic WSN nodes.

References

[1] TEXAS INSTRUMENTS INC., Dallas (USA): *MSP430x161x Mixed Signal Microcontroller (Rev. D)*, 2005.

[2] CHIPCON AS, Oslo (Norway): *CC1100 Data Sheet*, 2005.

[3] KONRAD LORINCZ, MATT WELSH: *A Robust, Decentralized Approach to RF-Based Location Tracking*. Technical Report TR-04-04, Harvard University, 2004.

[4] ATMEL CORP., San Jose (USA): *AT45DB161B Data Sheet*, 2004.

[5] LUTTER, GERALD: *An AT54DB161B file system for YaOS*. Technical report, Department of Computer Engineering, University of Wuerzburg, Germany, 2006.

[6] CROSSBOW TECHNOLOGY INC., San Jose (USA): *MICA2 Wireless Measurement System*, 2005.

[7] BEUTEL, J., O. KASTEN, F. MATTERN, K. RÖMER, F. SIEGEMUND, and L. THIELE: *Prototyping wireless sensor network applications with BTnodes*. In *Proc. 1st European Workshop on Sensor Networks (EWSN 2004)*, volume 2920 of *Lecture Notes in Computer Science*, pages 323–338. Springer, Berlin, January 2004.

[8] *FU Berlin: ScatterWeb - ESB*, 2006.

[9] HOESEL L.F.W., DULMAN S.O., HAVINGA P.J.M. KIP H.J. VAN: *Design of a low-power testbed for wireless sensor networks and verification*, 2003.

[10] JOSEPH POLASTRE, ROBERT SZEWCZYK, DAVID CULLER: *Telos: Enabling ultra-low power wireless research*. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, 25.-27. April 2005.

Energy Measurements for MicaZ Node

Marc Krämer, Alexander Geraldy

University of Kaiserslautern, Germany
{kraemer,geraldy}@informatik.uni-kl.de

Abstract. Energy consumption and power supply are important aspects of mobile devices. To determine the operation time, knowledge about the energy consumption of the mobile device and the capacity of the battery are needed. By using energy saving techniques and graceful degradation, we can extend the life time significantly. To use these techniques efficiently with respect to response time and functions of the device, measurements on the device and the battery are required.

1 Introduction

Mobile devices are often powered by batteries. Manufacturer of rechargeable batteries specify the capacity of the cell, which is determined using measurements on a certain load. For lithium polymer batteries this load is often relatively high compared to the consumption of a small device. The usable capacity of a battery depends mainly on the minimum operational voltage needed by a device, the current and the temperature. To determine the runtime of a mobile device, there are two things necessary: the useable capacity of the battery and the energy consumption of the mobile device. Since the device can be in idle state, computing, transmitting or in some energy saving state, its energy consumption varies.

If the device is never idle, it is hard to save energy by using the standard saving techniques, i.e. switching to energy saving mode in idle period. Even if these modes should be used, the problem is that the microcontroller of the MicaZ provides six energy save modes to choose from. We propose a graceful degradation of functionality if the battery is running low. Graceful degradation is the degradation of a system in such a manner that it continues to operate, but provides a reduced level of service rather than failing. For example consider a device with some core functions and additional functions, that works for some time t. If the battery is getting low, the additional functions are degraded, which means they are not switched off abruptly, but their runtime is shortened and finally set to zero. Even the core functions may have parts that can still work with less runtime or the execution frequency can be degraded. During the new idle periods it is now possible to switch to sleep modes and save energy. To determine when the battery is at a low level, measurements for the used battery are required.

Switching to save modes has some side-effects, i.e. stopping the CPU, stopping of timers or discarding external events. Therefore the requirements of the

application running on the device need to be considered as well as the consumed energy.

2 Related Work

For the Mica2 node, an earlier version of sensor nodes from Crossbow, there exist a few measurements [1] and an integration of these measurements into AVRORA [2], a simulator for ATMEL AVR microcontrollers. Since the Mica2 node uses a different transceiver than the MicaZ node, at least measurements of the new transceiver are required. Since the redesign of the node for the new transceiver can have influence on the energy consumption, the other components have to be measured too.

Lithium polymer batteries are often used for high current purposes, so there exist measurements with high loads (some amperes). We have only low current (a few mili-amperes), so we cannot rely on the specifications of the manufacturer of the batteries.

3 Measurements of a Lithium Polymer (LiPo) Battery

For most applications, it is necessary to know when the battery capacity is below a certain level. If the battery is below that level, a warning can be given or a controlled shutdown of the device can be made. We chose the LiPo batteries, because mobile devices should have low weight energy source with high energy density. For the used LiPo batteries there exist only a few discharging curves and most of them use a discharge current at a level multiple of the capacity [3]. The reason for this is, that they are often used in model aircrafts, where high current is needed. In our case, where no appropriate curve is available, we have to make the measurements on our own.

3.1 Experiment Setup

For the experiment, we use a lithium polymer rechargeable battery from Kokam [4] with a nominal capacity of 1.5 Ah and a nominal voltage of 3.7 V. We did the discharge at a constant current of 37.8 mA which is ensured by an added DC/DC regulator and a constant resistor. The voltage of the battery, as well as the output voltage of the DC/DC regulator are recorded by a digital oscilloscope [5].

3.2 Experiment Results

The result of the experiment is shown in Figure 1. On the x-axis, the time is in units of 30 seconds, on the y-axis the voltage of the battery and the output of the DC/DC regulator is drawn. The curve can be divided in four parts. The first section, till 250 time units, is decreasing slightly faster than the middle section.

In both sections, the relationship between voltage and time is nearly linear. Only in the last section, at about 3250 time units, the voltage is roughly constant for a small amount of time and afterwards decreasing really fast to the shutdown point of the voltage regulator at $t = 3816$.

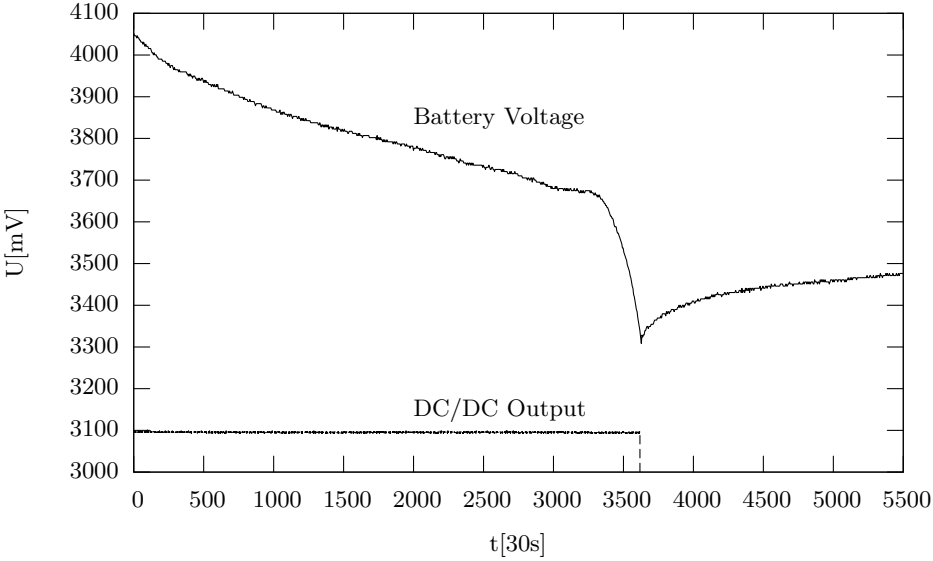


Fig. 1. Battery discharge at 37.8 mA

The useable capacity can be calculated to $\frac{3816 \cdot 30\text{s}}{3600\frac{\text{s}}{\text{h}}} \cdot 37.8\,\text{mA} = 1202\,\text{mAh}$. The important result of this experiment are the voltage changes at discharge. Due to the correlation between current battery voltage and the capacity left in the battery it is sufficient to measure the current voltage. This fact is interesting, because the voltage can be determined by the MicaZ using its built-in AD converter.

4 Measurements of the MicaZ Node

To determine the operating time of the MicaZ node [6], exact measurements on its energy consumption are needed. The MicaZ node consists of a microcontroller [7], a transceiver (transmitter/receiver) [8], three light emitting diodes (LED) and a few other components that are not relevant here. All components can be switched on, off or have some energy saving mechanisms. For most of these electrical components, we can find the energy consumption in their data sheets. The sum of all these ratings may differ from the real value, due to additional components like resistors etc. Another problem is that for some functions,

e.g. sleep modes, no power consumption is given. In the master's thesis of Dominik Domis [9], measurements of another node called *particle* [10] show that there can be a big gap between the data presented in the data sheet and the measured value. According to this, measurements of the microcontroller in different operation modes, the transceiver and the LEDs were performed. Since the node is considered as a whole inseparable node, the measurements were taken with the whole node. To get better results, each unused component in a specific measurement is put into its highest energy save mode.

4.1 Experiment Setup

For this experiment, we use the same DC/DC regulator as in the prior experiment to reduce the voltage to 3.1 V for the MicaZ. At the output of the regulator we add the MicaZ. For the measurement, the current of the MicaZ and the regulator as well as the voltage of the battery are recorded by a digital oscilloscope [5].

4.2 Experiment Results

For each of these states, we did measurements of the energy consumption of the MicaZ. Since the MicaZ needs to be considered as a whole, the consumption of the node with all components shutdown is the minimum energy consumption of the node, even if no program is running on it. We perform our measurements including the DC/DC regulator, because it is needed for a proper functionality of the node.

Microcontroller
The microcontroller of the MicaZ is an ATMEL ATMega 128L, which has various functions to save energy. The obvious ones are the six different energy saving modes. Another one is to decrease the operation frequency by the built-in frequency scaler. Figure 2 shows the different save modes and their consumption. Before entering any of these modes, the program waits a while to stabilize the consumption, which can be seen in the figure.

The first three measurements concern normal operation mode. Two of them represent duty cycles, while the third one is an idle loop. Both busy loops have a high consumption, while the NOP loop has a lower consumption of 1 mA. All other measurements show energy saving modes. The idle mode has the lowest savings, but has the advantage that almost all parts of the microcontroller still work. All other modes, save more energy, but have side-effects resulting from disabling more parts of the microcontroller. In this measurement, the standby and the powerdown mode have not been considered. These two modes stop the main clock, so a wake up is not possible after a defined time, which is needed for our future work. Only an external interrupt can wake up the device. In Table 1, the average consumption of the microcontroller in every mode is listed. Details on different modes are given in the data sheet [7, p. 44].

