RWTH Aachen

Department of Computer Science
Technical Report

6. Fachgespräch Sensornetzwerke

Klaus Wehrle

ISSN 0935–3232 · Aachener Informatik Berichte · AIB-2007-11

RWTH Aachen · Department of Computer Science · July 2007

6. Fachgespräch Sensornetzwerke

Klaus Wehrle

LuFG Informatik IV, Verteilte Systeme
RWTH Aachen, Germany
Email: klaus.wehrle@cs.rwth-aachen.de

6. Fachgespräch Sensornetzwerke

der GI/ITG Fachgruppe „Kommunikation und Verteilte Systeme"

Herausgeber: Distributed Systems Group
RWTH Aachen University
http://ds.rwth-aachen.de

Technischer Bericht der RWTH Aachen

16.7/17.7
2007

ISSN 0935-3232 AIB 2007-11

Distributed Systems UMIC ITG PHILIPS GI

Inhaltsverzeichnis

I Betriebssysteme / Hardware

II Simulation / Evaluation

III Abstraktion

IV Sicherheit

V Einsatz

VI Routing

VII Drahtloses

VIII Anwendungen

IX Protokolle

Introduction to a Small Modular Adept Real-Time Operating System

Baunach, Marcel Kolla, Reiner Mühlberger, Clemens
Department of Computer Engineering, Am Hubland, University of Würzburg, Bavaria, Germany
{baunach, kolla, muehlberger}@informatik.uni-wuerzburg.de

Abstract

In this paper we present *Smart*OS, a preemptive real-time operating system for embedded systems like sensor/actor nodes. After a short introduction and motivation, we present the basic concepts along with a simple example and conclude with current work and further outlook.

Keywords operating system, real-time, multitasking, wireless sensor network, sensor node

1. Introduction

Distributed systems are made of several interacting components. Wireless sensor networks in particular may consist of a huge number of cooperating nodes handling even complex tasks. However, such embedded systems are subject to tight power, little memory and hard energy constraints. For some applications it is even necessary to offer real-time operation within control and feedback control systems on sensor/actor nodes.

Up to a certain complexity, a single-loop software system might accomplish these demands, but this would not be very comfortable at all as it is inflexible and consumes lots of energy and space. Moreover, the response time of such single-task systems is high, the processor load is poor and the program code is hardly maintainable or reusable.

An operating system (OS) might grant efficient and suitable solutions for these problems. Obviously, it has significant influence on the performance of embedded devices as it coordinates hardware access as well as execution and cooperation of all software parts. Since subtle optimizations require deep analysis of the overall system software, it is common practice in embedded software development to link operating system, drivers and the application itself to one monolithic block.

Further challenges arise from the distribution of applications over a large number of nodes. This way, communication becomes another main focus during the development process to which a good operating system should contribute. Special problems here are (wireless) networking, node synchronization, information propagation, security and in some cases the interaction of different node architectures within heterogeneous networks.

This paper introduces *Smart*OS, a **s**mall **m**odular **a**dept **r**eal-**t**ime **o**perating **s**ystem for sensor nodes. It starts with the motivation for its development despite of the availability of other similar systems. Next, basic concepts, some implementation details and a short software example follow. We conclude with an overview of current *Smart*OS based projects and an outlook to further development goals and ongoing research.

2. Motivation

For embedded systems and wireless sensor networks in particular, some operating systems are yet available.

The component based tinyOS [1] applies an execution model driven by events and commands. The run-to-completion tasks are non preemptive and share a single stack. Applications are implemented using the programming idiom nesC. The operating system freeRTOS [2] is capable of real-time operation and manages a message queue for its alternatively preemptive or cooperative tasks. Applications are written in plain C and a trace visualization tool exists. Preemptivity is also found within the small operating system SOS [3]. It offers an event-based design and support for critical sections without priority ceiling. Up to 7 independent timers can be handled. The main focus of BTnut [4] lies on network application. It deals with prioritized and cooperative but non preemptive tasks. Dynamic heap allocation is implemented and a system tracer is offered.