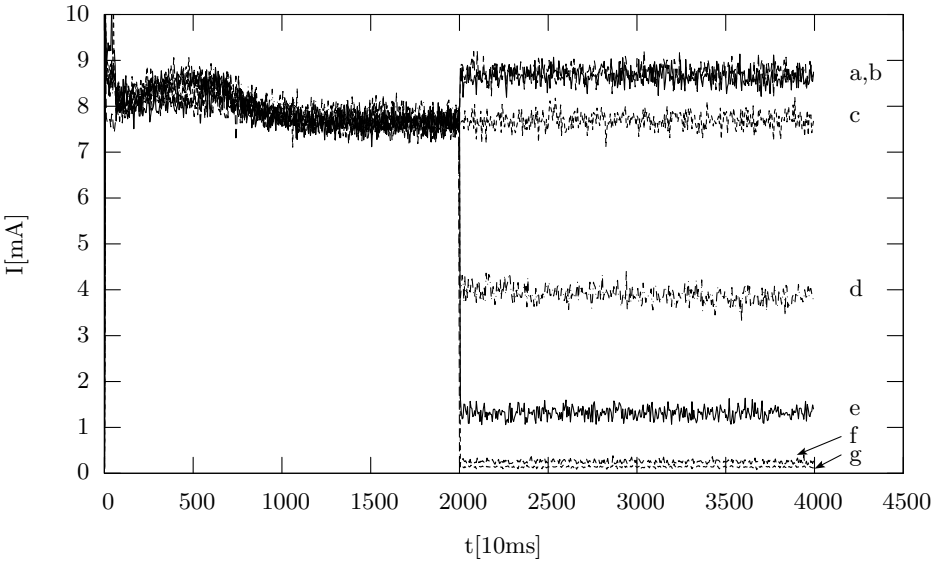


Fig. 2. Energy save modes of ATMega 128L – a:Busy (mul), b:Busy (jmp), c:NOP, d:Idle, e:ADC, f:Ext. Standby, g:Save

The second way for saving energy is decreasing the internal operation frequency. If the next task should be executed at time t, the frequency can be decreased to a level where the current task ends at time t. Therefore no waiting or switch to a energy save mode is needed. An example for this is given in Figure 3. By measurement we determined that the consumption at $\frac{\text{speed}}{32}$ is not $\frac{P}{32}$, but it is $\sim \frac{1}{8}P$. Considering this, the expected consumption in example in Figure 3 can be calculated to 13.8 J. So there is no energy saving anymore. Nevertheless, if we cannot use a sleep mode, this technique can have some benefit. Consider the CPU is waiting for some external event by using a busy loop, or timers are running, thus entering a *deep sleep mode* is not possible. In many sleep modes the CPU or the timers except timer 0 are stopped, so entering any sleep mode is impossible. In this case, one can scale down the frequency i.e. by 16 and enter the idle mode. Since we have nothing to compute and are only waiting for an event, the energy consumption for the same time of waiting is lower, without entering a save mode. By comparing both values in Table 1 one can see, that the consumption at Speed/16 in combination with idle mode is even lower than in ADC mode.

Transceiver

The transceiver is a component which has the highest energy consumption of all relevant components of the MicaZ. This component has four different states: *down*, *idle*, *send* and *receive*. According to the data sheet, the receive mode

Operation Mode	average current [mA]
Busy (mul)	8.65
Busy (jmp)	8.73
NOP	7.69
Idle	3.88
ADC	1.32
Extended Standby	0.25
Save	0.14

Speed Level	average current [mA]
1/2	4.96
1/4	3.57
1/8	2.73
1/16	1.65
1/127	0.76
1/2 + Idle mode	2.23
1/16 + Idle mode	0.84

Table 1. Average energy consumption

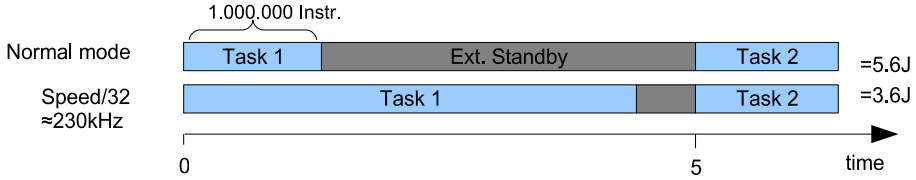


Fig. 3. Theoretical benefit of decreasing frequency on ATMega 128L

should have a higher consumption than the send mode. By measurement we determined the current in both modes to 20 mA. The idle mode has a lower power up delay and the consumption is about 0.4 mA higher than in down mode.

Figure 4 shows the mentioned measurements. For this measurement, the current of the microcontroller is not interesting and should be low to get higher precision. Therefore the frequency is set to $\frac{1}{16}$. The transceiver initializes and is set to receive mode. After this period a send operations is started, which can not be seen by measurement and finally the transceiver is automatically set back to receive mode. After the measurement the transceiver is set to down state again.

LEDs

For each of the LEDs we determined an average consumption of 10.2 mA including the processor. By subtracting the NOP consumption (7.69 mA) of the microcontroller, as stated in Table 1, the consumption of a LED is 2.5 mA.

Other Components

The other components on the MicaZ, like the serial data logger or the unique identifier, are not interesting for our research or only used once at start-up and therefore have not been analyzed.

5 Conclusion

Measurements on LiPo batteries have shown, that there is a good relation between the capacity left in the battery and the current measured voltage. There-

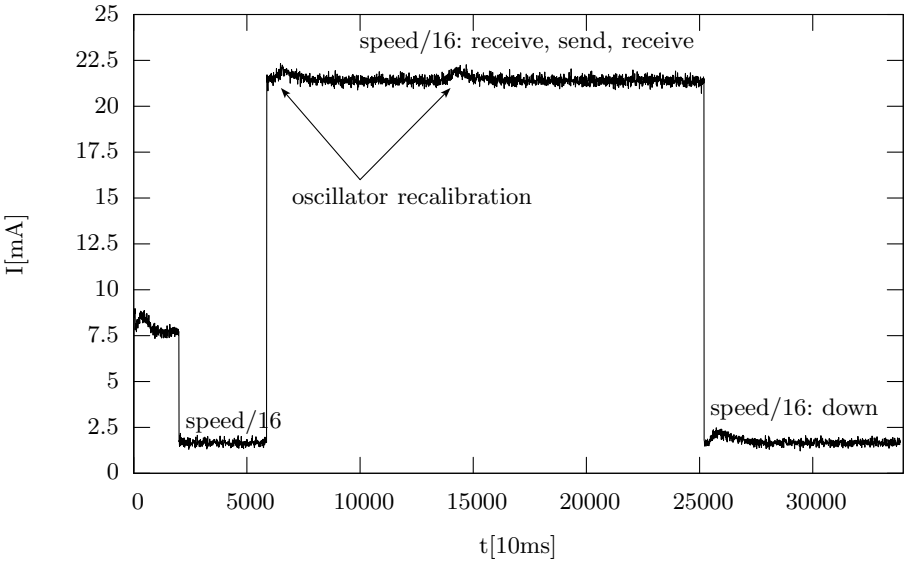


Fig. 4. Energy consumption of transceiver CC2420

fore it is easy to determine the current battery level. Hence it is now possible to achieve a graceful degradation of functionality of the device. With the measurements of the device, we can determine how much energy is consumed in each mode. Some power saving modes that seemed useful while reading the data sheet cannot be used to save energy. However, the promising method of frequency-scaling can be used for power saving whenever the sleep mode is not applicable.

Using these measurement results, we can improve the precision of the energy component of the AVRORA simulator. Thereby we can simulate energy scheduling techniques without measurement. In the next step we will develop a dynamic scheduler, which will on the one hand schedule tasks based on the current energy level, the respective power consumption of the tasks and their priorities to enable graceful degradation. On the other hand, we will be able to decrease the frequency of the microcontroller to execute the tasks in time but with minimized energy consumption.

References

[1] Olaf Landsiedel, Klaus Wehrle, Stefan Götz: Accurate Prediction of Power Consumption in Sensor Networks. In: Proceedings of The Second IEEE Workshop on Embedded Networked Sensors (EmNetS-II), Sydney, Australia (2005)

[2] AVRORA: AVRORA Homepage. `http://compilers.cs.ucla.edu/avrora/` (2006)

[3] Rinninsland, D.: Discharge curve of a LiPo battery. http://www.lipoly.de/modellbau/akkus/kokam/kokam2000hd04.gif (2006)
[4] Kokam: Kokam 603870H. http://www.kokam.com/product/product_pdf/high_power/PM-201_SLB603870H_1500mAh_Grade.pdf (2006)
[5] Meilhaus: Meilhaus Electronic DS1M12. http://www.meilhaus.de/e_me/ds1m12.htm (2006)
[6] Crossbow: Crossbow datasheet on MicaZ. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf (2006)
[7] ATMEL: ATMEL ATMega 128L datasheet. http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf (2006)
[8] Chipcon: Chipcon CC2420 datasheet. http://www.chipcon.com/files/CC2420_Data_Sheet_1_3.pdf (2005)
[9] Domis, D.: Komponentenbasierte Energiemodellierung am Beispiel eines Ambient Intelligence Systems. Master's thesis, Fachbereich Informatik, Technische Universität Kaiserslautern (2006)
[10] Particle Computer: Homepage of Particle Node. http://www.particle-computer.de (2006)

Receiver-based CDS Algorithm for Wireless Networks*

Markus Waelchli and Torsten Braun

Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 12, CH-3012 Bern
waelchli@iam.unibe.ch

Abstract. Energy savings and topology control are inherent tasks of many wireless sensor networks. Sensors are assumed to be randomly deployed and shall organize themselves independently after deployment. Moreover, sensor networks shall operate as long as possible warranting network connectivity. To deal with these tasks we propose the maintenance of a virtual backbone where a fuzzy logic control (FLC) engine is proposed to handle the local backbone access over time. Nodes not participating in the backbone shutdown their radios and go to sleep for a certain time.

1 Introduction

A distributed event detection and tracking framework intrinsically performs application specific tasks such as distributed localization, observer determination, in-network processing, etc. Additionally, such a application needs topology control and routing to supply a communication infrastructure that runs as energy efficient as possible. To support this communication infrastructure we propose the maintenance of a virtual backbone built by a connected dominating set (CDS). The CDS thereby adapts itself to local energy distributions and node conditions in the network. A fuzzy logic control (FLC) engine is proposed to handle the local backbone access over time. Nodes not participating in the backbone shutdown their radios and go to sleep for a predefined long sleep period. In this work we discuss our current approach and outline directions of future research.

The new idea of our approach is to consider multiple properties of the nodes, such as their movement pattern, energy level, and distance to known backbone nodes and make local decisions based on that information.

* The work presented in this paper was supported 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.

2 Related work

There exist a number of approaches that enable the selective disconnection of redundant nodes. ASCENT [1] and SPAN [2] both are distributed and randomized algorithms where nodes make local decisions on whether to sleep, or to join a forwarding backbone. Both schemes demand periodical information exchange among the network nodes to identify the redundant nodes that may go to sleep. Neither SPAN nor ASCENT do consider the movement patterns of the nodes or their locations and are thus not able to make advantage of this information. In GAF [3] nodes form virtual clusters where redundant nodes are timely disconnected from the network. The state of the nodes bases on global virtual positions. The main drawback of GAF is that it does not adapt to the network topology and the nodes individual energy state.

Another set of algorithms determine a connected dominating set (CDS) of a given network graph. The algorithm proposed by [4] first determines a CDS that consists of all nodes which have at least two non-adjacent neighbors. To reduce this CDS two pruning rules are introduced. In [5] the authors enhance their algorithm with the possibility of considering a node's remaining energy level instead of its link degree. In [6] a CDS is built in two phases. In a first step a maximal independent set (MIS) is constructed. In a second step a set of nodes connecting the MIS is determined. The resulting set of nodes builds a CDS. The approach of [7] converges in only one algorithmic step. Each node receiving a dominator message determines a timer based on the number of not yet covered downstream nodes. As soon as a timer expires the node enters the CDS and broadcasts a dominator message. Nodes that do not reach additional nodes cancel their timer and become dominated.

3 Receiver-based CDS algorithm

In this section we introduce our approach, discuss the current state and outline future research directions. In order to set up a virtual backbone nodes have to periodically exchange control messages with each other. The control messages may thereby be enhanced with the node's ID, link degree, position, energy level, and velocity. Based on this information, nodes make decisions like accessing the backbone or determining neighbors to take over their operation in the backbone.

3.1 Preliminaries

A dominating set (DS) of a graph $G = (V, E)$ is a subset $V' \subset V$ where each node in $V - V'$ is adjacent to some node in V'. A connected dominating set (CDS) is a dominating set which builds a connected subgraph of G. To minimize the number of set members it is desirable to find a minimum connected dominating set (MCDS) of G what is however shown to be NP-complete [8]. Moreover, the MCDS does not reflect energy distributions and need of updates within

the network. We propose a distributed approximation algorithm that aims on efficiency and network adaptivity.

The decision whether a node enters the CDS is done in a distributed manner based on a fuzzy logic control engine (FLC). The advantage of using a FLC engine is that energy distributions, node movement patterns, etc. are included in the dominator election. Thus, we hope to get a more flexible algorithm that adopts to the current network state and allows the election of the most appropriate nodes in the backbone. Including the movement patterns of the nodes we can prioritize nodes moving at slow speed, thus prematurely excluding nodes with an increased probability of affecting the backbone connectivity in the near future. To derive a node's speed pattern we need the support of location information what we assume to be satisfied by most sensor network applications as they are intrinsically location dependent.

3.2 Setting up the connected dominating set

In this section we describe the CDS algorithm. The FLC engine in not yet implemented and we discuss the CDS construction solely based on the node degree. In the future we will substitute this operations with a FLC engine.

The CDS setup is considered as a graph coloring problem. Initially, all nodes are white. The base station starts the CDS algorithm, coloring itself black and broadcasting a DOMINATOR message. This message contains the node's ID and a list of its neighbors. Thus, each node receiving the DOMINATOR message is able to check if it covers additional nodes. Each node overhearing this DOMINATOR message broadcasts a DOMINATEE message containing a copy of the DOMINATOR message. All nodes that are two hops away from the last elected dominator and overhear a DOMINATEE message compare their

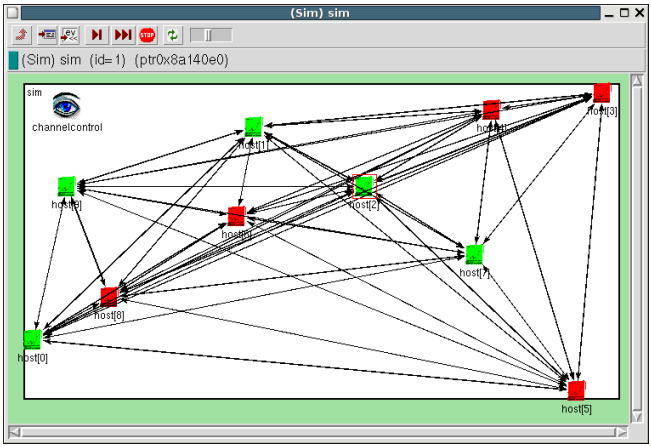


Fig. 1: Receiver-based CDS construction: Colors indicate different node states. Green nodes are dominators, red nodes dominated.

own neighborhood information with the neighborhood information of the last dominator. Based on that information they determine one of their upstream neighbors as dominator, prioritize it, and schedule a DOMINATOR_ELECTED message for it. This message is delayed according to the priority of the elected node. Nodes with only one path to the last dominator are favored and therefore get a higher priority. If multiple paths exist one of them is randomly chosen.

The algorithm terminates as soon as no white nodes remain. Our algorithm is receiver-based and an approximation of the MPR protocol [9] which is used for an efficient broadcast in the OLSR routing protocol [10]. In contrast to MPR our CDS algorithm does not depend on the knowledge of any two-hop neighborhood information. The algorithm described so far is implemented in the OMNeT++ network simulator [11]. An example is depicted in Fig. 1.

Currently we are implementing the replacement of the link based election scheme with the FLC-based election scheme. The link based dominator election seems to have a good MCDS approximation factor. However, the algorithm is inflexible in terms of energy distributions and node movement patterns. To deal with these conditions we expect good results applying a FLC engine.

3.3 Local Path adaptation

In this section we propose a local path adaptation scheme to support local backbone updates. After the CDS setup, all dominated nodes go to sleep for a given amount of time. After this sleep period a dominator may decide to maintain or abandon its work in the backbone. To ensure network connectivity, neighboring dominators that intend to release their backbone job concurrently have to negotiate who remains in the backbone and who goes to sleep. This is achieved using a timer depending on the node's priority. The priority is again derived from the FLC engine that considers the node's speed, location, node degree, etc. The dominator with the shortest delay informs its neighbors and determines the nodes that take over its dominator function in the CDS. Three cases may occur:

1. If there exists a node with a high priority that interconnects all neighboring dominators it is elected and all relevant nodes are informed about the new status.
2. If there is no node with a high priority that connects to all dominators, but there is a set of nodes that interconnect all dominators, chose the most appropriate subset and inform all affected nodes about that election.
3. If there exists at least one dominator that is no longer covered after the disconnection of the node inform the base station to reinitiate the whole CDS algorithm.

A node estimates the connectivity among neighbor nodes by knowing their coordinates. Thus, a node does not need to know its two-hop neighborhood information, which implies high overhead for many applications. A further advantage of applying fuzzy logics is the possibility to 'weaken' nodes that are close to the transmission range. As we base our local update computations on the assumption of circular transmission ranges we may run into problems when having irregular transmission ranges.

4 Conclusions and future work

One of the reasons to propose a topology control mechanism is the need of having an energy efficient backbone structure that supplies routing information for our event detection framework [12]. In the current state it seems that the CDS algorithm is performed efficiently and the MCDS approximation factor of the algorithm looks promising. In our future work we will implement the FLC engine and substitute the current link-based decision making process with it. We will furthermore analyze and evaluate the resulting simulations and refine our protocol based on these insights. To get indications on the performance of our approach we will implement other approaches and compare them to ours. Finally, we will implement the whole event detection and tracking framework, including the virtual backbone, in a real testbed and investigate the impacts and the performance under real world conditions.

References

1. Cerpa, A., Estrin, D.: Ascent: Adaptive self-configuring sensor networks topologies. IEEE Transaction on Mobile Computing **3**(3) (2004) 272 – 285
2. Chen, B., Jamieson, K., Balakrishnan, H., Morris, R.: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In: Proceedings of the 7th ACM International Conference on Mobile Computing and Networking, Rome, Italy (2001) 85–96
3. Xu, Y., Heidemann, J., Estrin, D.: Geography-informed energy conservation for ad-hoc routing. In: MobiCom '01, Rome, Italy (2001)
4. Wu, J., Li, H.: On calculating connected dominating set for efficient routing in ad hoc wireless networks. In: DIALM 99, Seattle, WA, USA (1999)
5. Wu, J., Dai, F., Gao, M., Stojmenovic, I.: On calculating power-aware connected dominating sets for efficient routing in ad hoc wireless networks. In: Journal of Communications and Networks. Volume 4. (2002)
6. Wan, P.J., Alzoubi, K.M., Frieder, O.: Distributed construction of connected dominating set in wireless ad hoc networks. In: Proceedings of Infocom 2002. (2002)
7. Zhou, D., Sun, M.T., Lai, T.H.: A timer-based protocol for connected dominating set construction in ieee 802.11 multihop mobile ad hoc networks. In: Proceedings of the 2005 Symposium on Applications and the Internet (SAINT'05). (2005)
8. Clark, B.N., Colbourn, C.J., Johnson, D.S.: Unit disk graphs. Discrete Mathematics **86** (1990) 165–177
9. Quayyum, A., Viennot, L., Laouiti, A.: Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, INRIA, Sophia Antipolis, France (2000)
10. Clausen, T., Jacquet, P.: Optimized link state routing protocol. RFC 3626 (2003)
11. Varga, A.: (Omnet++ discrete event simulation system)
12. Wälchli, M., Scheidegger, M., Braun, T.: Intensity-based event localization in wireless sensor networks. In: Proceedings of IFIP Third Annual Conference on Wireless On Demand Network Systems and Services (WONS'06), Les ménuires, France (2006)

Discovering topologies in Wireless Sensor Networks

Daniela Krüger, Carsten Buschmann, and Stefan Fischer

Institute of Telemactics, University of Lübeck, Germany
{krueger, buschmann, fischer}@itm.uni-luebeck.de,
http://www.itm.uni-luebeck.de

Abstract. Using self-organizing wireless sensor networks for object tracking requires ordering events with regard to time and location. In labyrinth-shaped topologies, one-dimensional ordering suffices within the different parts of the network. We present an algorithm to derive this ordering without the use of location information. Local order knowledge in the nodes can then be additionally used to detect junctions.

1 Introduction

An important aspect of wireless sensor networks is the logical partitioning into sub-networks. This aspect is fundamental for a lot of location-aware applications. Furthermore, applications often require an ordering of the nodes with respect to their environment, e.g. for object tracking. Computation of such location information can either be done at a central point or by the nodes themselves. The latter is beneficial in order to reduce the amount of transmitted data. Since the energy consumption decreases with decreasing amount of data transmissions, a localized algorithm helps to increase the overall lifetime of the network [1],[2]. We assume that the wireless sensor network consists of a topological structure resembling road networks or corridors in buildings and that the devices are equipped with motion detectors. In [3] a method for topology recognition is described, particularly with regard to border and junction detection, but while this paper deals with extreme high densities, we assume rather sparse networks. The problem requires detecting the one-dimensional spatial ordering of neighbors within the communication range. Since passing objects cause ordered chains of motion detection events, movement directions can be inferred by mapping object detection events to this spatial order of the nodes.

2 Our approach

This section describes our localized algorithm to detect the one-dimensional ordering of a node and its neighbors. Let N be the node computing its local topology, i.e. the ordering of its neighbors. Our algorithm comprises 3 steps: 1. Detection of border nodes, i.e. far away nodes near the maximum communication distance 2. Computation of chains from border nodes to N 3. Mapping the

remaining nodes onto the path nodes Note that N only needs information of its direct neighbors, which minimizes the communication costs. Figure 1 shows exemplarily the three steps of the algorithm.

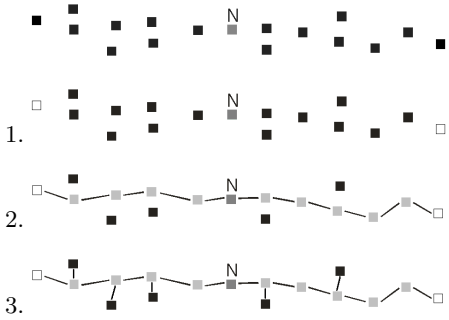


Fig. 1. Algorithm steps

First, N has to detect its border nodes. We base this detection of border nodes on the number of common neighbors. Border nodes have fewer neighbors in common with N than nodes closer to N. With an exchange of the neighborhood lists a counting of the common neighbors becomes possible. So N counts these numbers and selects border nodes using a threshold, e.g. all nodes which can communicate to less than the half of its neighbors. Second, N orders these selected border nodes into groups by estimating the distances between them. The distance estimates are also based on the number of common neighbors. Border nodes are sorted into the same group if they are close to each other, i.e. if they have many neighbors in common. For each group N determines one representative node, preferably the most distant one. Then, N computes chains from the representative nodes to itself by searching a path from the representatives to it. Third, nodes which are not within one of the computed chains will be mapped onto the nearest chain node, i.e. if a remaining node detects a motion event it is considered as if the corresponding chain node had detected it. The topology recognition depends on the quality of distance estimation. Here it is not important to get correct absolute values but correct distance relations, i.e. the estimated distance to a closer neighbor must be smaller than the estimated distance to a far away neighbor. Otherwise nodes are mapped in a wrong way. For this reason, we use an estimation method that does not rely on unreliable measurements of physical wireless communication properties. Radio interferometry features promisingly low errors but brings in specific requirements to the RF chip [4]. A disadvantage of using the number of common neighbors is that it assumes a uniform node distribution. Local differences in the node density result in failing detection of representatives as depicted in figure 2. Hence node density fluctuations must be detected and compensated.

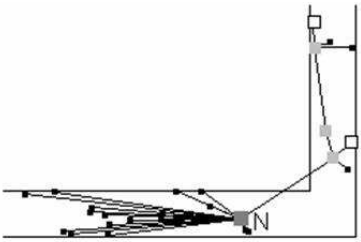


Fig. 2. Failing detection of border nodes

3 Analysis

We chose SHAWN [5] for a simulative analysis and conducted studies both, grid-based and uniform distributed arrangements. For each distribution we evaluated our algorithm with exact, random error afflicted and estimated distances. Figure 3 shows the evaluation results. The values signify the correctness of the computed chains, i.e. the percentage of correctly ordered nodes.

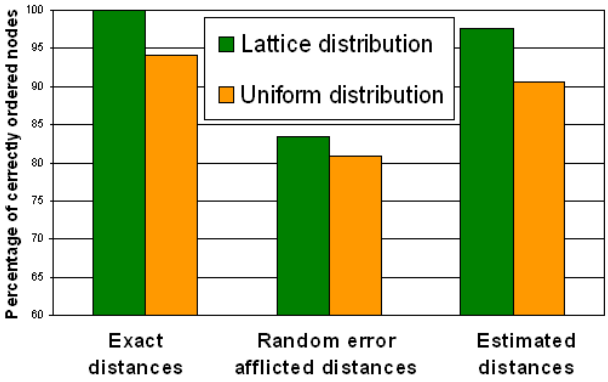


Fig. 3. 82 nodes with 12 neighbors per node on the average

The results using the estimated distances (right) are nearly as good as with the exact distances (left) although prone to estimation errors of 12% to 20% of the original distance. This dates back to the preserved distance relations in contrast to estimates afflicted with the random error of 20% (center). Here the nodes compute not only wrong paths but also wrong mappings. Furthermore, the results of the simulations based on the uniform distributions are nearly as good as the grid-based ones.

In addition, we executed simulations with different node densities. While low node density causes short chains and hence more fragile object detection and

tracking, high density causes an increased fraction of nodes mapped to chain nodes as illustrated in figure 4. This leads to less significant chains because of mapping errors.

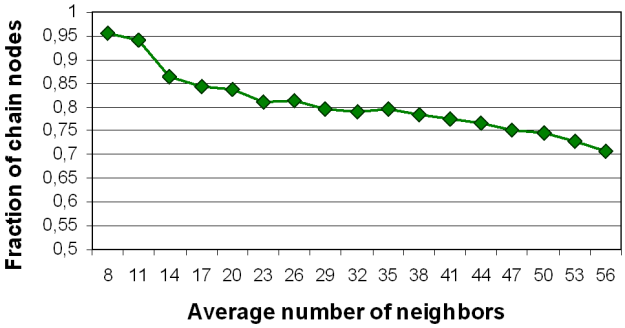


Fig. 4. 82 nodes with 12 neighbors per node on the average

This paper proposes a localized algorithm to detect the one-dimensional spatial ordering of nodes within the communication range. We have shown that it provides rather good results in spite of error afflicted distance estimates. For the algorithm it is less important to work with exact distance values. It only matters that the distance relations are preserved. Furthermore we have seen that high densities cause relatively more mappings, so it is not reasonable to increase the node density more than an appropriate quantity. Our research shows promising results for detecting network topologies in simulated environments. Future work will focus on the one hand on grouping the nodes according to the network part they belong to. On the other hand we want to evaluate the algorithm within a real world study.

4 Conclusion

This paper proposes a localized algorithm to detect the one-dimensional spatial ordering of nodes within the communication range. We have shown that it provides rather good results in spite of error afflicted distance estimates. For the algorithm it is less important to work with exact distance values. It only matters that the distance relations are preserved. Furthermore we have seen that high densities cause relatively more mappings, so it is not reasonable to increase the node density more than an appropriate quantity. Our research shows promising results for detecting network topologies in simulated environments. Future work will focus on the one hand on grouping the nodes according to the network part they belong to. On the other hand we want to evaluate the algorithm within a real world study.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks **38**(4) (2002) 393–422
2. Mattern, F., Römer, K.: Drahtlose sensornetze. Informatik-Spektrum **26**(3) (2003) 191–194
3. Kröller, A., Fekete, S.P., Pfisterer, D., Fischer, S.: Deterministic boundary recognition and topology extraction for large sensor networks. In: Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2006). (2006)
4. Buschmann, C., Hellbrück, H., Fischer, S.: Radio model-aware distance estimation based on neighborhood comparison. In: The 4th ACM Conference on Embedded Networked Sensor Systems. (2006) under review.
5. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems, Part of the SpringSim. (2005)

Routing in Sensor Networks based on Symbolic Coordinates

Matthias Gauger, Pedro José Marrón, Marcus Handte, and Kurt Rothermel

IPVS, Universität Stuttgart, Universitätsstr. 38, 70569 Stuttgart, Germany
{gauger, marron, handte, rothermel}@informatik.uni-stuttgart.de

Abstract. The routing of messages between mobile nodes and a sensor network is an important but challenging task. In this paper we present our approach for this problem that is based on the use of symbolic coordinates and that divides the task among client and sensor nodes.

1 Introduction

Sensor networks have the potential to play an important role in ubiquitous computing scenarios as sources of environmental data and context information. A variety of such data is required by typical ubiquitous computing applications for providing their services to the user. However, the expected large number of nodes in deployed sensor networks as well as their strong resource constraints make the interaction with the mobile ubiquitous computing devices a challenging task.

One fundamental problem is the communication between the mobile client devices and the sensor nodes. The client devices should be able to address a specific area of the network and the sensor nodes should then efficiently route such request messages to the appropriate nodes. Global routing tables are not well suited to this task due to the potentially large size of the network, the limited resources of the sensor nodes and due to the frequent topology changes. Flooding the network with request messages and building routing structures on-demand as done by classic source routing approaches [1] also conflicts with scalability requirements and resource constraints.

In this paper we present our approach for routing messages between mobile client nodes and sensor networks that is based on the use of symbolic coordinates. The basic idea is to have the client nodes calculate and provide a symbolic route to the destination and then let the sensor nodes transform this symbolic route into individual node-to-node routing steps.

Symbolic coordinates – in contrast to geographic coordinates – do not represent exact geographic locations but rather certain geographic areas of different sizes. In many cases, symbolic coordinates represent areas which are directly meaningful to applications like rooms in a building or streets and buildings in a city scenario. However, symbolic coordinates by themselves do not allow to infer spatial information like the distance or the spatial relationship between two coordinates. For such tasks, a symbolic location model is required.

In the rest of this paper we first give a short overview of related work and then describe our routing algorithm, its advantages and problem fields and some solution approaches for the problems.

2 Related Work

The use of symbolic coordinates has been widely discussed in the area of ubiquitous computing and different symbolic location models have been developed [2][3][4]. Our approach is largely independent of the specific location model and its properties so that most of these location models can be used.

Using symbolic coordinates in sensor networks has received much less attention so far. The general idea has been formulated by Fekete et al. [5] who propose to automatically create clusters and organize these clusters in a weighted graph structure expressing their neighborhood relations. They argue that such graphs of symbolic coordinates should in most cases be small enough to allow the distribution and use in all nodes of the network. This provides the nodes with some abstract location awareness based on the position of their cluster in the graph. The authors describe routing as one possible application of graphs of symbolic coordinates. However, they do not elaborate on how routing to specific nodes or areas could be achieved.

One alternative routing approach quoted in many sensor network publications is geographic routing [6]. However, geographic routing can only be used when all nodes in the network know their exact geographic coordinates. The required precision of this location information rises with the node density in the network. In many cases requests also first need to be mapped to a destination coordinate with some kind of location service before sending out messages.

3 Routing with Symbolic Coordinates

Assumptions We build upon a small set of assumptions concerning the system model. First, we assume that there is a symbolic location model available in the system that allows determining symbolic coordinates for all locations covered by the sensor network. We do not require special properties of the location model other than the availability of some kind of `neighborOf`-relationship among symbolic coordinates.

Concerning the sensor network, we only assume that the sensor nodes are able to maintain local neighborhood information without any knowledge of the global topology. Additionally, we require each sensor node to store its own symbolic coordinate. Note that aquiring this coordinate should be much simpler than determining the exact geographical coordinate making it relatively easy to, for example, assign the coordinate at deployment time. We are currently working on different methods for semi-automatically assigning symbolic coordinates to sensor nodes.

Mobile nodes are typically less resource-constrained than the nodes of a sensor network. For that reason, it is reasonable to assume that mobile nodes have

access to the symbolic location model either by storing model data on the node or by dynamically loading required information from an infrastructure. They are also able to process and use this information for querying the sensor network.

Mobile client nodes and sensor nodes share a common communication interface so that a mobile node is able to communicate with any sensor node in its direct neighborhood.

Basic concept The basic idea of our approach is to divide the task of routing messages from mobile nodes to specific areas in the sensor network among clients and sensor nodes. The mobile client nodes are responsible for the global routing task whereas the sensor nodes manage the local node-to-node routing of messages.

All sensor nodes periodically exchange beacon messages that contain their own symbolic coordinate and hop distance information to neighboring symbolic coordinates they have heard of. Based on such beacon messages received from neighbors, a sensor node fills a routing table with next-hop (and distance) information to symbolic coordinates in the neighborhood of the node's symbolic coordinate. Note that this list of neighboring coordinates isn't preconfigured but only learned from beacon messages.

When a client node wants to send a message to a specific symbolic area of the network it first has to calculate the symbolic route from its current position to the destination coordinate based on the stored symbolic location model. The client node then includes this route information in the message and passes it to an arbitrary sensor node in its neighborhood.

When a sensor node receives a message from a neighboring node it investigates the symbolic route stored in the message header. Based on its own symbolic coordinate, the node can retrieve the next symbolic coordinate the message should visit. It then queries its local routing table to retrieve the next-hop node on the route to this next-hop symbolic coordinate and forwards the message to this node.

When the message reaches the first node lying in the destination area three different message distribution semantics are possible: The message can be delivered to this node only (**area anycast**), to all nodes in the area (**area broadcast**), or to a specific node identified by a node identifier (**area unicast**). Area broadcast and area unicast can be implemented using a broadcast limited to the respective symbolic area.

Figure 1 shows an example of a query forwarded from symbolic coordinate "Room 6" to coordinate "Room 4". At the bottom it also shows as an example the local routing table of node 3.

Advantages The main advantage of our approach is the division of the routing task among client and sensor nodes with both parties contributing based on their respective strengths. Sensor nodes do not have maintain global routing information as they only have to perform local routing decisions. The amount of state a single sensor node has to manage neither depends on the size of the

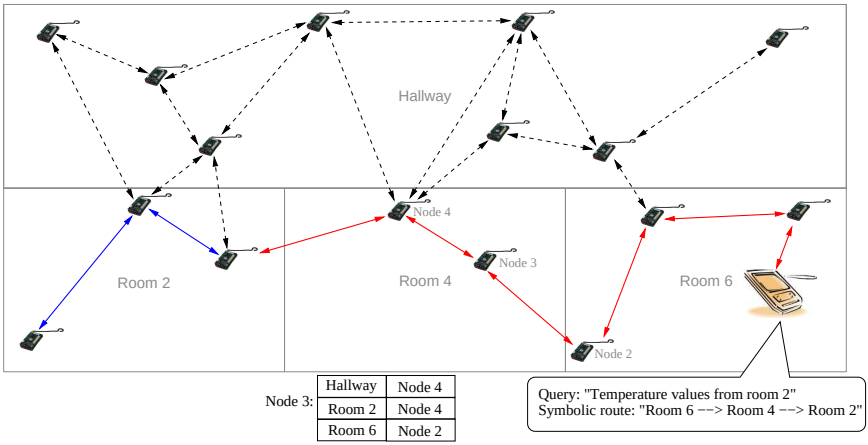


Fig. 1. Message routing example

network nor on the number of nodes in its neighborhod but only on the number of symbolic areas adjacent to the symbolic area the node resides in.

From the client's point of view, the main advantage is that the routing can be done independent of knowledge about the current sensor network topology. The failure of nodes or communication links only affects local routing within the respective area. The global routing from a client to a symbolic destination area does not change and is therefore relatively insensitive to node failures.

Problem Fields Three types of problems can prevent a successful communication when using the basic algorithm described above. First, not all neighboring symbolic areas have to be connected by sensor node communication links (**communication hole**). Secondly, complete symbolic areas might lack coverage by sensor nodes (**coverage hole**). Thirdly, the subgraph formed by the nodes inside of a symbolic coordinate might be disconnected although the complete graph is connected (**area partitionings**). In all three cases message forwarding can fail because the next-hop symbolic coordinate cannot be reached. Coverage holes and area partitionings can also prevent successful completion of area broadcasts and area unicasts once the destination coordinate is reached.

Figure 2 shows examples of all three types of problems with a communication hole between room 1 and room 2, a coverage hole in room 6 and an area partitioning in room 4.

Several ways of preventing or reacting to these problems are possible. The most simple way of preventing holes and partitionings is to assume or rather require an extremely dense topology that makes holes or partitionings extremely unlikely. Alternatively, static information about holes and partitionings might be available in the location model so that the client can plan its routes accordingly. A related method is to maintain weight values for all symbolic areas that represent the (expected) density of nodes in this area. The larger the accumu-

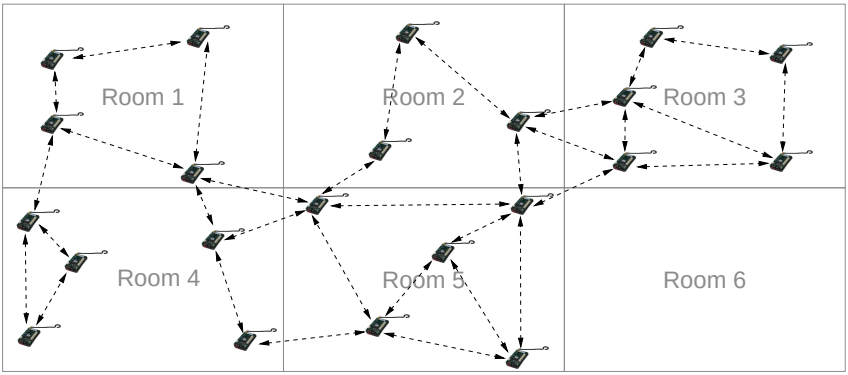


Fig. 2. Example showing communication holes, coverage holes and area partitionings

lated weight of a symbolic route the higher the probability that a message can be forwarded to the destination without getting stuck in holes and partitionings.

If none of the prevention methods works or the required information is not available then the sensor network must react to communication failures caused by holes and partitionings. We are working on two possible reactions: First, the sensor node detecting the problem can send a feedback message to the original sender of the message by following the symbolic route of the message backwards. Notified of this routing failure, the original sender is then able to resend the message specifying an alternative route. It can also buffer information about where the communication failed so that subsequent messages directly circumvent the hole. A second possible reaction for the node detecting the problem is to try to find a by-pass locally. For doing this it needs to broadcast the message to all of its neighboring symbolic coordinates which can then reinvestigate the symbolic route and either find the required symbolic next-hop in their neighbor list or broadcast the message to their neighbors. How deep such a symbolic broadcast is allowed to propagate determines both the likelihood that the original route can be taken up again but also the cost for distributing the message in multiple directions.

4 Conclusions

In this paper we described our approach for the use of symbolic coordinates in sensor networks. We concentrated on their use for a better integration of sensor networks in ubiquitous computing scenarios. We presented the basic concept showing both the advantages as well as potential problem fields. We are currently working on an in-depth evaluation of the concept including different solutions for the described problem fields and aim to improve the algorithm based on these results as part of future work.

References

1. Johnson, D.B., Maltz, D.A.: Dynamic source routing in ad hoc wireless networks. In Imielinski, Korth, eds.: Mobile Computing. Volume 353. Kluwer Academic Publishers (1996)
2. Becker, C., Dürr, F.: On location models for ubiquitous computing. Personal Ubiquitous Computing **9**(1) (2005) 20–31
3. Brumitt, B., Shafer, S.: Topological world modeling using semantic spaces. In: Workshop Proceedings of Ubicomp 2001: Location Modeling for Ubiquitous Computing. (2001)
4. Jiang, C., Steenkiste, P.: A hybrid location model with a computable location identifier for ubiquitous computing. In: UbiComp '02: Proceedings of the 4th international conference on Ubiquitous Computing, London, UK, Springer-Verlag (2002) 246–263
5. Fekete, S.P., Kröller, A., Buschmann, C., Fischer, S., Pfisterer, D.: Koordinatenfreies Lokationsbewusstsein. it - Information Technology **47** (2005) 70–78
6. Karp, B., Kung, H.T.: GPSR: greedy perimeter stateless routing for wireless networks. In: MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking, New York, NY, USA, ACM Press (2000) 243–254

Delivery Semantics for Geographic Routing

Matthias Witt and Volker Turau

Hamburg University of Technology, Institute of Telematics
Schwarzenbergstraße 95, 21073 Hamburg, Germany

Abstract. Geographic routing algorithms use locations rather than node addresses as destinations. But since it is not always guaranteed that there is indeed a node exactly at the destination location, there must exist a description which nodes shall be allowed to consume a message. In some cases, the message must not be consumed by a node at another location (e. g., if the sink is the destination), whereas in other cases, nodes in the vicinity may also consume it. This has to be chosen by the application, so the routing protocol should support different delivery semantics. Another question is if only one node may consume the message, or if multiple destination nodes are allowed.
This paper proposes three orthogonal delivery parameters: closeness, multiplicity, and accept-outside.

1 Introduction

Geographic routing is an important technique for wireless sensor networks, since it avoids the storage of routing tables or path information in the sensor nodes, which are usually highly resource-constrained. The presence of information about the node location (either through GPS or some localization algorithm) and—for some algorithms—the direct neighbors is sufficient.

Several geographic routing algorithms for sensor networks have been proposed. They use different, but elaborated techniques to forward packets using location information. One important question, however, that is often neglected is to describe the conditions under which a node is allowed to declare itself as destination and consume the message. Since the destination is not a node address but a location, this is substantially different from traditional routing schemes. Many algorithms assume that there exists a node exactly at the destination location [1, 2], the node locations being published for instance by a distributed location service like GLS [3]. The location service, however, imposes an extra overhead. Besides, the destination locations may be incorrect due to location errors [4]. Additionally, nodes may fail or move, so it is not always possible to keep track of the current state. Other algorithms perform a limited flooding in the vicinity of the destination [5], which also requires the sending of many packets. An interesting alternative is utilized by GHT [6], which takes advantage of GPSR's perimeter mode: The destination location is traversed using the right-hand rule; when the message has looped around the location, the first node that receives the message for the second time in perimeter mode declares itself as destination and consumes the message.

Although there are different strategies for finding a suitable destination node, each algorithm uses only a single strategy. It is, however, dependent on the specific application if, for example, the destination's exact location is known by the sender or if the message should be delivered "somewhere" in the vicinity of a specific location. In some cases, it is important that *at most* one node consumes the message, whereas in other cases it is more important that *at least* one destination node is found at all.

Anycast is a well-known technique for routing messages to any out of several possible destinations. Within the scope of sensor networks, anycast can be used when multiple sinks exist; messages are routed to the nearest sink [7–9]. However, anycast routing schemes do not use geographic routing and consequently are highly application-specific.

Other algorithms include time in delivery semantics. *Mobicast* [10] is a spatiotemporal multicast scheme that supports moving destination areas. Since the semantics proposed in this paper are not dependent on time, such schemes are not covered.

This paper provides a taxonomy of delivery semantics and proposes three parameters, which can be combined independently. An application that sends messages can set these parameters according to the specific requirements. A concrete routing algorithm must then deliver the message with respect to the desired semantics. The proposed delivery semantics are independent from concrete routing algorithms. The implementation of the semantics is beyond the scope of this paper.

2 Delivery Semantics

Basically, there are three questions that have to be answered when describing the geographical destination for a message in a sensor network:

1. How close to the destination must a sensor node be in order to consume the message?
2. Is it acceptable that multiple nodes consume the message, or should only one node consume it?
3. When the message gets stuck because there is no node that is close enough to the destination, should the node where the message got stuck drop the message or consume it if the node is sufficiently close to the destination?

In the following, these questions are discussed in detail.

2.1 Closeness

Assume a sensor network for environmental monitoring, where neither the user nor the base station know the exact positions of the nodes. The user requests the sensor data for a specific location and initiates a query, having this location as destination position. However, he does not know whether there is a sensor node exactly at this location. Moreover, this is extremely improbable. Hence, it

should be acceptable that a node consumes the message if its distance to the destination location lies within a designated limit of tolerance t. Using Euclidean metrics, the destination area is a circle with radius t. This approach raises the probability that a destination node is found. The limit can be set depending on the network density. Assume that the network consists of n nodes spread uniformly in a field of surface area A. Then the probability that the destination area is empty is

$$\left(1 - \frac{\pi t^2}{A}\right)^n,$$

not regarding border effects. If this probability should be less than p, t has to be set to a value such that

$$t > \sqrt{\frac{A}{\pi}\left(1 - p^{\frac{1}{n}}\right)}.$$

Consider, on the other hand, the case that the measurement message is sent from the sensor node to the sink that initiated the query. The message is explicitly addressed to the sink, so it is known that there is a node at the designated location. In this case, the limit of tolerance can be set to zero; this also ensures that no other node than the sink consumes the message.

These examples show that at least two different semantics are needed for describing destination locations: *exact* and *nearby*. In the latter case, a limit of tolerance has to be given, in the former one, it is zero.

There may be cases in which a circle is not appropriate as destination area, e. g., when the nodes are distributed in a regular mesh.

2.2 Multiplicity

If the *nearby* semantics is used, there may be more than one potential destination node. The routing protocol may take care that only one of these nodes consumes the message; however, this leads to more overhead, since the nodes must somehow agree about the winner, which comes at additional communication costs. But this is not necessary in cases where it is not important that only one node receives the message. For example, if the sensor node at (or nearby) a specific location shall be set in an alarming mode, because the user wants to have special attention to this location, it is not crucial that only one node receives the message. To account for node failures, it is even better if multiple nodes receive it.

On the other hand, there are cases where only one node shall receive the message. In general, this is the case when the nodes serve different tasks. For instance, when the message is addressed to the sink, no other node is allowed to consume it.

Inspired by RPC semantics, the following semantics are possible for the multiplicity of the destination:

maybe: the message may reach a node or not,
exactly-one: the message must be consumed by exactly one node,

at-most-one: the message must be consumed by zero or one node,
at-least-one: the message must be consumed by one or more nodes,
all: the message must be consumed by all nodes meeting the closeness semantics.

Note that these semantics are not meant for describing *how many times* a node receives the message. This aspect of delivery semantics is orthogonal to the ones discussed here. The semantics specify *how many nodes* receive the message.

2.3 Accept-outside

Consider the case that there is no node that matches the destination description, but the message has reached a node that is already near the destination. The destination location is within the transmission range, the current node does not yet match the destination description, but no closer node can be found, either because closer nodes do not exist or because they are currently not reachable. Now, there are two possibilities: Either the message is regarded as non-deliverable and dropped, or the node where the message got stuck declares itself as destination and consumes the message. The strategy to be applied depends on what has higher priority: that a node which exactly matches the destination description receives the message, or that actually any node receives it at all. For example, if data at a specific location shall be measured, it may be tolerable if merely data near this location are measured; however, if the sink is the destination, it is not acceptable when another node consumes the message. This demands for a Boolean parameter *accept-outside*.

This parameter can be used in combination with *exact* semantics to deliver the message to the node that is closest to a location. This is not possible with *nearby* semantics, since in this mode the message is consumed by any node within the area of tolerance, not necessarily by the closest one.

Not all combinations of these parameters seem to be useful. For instance, in *exact* semantics with *accept-outside=false*, the semantics *at-least-one* and *exactly-one* seem to yield the same results (when there are not several nodes with identical positions). However, the routing protocol may act differently; for example, it may deliver the messages faster or send fewer packets in *at-least-one* semantics. Admittedly, the concrete implementation of the semantics is beyond the scope of this paper.

An important issue is that for some semantics it is not guaranteed that the same node(s) consume(s) subsequent messages sent to the same location. The messages may be forwarded on different paths and hence reach different nodes. Only using *exact* semantics or *all* semantics, with *accept-outside=false* in both cases, it can be guaranteed that all messages are consumed by the same node(s).

3 Conclusion

For geographic routing algorithms it is essential to include a clear description of the destination. Different applications have different requirements regarding the node or nodes that shall consume the messages. Therefore, the routing protocol must support different delivery semantics. In this paper, three orthogonal parameters have been proposed: *closeness*, *multiplicity*, and *accept-outside*.

Implementations of these semantics are currently investigated by extending selected geographic routing algorithms.

References

1. Karp, B., Kung, H.T.: GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, Massachusetts (2000) 243–254
2. Füßler, H., Widmer, J., Mauve, M., Hartenstein, H.: A Novel Forwarding Paradigm for Position-Based Routing (with Implicit Addressing). In: IEEE Computer Comm. Workshop (CCW 2003), Dana Point, California (2003) 194–200
3. Li, J., Jannotti, J., De Couto, D., Karger, D., Morris, R.: A Scalable Location Service for Geographic Ad Hoc Routing. In: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking, Boston, Massachusetts (2000) 120–130
4. Witt, M., Turau, V.: The Impact of Location Errors on Geographic Routing in Sensor Networks. In: Proc. International Conference on Wireless and Mobile Communications (ICWMC'06), Bucharest, Romania (2006)
5. Heissenbüttel, M., Braun, T., Bernoulli, T., Wälchli, M.: BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. Elsevier's Computer Comm. Journal **27**(11) (2004) 1076–1086
6. Ratnasamy, S., Karp, B., Shenker, S., Estrin, D., Govindan, R., Yin, L., Yu, F.: Data-Centric Storage in Sensornets with GHT, A Geographic Hash Table. Mobile Networks and Applications **8**(4) (2003) 427–442
7. Intanagonwiwat, C., De Lucia, D.: The Sink-based Anycast Routing Protocol for Ad Hoc Wireless Sensor Networks. Technical Report 99-698, Computer Science Department, University of Southern California (1999)
8. Thepvilojanapong, N., Tobe, Y., Sezaki, K.: HAR: Hierarchy-Based Anycast Routing Protocol for Wireless Sensor Networks. In: Proc. 2005 IEEE/IPSJ Int. Symposium on Applications and the Internet (SAINT'05), Trento, Italy (2005) 204–212
9. Hou, Y.T., Shi, Y., Sherali, H.D.: Optimal Base Station Selection for Anycast Routing in Wireless Sensor Networks. IEEE Trans. on Vehicular Technology **55**(3) (2006)
10. Huang, Q., Lu, C., Roman, G.C.: Spatiotemporal Multicast in Sensor Networks. In: Proc. 1st Int. Conference on Embedded Networked Sensor Systems (SenSys'03), Los Angeles, California (2003) 205–217

Bestimmung Optimaler Startwerte zur Exakten Lokalisierung mittels Geodätischer Ausgleichung

Alexander Born[1], Frank Reichenbach[2], Ralf Bill[1] und Dirk Timmermann[2]

[1] Universität Rostock
Professur für Geodäsie und Geoinformatik
{alexander.born,ralf.bill}@uni-rostock.de
[2] Universität Rostock
Institut für Angewandte Mikroelektronik und Datentechnik
{frank.reichenbach,dirk.timmermann}@uni-rostock.de

Zusammenfassung Knoten in einem Sensornetzwerk werden in den meisten Fällen zufällig über einem Gebiet verteilt. Um Messungen räumlich zuweisen zu können, muss jeder Knoten in der Lage sein, seine Position zu bestimmen. Aufgrund der limitierten Ressourcen der Knoten werden energieeffiziente Lokalisierungsalgorithmen benötigt.
In diesem Paper werden Ergebnisse zur geodätischen Ausgleichung von Sensorknotenpositionen mit Näherungskoordinaten als Eingangswerte aus approximativen Lokalisierungsmethoden präsentiert. Weiterhin wurde die Berechnung der Koordinaten so verteilt, dass die komplexeren Rechenschritte von den ressourcenreichen Beacons übernommen und so die Sensorknoten vollständig entlastet werden.

1 Einleitung

Sensornetzwerke bestehen aus bis zu hunderten oder tausenden kleiner Sensorknoten, welche über einem Gebiet oder innerhalb eines Objektes von Interesse platziert werden. Ihre Aufgabe besteht im Sammeln und Weiterleiten bestimmter Informationen, z.B. Temperatur, Luft- oder Bodenfeuchte, etc. Diese Daten werden in der Datensenke gesammelt und, z.B. über das Internet, an den Benutzer weitergeleitet. Einen umfangreichen Einblick in die Thematik der Sensornetzwerke gibt [1].
Durch die Entwicklung immer kleiner werdender Sensorknoten sind die Dimensionen des Kommunikationsmoduls und der Batterie besonders kritisch. Folglich ist die zu Verfügung stehende Energie die knappste Ressource [2]. Aus diesem Grunde benötigen Sensorknoten energiesparende Bauelemente und energieeffiziente Algorithmen.
Die mehr oder weniger zufällige Verteilung von Sensorknoten durch die Art der Ausbringung (z.B. mit dem Flugzeug) verhindert eine sofortige Zuordnung des Messdatums zum Ort der Messung. Aus diesem Grund ist eine Positionsbestimmung der Knoten notwendig, die zusätzliche Energie für Datenübertragungen und Berechnungen auf den einfachen Knoten benötigt.

2 Positionierung

Ein Lokationsbewusstsein des Sensorknotens ist erforderlich, da ein Messdatum ohne dazugehörige Position nahezu nutzlos ist. Selbstkonfiguration und Selbstheilung ermöglichen ein robustes und skalierbares Sensornetzwerk. Mit Standortinformationen sind diese Funktionen ohne hohen Aufwand zu erreichen. Ein energiearmes und effizientes Routen der Informationen ist eine der Grundvoraussetzungen für drahtlose Sensornetzwerke. Standortinformationen der Knoten ermöglichen gegenüber klassischen Routingmethoden ein energiearmes, räumliches Routing. Weiterhin ist in einigen Sensornetzwerken die Position des einzelnen Sensorknotens selbst die gesuchte Kenngröße, die es zu ermitteln gilt. Denkbare Positionierungsmethoden sind einerseits satellitenbasierte Positionierungssysteme wie das Global Positioning System (GPS), zukünftig GALILEO oder funknetzbasierte Dienste wie das Global System for Mobile Communication (GSM) [3],[4],[5]. Durch ihre Größe, die hohen finanziellen Kosten und den hohen Energiebedarf sind diese Systeme nur bedingt für die Positionierung geeignet. Denkbar wäre der Einsatz auf wenigen, ressourcenstärkeren Sensorknoten, im Folgenden Beacons genannt.

2.1 Klassifikation

Die Positionsbestimmung in Sensornetzwerken wird in zwei Teilbereiche unterschieden: die approximativen und die exakten Lokalisierungsmethoden.
Die approximativen Verfahren zeichnen sich durch einfache und schnelle Berechnungen sowie einen geringen Kommunikationsaufwand aus. Der Nachteil liegt hier allerdings in den relativ großen Positionierungsfehlern. Aufbauend auf der Schwerpunktbildung [6], durch Einführung von Zwangsbedingungen [7], durch Flächenüberlagerung [8] oder durch Einführung lokaler Koordinatensysteme [9] nutzen diese Methoden zumeist nur die Koordinaten von bekannten Knoten zur Positionsbestimmung, gelegentlich jedoch auch Distanz- und Winkelmessungen. Im Gegensatz dazu erreichen die exakten Lokalisierungsmethoden durch Anwendung aller vorliegenden Beobachtungselemente eine sehr hohe Genauigkeit. Allerdings sind diese mit einem sehr hohen Aufwand für die Berechnung und Kommunikation verbunden. Zudem werden hohe Anforderungen an die Qualität der Messgrößen, z.B. Signalempfangsstärken (RSSI-Werte) oder Signallaufzeitmessungen, gestellt, die sich in der Praxis, z.B. wegen Signalreflexionen an Wänden etc., oftmals als fehlerhaft bzw. verrauscht erweisen.

2.2 Geodätische Ausgleichung

Für die Koordinatenbestimmung eines unbekannten Sensors $P(x, y)$ in der Ebene werden mindestens drei Beacons benötigt. Voraussetzung dafür sind allerdings exakte Distanzbestimmungen, die in der Realität nicht vorliegen. Da ein Sensornetzwerk aus einer hohen Anzahl von Knoten besteht, können die daraus resultierenden Distanzen zwischen dem Unbekannten und den Nachbarn für

eine Ausgleichung verwendet werden. Dadurch können die Genauigkeit und Zuverlässigkeit der Ergebnisse gesteigert werden. Den n Beobachtungen stehen damit u gesuchte Größen gegenüber, wobei $n \geq u$ gilt. Die sich aus zufälligen Messfehlern ergebenden Widersprüche werden nach der Methode der kleinsten Quadrate derart verteilt, dass plausible und eindeutige Werte für die Beobachtungen und daraus abgeleitete Unbekannte folgen. Stochastisch spricht man von sogenannten „bestlinear unbiased estimators“ (BLUE).
Bei Einführung von u Unbekannten ergibt sich das lineare Beobachtungsgleichungssystem

$$\hat{l} = l + v = A\hat{x} \tag{1}$$

mit

$\hat{l}$ - ausgeglichene Beobachtungen,
l - fehlerbehaftete Beobachtungen,
$\hat{x}$ - ausgeglichene Unbekannte.

Die genaueste und widerspruchsfreie Lösung des überbestimmten Gleichungssystems kann nach der Methode der kleinsten Quadrate durch Minimierung der Zielfunktion $v^T Pv = Minimum$ abgeleitet werden. Die Matrix A, auch Konfigurationsmatrix genannt, beinhaltet die geometrischen Zusammenhänge zwischen den Beobachtungen und Unbekannten in linearisierter Form. Dies führt mit P als Gewichtsmatrix zum Normalgleichungssystem [10].

$$A^T PA\hat{x} - A^T Pl = 0 \tag{2}$$

Das resultierende Gleichungssystem ist nicht-linear und muss zur Lösung zuvor linearisiert werden. Eine sehr genaue und robuste Linearisierung steht durch die Taylorreihenentwicklung zur Verfügung, welche gleichzeitig auch die Standardmethode ist. Für die Linearisierung ist es dabei ausreichend, wenn nach dem Glied erster Ordnung die Taylorreihenentwicklung abgebrochen wird. Allerdings werden Startwerte für die Linearisierung benötigt. Im speziellen Fall der Lokalisierung eines Knotens sind das die Näherungskoordinaten des Sensors. Zur Berechnung der Näherungswerte für die nachfolgende Ausgleichung eignen sich folgende Verfahren.

Schwerpunktbestimmung Bei der Schwerpunktbestimmung ist der Schätzwert für alle Knoten innerhalb einer Überlappungsregion konstant, im Gegensatz zum Positionierungsfehler. Er ist definiert als Abstand zwischen der geschätzten und der exakten Position des Unbekannten. Jeder Knoten berechnet seine Position mit einem Positionierungsfehler aufgrund der Algorithmuseigenschaften. Der Fehler ist null, wenn die Koordinaten des Knotens mit dem Schwerpunkt aller Nachbarbeacons übereinstimmen. Der Fehler variiert mit den geometrischen Eigenschaften des Feldes [11].

Gewichtete Schwerpunktbestimmung In einem erweiterten Ansatz betrachten wir aufbauend auf der Schwerpunktbestimmung zusätzlich Distanzen, um die Präzision der Positionsbestimmung zu erhöhen. Beim Empfang eines Paketes wird der

RSSI-Wert mit verhältnismäßig geringem Mehraufwand gemessen. Aufgrund der oben genanten Signalveränderungen ist er für eine exakte Distanzbestimmung jedoch nur eingeschränkt brauchbar. Deshalb wird der RSSI-Wert nur als Gewicht innerhalb einer gewichteten Schwerpunktbestimmung genutzt [12].

Nearest Beacon Hier wird der Beacon mit der kürzesten Entfernung zu dem Unbekannten genommen. Dabei kann davon ausgegangen werden, dass sich der nächste Beacon nicht weit entfernt von der tatsächlichen Position des Sensorknotens befindet und somit als Näherungswert geeignet scheint.
Im Gegensatz zu den beiden erstgenannten Lokalisierungsmethoden benötigt die Nearest Beacon-Methode keinen Berechnungsaufwand der Näherungskoordinaten. Diese können nach Senden der Beaconpositionen aus der Beaconpositionstabelle im Speicher der Knoten entnommen werden.

3 Hybrider Algorithmus zur Lokalisierung

In unseren Simulationen werden die ausgeglichenen Koordinaten mittels einer iterativen Ausgleichung bestimmt. Dabei untersuchen wir die Qualität der Eingangswerte hinsichtlich der benötigten Iteration für die Ausgleichung. Eine steigende Anzahl Iterationen hat zur Folge, dass die Berechnung der ausgeglichenen Koordinaten mehr Rechenzeit benötigt und somit energielastiger ist.
Die bereits eingeführten Berechnungsschritte sollen an dieser Stelle in einen verteilten Algorithmus überführt werden. Dabei muss entschieden werden, welche Knoten welche Teilaufgabe übernehmen. Dazu kommen 3 verschiedene Möglichkeiten in Betracht.

1. Der Algorithmus kann vollständig zentral auf einen sehr leistungsstarken Knoten, z.B. der Basisstation, ausgelagert werden. Obwohl hierbei die Energiereserven für die Berechnung nicht ins Gewicht fallen, müssen alle Daten durch das Netz an die Basisstation geleitet werden, was zu einem unnötig hohen Kommunikationsaufwand führt. Weiterhin blockiert ein Ausfall der Basis oder eine fehlerhafte Wegleitung die Funktionsweise des Sensornetzwerkes.
2. Eine weitere Möglichkeit ergibt sich in der Ausführung des Algorithmus auf jedem Knoten also vollständig verteilt. Aufgrund der sehr geringen Energiereserven auf jedem Knoten, ist diese Möglichkeit jedoch nicht realistisch. Die Lebenszeit des Sensornetzwerkes würde sich stark verringern. Der Netzwerkverkehr wäre jedoch weitestgehend minimiert.
3. Dies führt zur Entwicklung einer hybriden Variante, bei der die Aufgaben sinnvoll verteilt werden. Ein entsprechender Algorithmus dazu gestaltet sich wie folgt:
   - **Phase 0: Initialisierung:**
     Alle Beacons senden ihre Position $B(x, y)$ zur Basisstation und allen Knoten in Reichweite.
     Die Sensorknoten messen die Distanz zu den Beacons.

- **Phase 1: Näherungskoordinatenberechnung:**
  Die Sensorknoten berechnen die Näherungskoordinaten nach einer der aufgeführten Methoden.
- **Phase 2: Kommunikationsphase:** Die Sensorknoten senden ihre Näherungskoordinaten und die gemessenen Distanzen an die Basisstation.
- **Phase 3: Ausgleichungsphase:**
  Die Basisstation berechnet die Ausgleichung und führt Qualitätstest durch.
- **Phase 4: Kommunikationsphase (verteilt):**
  Die Basisstation sendet die ausgeglichenen also fehlerminimierten Koordinaten an die Sensorknoten.

Bei dieser Verteilung übernimmt die Basisstation die komplexere Ausgleichung und entlastet die Sensorknoten vollständig.

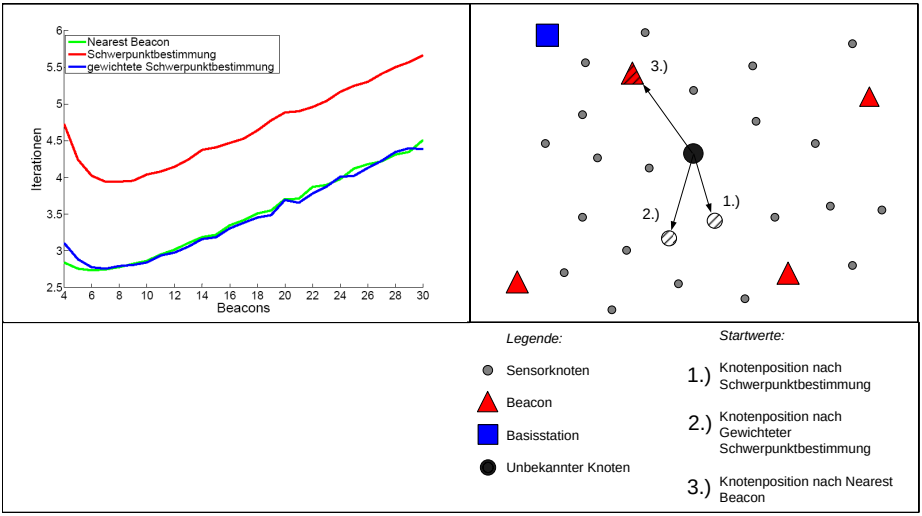


Abbildung 1. 1a) Anzahl Iterationen über einer steigenden Anzahl Beacons; 1b) Näherungskoordinatenbestimmung

Die durchgeführte Simulation umfasste ein Testfeld von 100x100. In der Abbildung 1a sind erste Ergebnisse der Ausgleichung mit Eingangswerten aus den drei Verfahren zur Näherungskoordinatenbestimmung dargestellt. Dabei ist die Anzahl Iterationen über einer steigenden Zahl Beacons angegeben. Die Ausgleichung mit den Koordinaten aus der Schwerpunktbestimmung benötigt mehr Iterationen als es bei den anderen Methoden der Fall ist. Die Schwerpunkbestimmung liefert ungenauere Näherungswerte für die Koordinaten der Unbekannten. Das rechte Bild 1b zeigt eine schematische Darstellung eines Sensornetzwerkes mit den möglichen Startpositionen.

4 Ausblick

Gegenstand der weiteren Forschung ist die Einbeziehung der Distanzen zwischen den Sensorknoten, die damit eine weitere Überbestimmung des Gleichungssystems zur Folge haben wird. Ein geeigneter mathematischer Algorithmus zur energieeffizienten Berechnung des Gleichungssystems ist ebenso Ansatz für weitere Untersuchungen wie auch eine Umsetzung auf reale Sensorplattformen, um die Ergebnisse in der Praxis zu testen.

Danksagung

Dieses Projekt wird durch die Deutsche Forschungsgemeinschaft (DFG) unter den Nummern BI467/17-1 und TI254/15-1 (Schlüsselwort: Geosens) gefördert.

Literatur

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **38** (2002) 393–422
2. Min, R., Bhardwaj, M., Cho, S., Sinha, A., Shih, E., Wang, A., Chandrakasan, A.: Low-power wireless sensor networks. In: International Conference on VLSI Design. (2001)
3. Bill, R., Cap, C., Kofahl, M., Mund, T.: Indoor and outdoor positioning in mobile environments – a review and some investigations on wlan positioning. Geographic Information Sciences **10** (2004) 91–98
4. Gibson, J.: The mobile communications handbook. CRC Press (1996)
5. Alouini, M.: Global Positioning System: An Overview. California Institute of Technology (1996)
6. Bulusu, N.: Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine **7** (2000) 28–34
7. Doherty, L., Pister, K., Ghaoui, L.E.: Convex position estimation in wireless sensor networks. In: Proceedings of IEEE Infocom. (2001) 1655–1663
8. Tian, H., Chengdu, H., Brian, B.M., John, S.A., Tarek, A.: Range-free localization schemes for large scale sensor networks. In: 9th Annual International Conference on Mobile Computing and Networking. (2003)
9. Nagpal, R., Shrobe, H., Bachrach, J.: Organizing a global coordinate system from local information on an ad hoc sensor network. In: 2nd International Workshop on Information Processing in Sensor Networks. (2003)
10. Niemeier, W.: Ausgleichungsrechnung. de Gruyter (2002)
11. Reichenbach, F., Blumenthal, J., Timmermann, D.: Improved precision of coarse grained localization in wireless sensor networks. In: Proceedings of 9th Euromicro Conference on Digital system design, Croatia. (2006)
12. Blumenthal, J., Reichenbach, F., Timmermann, D.: Precise positioning with a low complexity algorithm in ad hoc wireless sensor networks. PIK - Praxis der Informationsverarbeitung und Kommunikation **28** (2005) 80–85

On the Relation of Neighborhoods and Distances

Carsten Buschmann, Dennis Pfisterer, and Stefan Fischer

Institute of Telematics, University of Lübeck, Germany
{buschmann, pfisterer, fischer}@itm.uni-luebeck.de,
http://www.itm.uni-luebeck.de

Abstract. In this paper we present a novel approach to distance estimation that does neither depend on special hardware nor on unreliable measurements of physical wireless communication properties. Instead, it is inspired by the observation that distant nodes have fewer neighbors in common than close ones and calculates distances from intersection cardinalities of sets of adjacent nodes. We discuss related work and present the new approach including its mathematical foundations. A first simulative performance analysis shows that our scheme yields competitive results.

1 Introduction

Information about distances to other nodes in wireless sensor networks has proven advantageous not only for location discovery but is also helpful for context-aware applications in general.

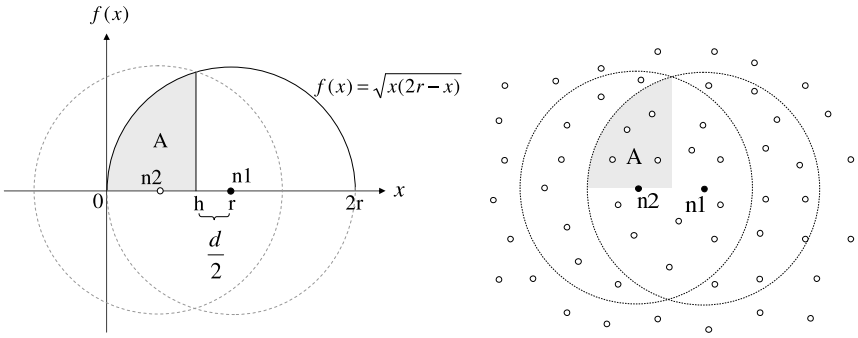
Many different systems for distance estimation (often also called ranging) have been developed. Nearly all of them employ a sender-receiver-scheme: one node emits some kind of signal, the other uses a receiver to measure physical signal properties. Two major approaches can be distinguished: The first one tries to infer the distance from the received signal strength indicator (RSSI). It tends to be quite accurate for short ranges if extensive post-processing is employed, but is imprecise beyond a few meters [1]. At ranges of 20 m and below, simple radio chips lead to errors of 50% to 100% while more sophisticated ones enable distance estimates exhibiting errors of about 10% after calibration [2].

The other group of schemes use a technique called "differential time of arrival" (DTOA). Two signals propagating with different speeds are sent out and their difference in time of arrival is measured. DTOA systems inherently require an extra actuator and detector pair which increases cost, size and energy consumption of the hardware platform. They yield accuracies of 0.5% to 10% but only achieve maximum ranges of 3–15 m [3], [4], [2].

In this paper we present the neighborhood intersection distance estimation scheme (NIDES). This novel approach to distance estimation does not rely on special hardware or unreliable measurements of physical signal properties. Instead, its operation is based on the observation that nodes that are located close to each other share many of their neighbors whereas distant nodes share only few neighbors.

2 Neighborhood-Based Distance Estimation

The basic idea behind NIDES is straightforward: if one assumes that the communication range of a node is represented by a circle with radius r around it, then the distance between two nodes can be inferred from the intersection of the circles. If the nodes are very close to each other, the intersection area has nearly the size of the full circle whereas longer distances lead to smaller intersections.



(a) Theoretic area-based function analytic approach.

(b) Practical example using neighborhood intersection.

Fig. 1. Mathematical foundation of NIDES.

What is needed for distance estimation is a functional relation that takes the cardinality of the intersection as its input and yields the distance as its output. Figure 1(a) depicts the two nodes n1 and n2 and their communication ranges, where n1 wants to calculate its distance to n2. Let us assume that the solid-line half circle represents the communication range with radius r of n1. The node n1 is indicated by the black dot at $(r, 0)$. In this idealized plot, the communication range is given by the graph of the circle function

$$f(x) = \sqrt{x(2r - x)}. \tag{1}$$

The area A (shaded grey) is one quarter of the intersection area of the two communication circles of n1 and n2 and is described by

$$F(x) = \int f(x)\,\mathrm{d}x = \frac{r^2 \arcsin(\frac{x-r}{r}) + (x - r)\sqrt{x(2r - x)}}{2} + c \tag{2}$$

which is f's antiderivative. We have determined $c = \frac{\pi r^2}{4}$ by setting $F(0)$ to 0. As for distance estimation, we have the situation that the size of the grey area A is known and that h is to be inferred. In other words, we want to deduce the value for x that corresponds to a certain integral value. Unfortunately F cannot

be solved to x. Hence we have approximated F by a Taylor approximation T of degree 3 around $x = r$:

$$T(x) = -\frac{2x^3 - 6rx^2 - 6r^2x + r^3(10 - 3\pi)}{12r}. \tag{3}$$

Now T can be solved to x, which yields three solutions of which the relevant one is

$$h = r + 2\sqrt{2}r \sin\left(\frac{\arcsin\left(\frac{3\sqrt{2}(4A-\pi r^2)}{16r^2}\right)}{3}\right). \tag{4}$$

Now the distance between the two nodes can easily be calculated using $d = 2(r - h)$. Like this, the distance between adjacent nodes can be inferred from their communication range r and A which represents one quarter of their communication circles' intersection area. We premise that the range r (or at least an upper bound) is known a priori and will be the same for all devices. Hence we treat it like a constant here.

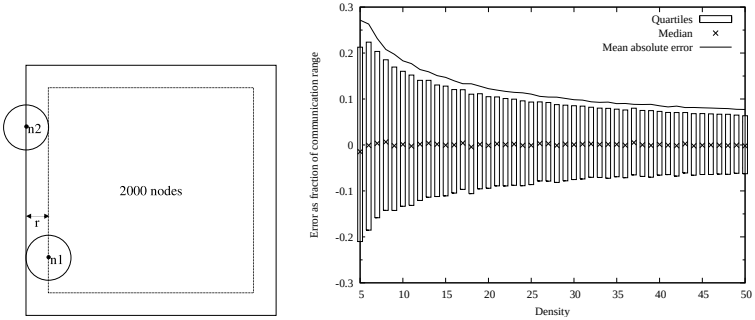
A is approximated using the heuristic that the size of A is proportional to the cardinality of the neighbor set intersection of the two nodes. The underlying assumptions are that the wireless nodes are spread out locally uniformly over the considered area and that the network is sufficiently dense. To clarify the underlying heuristic, imagine the nodes were positioned along a grid. Then each node would represent a quadratic area around it. This basic idea of each node representing a certain area also works even if nodes are randomly distributed.

Hence, the fraction of its neighbors that a node shares with a particular neighbor can be used to approximate the size of the area A to $A = 0.25\frac{s}{n}\pi r^2$ where n is the total number of neighbors the node has and s is the number of neighbors shared with the other node (see Figure 1(b)). Like this, nodes can estimate the distances to their neighbors using a simple neighbor list exchange protocol.

3 Simulative Evaluation

To evaluate the distance estimation accuracy of NIDES we ran a set of simulations. We decided not to simulate the actual exchange of data packets because we wanted to lay the focus of our work on effects that directly influence the distance estimate rather then on networking aspects. For this reason we chose SHAWN [5] for our simulations. It can analyze neighborhood relationships very efficiently and was developed for algorithmic simulations rather than network stack simulations.

We used the simulation scenario depicted in Figure 2(a). When analyzing the estimation accuracy, we only considered nodes that were located inside the dotted inner rectangle. The width of this inner area is $2r$ smaller than the simulation area, where r is the communication range. Thus the full communication range of the considered nodes resides within the simulation area (c.f. node n1). This



(a) Simulation scenario. (b) Estimation error distribution over density.

Fig. 2. Mathematical foundation of NIDES.

avoids edge effects that would be caused by missing neighbors for node n2 on the left.

For all simulations, the communication range was set to 10. However, the obtained results are valid for arbitrary communication ranges.

We iterated over all pairs of adjacent nodes inside the inner rectangle, estimated their distances using NIDES, and compared these estimates to their real Euclidean distances. To obtain statistically sound results, we averaged the results of 100 simulations with the same parameter set.

Figure 2(b) shows an interquartile diagram of the estimation error over the average neighborhood size. The error is expressed as a fraction of the communication range r.

In an interquartile diagram, each bar describes a value distribution where only 25% of the values lie above the upper bound and below the lower bound of the bar. Figure 2(b) also includes the mean absolute error and the median. For the earlier the absolute error values were summed up for all considered distances and divided by their number, while for the latter a value is selected so that half of the errors are bigger and the other half is smaller. Hence the median indicates whether the estimates are biased.

The spread of the interquartiles decreases with increasing network densities, i.e. neighborhood sizes. As expected, the estimates get more exact with bigger neighborhoods.

With many neighbors, the interquartile of the estimates lies within $0.06r$ around the correct distance (i.e. 50% of the estimates feature an error of less than 6% of the communication range), the mean absolute error is $0.075r$. For densities above 10 the mean absolute error is below 18% of the communication range. Note that the median error is always very close to zero which means that NIDES has no tendency to systematically over- or underestimate distances.

Research has shown that the required density depends on the network size. The more nodes a network consists of, the higher the required density is [6].

For a network comprising 300 nodes, a density of about 13 is needed. Hence we consider an average neighborhood size of 15 to be a reasonable choice for wireless sensor networks. With 15 neighbors, the mean error is about $0.15r$ with an interquartile spread of about 0.2. This means that half of the estimates exhibit an error of 10% of the communication range or less.

4 Conclusion

In this paper we presented the neighborhood intersection distance estimation scheme (NIDES). This novel approach computes distance estimates from intersection cardinalities of sets of adjacent nodes. Through simulations we assessed the ranging accuracy of NIDES. Because the scheme gets more accurate with increasing network density, it especially targets dense wireless sensor networks rather than sparse ones. But also if nodes have no more than 15 neighbors, NIDES achieves an average error of only about 15% of the communication range.

Thus NIDES categorizes between existing approaches for distance estimation. It achieves a higher accuracy than estimates based on RSSI, and is less susceptible to reflections, fading and multi-path propagation. Estimates based on differential time of flight feature a higher accuracy than NIDES but demand for specific hardware such as ultra sound emitters and receivers. These have negative effects on cost, size and power consumption, which seems disadvantageous especially for wireless sensor networks. In addition, these have only a very limited maximum range compared to NIDES.

We are currently deepening the simulative evaluation. In the future, we plan to extend NIDES to more realistic radio models and evaluate the performance in real world experiments.

References

1. Maroti, M., Völgyesi, P., Dora, S., Kusy, B., Nadas, A., Ledeczi, A., Balogh, G., Molnar, K.: Radio interferometric geolocation. In: SenSys "05: Proceedings of the 3rd international conference on Embedded networked sensor systems. (2005)
2. Whitehouse, K., Karlof, C., Woo, A., Jiang, F., Culler, D.: The effects of ranging noise on multihop localization: an empirical study. In: The Fourth International Conference on Information Processing in Sensor Networks (IPSN "05). (2005)
3. Whitehouse, K., Culler, D.: Calibration as parameter estimation in sensor networks. In: WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, New York, NY, USA, ACM Press (2002) 59–67
4. Savvides, A., Han, C.C., Strivastava, M.B.: Dynamic fine-grained localization in ad-hoc networks of sensors. In: MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking, New York, NY, USA, ACM Press (2001) 166–179
5. Kröller, A., Pfisterer, D., Buschmann, C., Fekete, S.P., Fischer, S.: Shawn: A new approach to simulating wireless sensor networks. In: Design, Analysis, and Simulation of Distributed Systems 2005, Part of the SpringSim 2005. (2005)
6. Xue, F., Kumar, P.R.: The number of neighbours needed for connectivity of wireless networks. IEEE Wireless Networks **10**(2) (2004) 169–181

Teilnehmerliste

Barth, Matthias	Dt. Zentrum f. Luft- u. Raumfahrt
Baunach, Marcel	Universität Würzburg
Blaß, Erik-Oliver	Universität Karlsruhe
Born, Alexander	Universität Rostock
Briones, Jorge Alfonso	Universität Stuttgart
Budelmann, Christoph	DFKI-Labor Bremen
Buschmann, Carsten	Universität Lübeck
Dietrich, Isabel	Universität Erlangen-Nürnberg
Dressler, Falko	Universität Erlangen-Nürnberg
Dürr, Frank	Universität Stuttgart
Emmert, Barbara	Universität Würzburg
Felser, Meik	Universität Erlangen-Nürnberg
Fenne, Martin	FH-Coburg
Fuhrmann, Thomas	Universität Karlsruhe
Gauger, Matthias	Universität Stuttgart
Gonzalo, Manuel	Universität Stuttgart
Guerrero, Pablo	TU Darmstadt
Hähner, Jörg	Universität Stuttgart
Hauswirth, Manfred	EPFL
Heinz, Christoph	Philipps-Universität Marburg
Hessler, Alban	NEC Europe Ltd.
Hof, Hans-Joachim	Universität Karlsruhe
Hoßfeld, Tobias	Universität Würzburg
Hurler, Bernhard	Universität Karlsruhe
Jacobi, Daniel	TU Darmstadt
Kapitza, Rüdiger	Universität Erlangen-Nürnberg
Katz, Bastian	Universität Karlsruhe
Kellner, Simon	Universität Karlsruhe
Koldehofe, Boris	Universität Stuttgart
Kolla, Reiner	Universität Würzburg
Krämer, Marc	TU Kaiserslautern
Krüger, Daniela	Universität zu Lübeck
Kuntz, Andreas	Universität Karlsruhe
Lachenmann, Andreas	Universität Stuttgart
Landsiedel, Olaf	RWTH Aachen
Marrón, Pedro José	Universität Stuttgart
Mattern, Friedemann	ETH Zürich
Minder, Daniel	Universität Stuttgart
Mühlberger, Clemens	Universität Würzburg
Müller, Jens	SAP Research

Niedermeier, Christoph	Siemens AG
Pfisterer, Dennis	Universität zu Lübeck
Reiser, Hans	Universität Ulm
Ringwald, Matthias	ETH Zürich
Römer, Kay	ETH Zürich
Rothermel, Kurt	Universität Stuttgart
Saukh, Olga	Universität Stuttgart
Sauter, Robert	Universität Stuttgart
Schmitt, Jens	TU Kaiserslautern
Schuster, Sebastian	Universität Karlsruhe
Seeger, Bernhard	Universität Marburg
Staehle, Dirk	Universität Würzburg
Vogt, Harald	SAP Research
Waelchli, Markus	Universität Bern
Weise, Thomas	Universität Kassel
Weyer, Christoph	TU Hamburg-Harburg
Wieland, Thomas	FH-Coburg
Witt, Matthias	TU Hamburg-Harburg
Wittenburg, Georg	FU Berlin
Wolf, Jonas	ETH Zürich

Organisation

Pedro José Marrón

Matthias Gauger	Andreas Lachenmann	Olga Saukh
Manuel Gonzalo	Daniel Minder	Robert Sauter

Autorenverzeichnis