However, our vision of an OS for sensor/actor nodes demanded fully preemptive tasks despite of a modular design, real-time operation and energy awareness. Hence, we desired a priority based scheduler, a high resolution time management with a local timeline and an unified interrupt and resource concept with priority ceiling. This allows periodic tasks as well as precise timestamping of internal and external events. Applications should be developed in plain C for efficient low level hardware access. The next section addresses the basic techniques in detail.

3. *Smart*OS concepts

*Smart*OS is a minimalistic operating system for small devices like sensor nodes. The main goals are preemptive multitasking, real-time operation and modularity despite of little RAM and ROM requirements on slow microcontrollers or processors. Due to hardware constraints on most microcontrollers, memory protection is not supported. A reference implementation for TI's MSP430 MCU family [5] is available and includes

- an economic and fast core, responsible for scheduling and context switching, time and resource management, energy savings and error recovery,
- support for various hard- and software resources like device drivers and communication protocols,
- low interrupt latency with automatic timestamping and demultiplexing of shared hardware interrupt vectors, and
- modular task and resource composition for assembling applications from code repositories.

The four foundation pillars are tasks, events, resources and time management (see Fig. 1). They allow efficient and easily maintainable software development even for complex embedded applications.

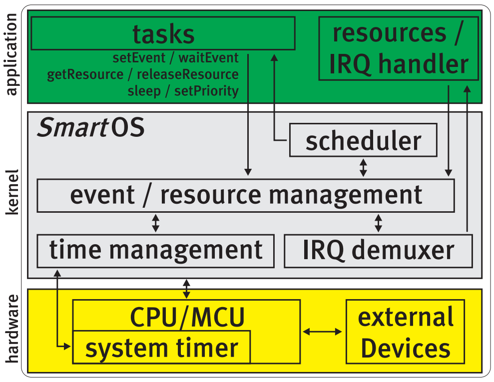


Figure 1. *Smart*OS architecture

*Smart*OS' *time management* uses a 64 bit timeline, driven by a hardware timer of the underlying MCU. This way, timeouts as well as all other time-dependent services gain a high precision. In case of the MSP430 MCU, this means a resolution of 1 µs for timestamping and task scheduling. An additional watchdog timer can be activated to recover from software failures within non-responding tasks (deadlocks, endless loops, etc.).

*Smart*OS *tasks* describe the behaviour of the overall system, i.e. they control software and hardware activities. Up to 254 user defined and preemptive tasks are supported, each possessing a never returning entry function, an initial priority (1 – 255) and an individual stack area. A source code profiler is available to compute an upper bound for the stack size of each task at compile time – if possible. As such analysis is a hard problem, adequate annotations within the C comments can be added to refine the estimation. The set of tasks is static after linking but their priority can be modified dynamically at runtime. The scheduler selects the execution order depending on the task priorities and allows context switching within e.g. 20 µs on an MSP430 MCU running at 8 MHz. Per default, the predefined idle task is scheduled with lowest priority (0) if no other task requires CPU time. This is essential for energy management as it controls architecture specific low power modes and performs dynamic frequency control.

*Smart*OS supports up to 255 user definable *events* to synchronize tasks and to interact with hardware components. Therefore, hardware interrupts are directly mapped to events. Each task may wait for the occurrence of an event with a relative or an absolute timeout. This allows further actions even if the expected event does not occur. Otherwise, if no timeout is given, the task might wait forever. The major advantage of absolute deadlines is the higher precision when implementing periodic tasks. If an event is set by a task or an interrupt handler, this causes the highest prioritized task waiting for this event to become ready (see Fig. 2). In consequence, this might even produce a context switch.

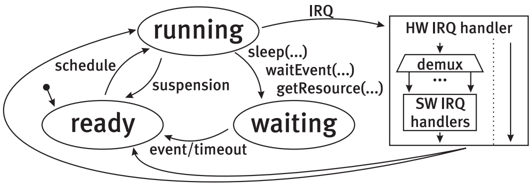


Figure 2. *Smart*OS scheduling and IRQ handling

*Smart*OS *resources* coordinate the (exclusive) access of tasks on hardware devices, like timers and buses, or on abstract entities, like data structures. Thus, semaphores can also be simulated easily. Furthermore, an individual initialization function per resource may be specified for automatic configuration of the underlying object at system startup. Internally, the assignment of a resource is managed via events. Again, tasks can wait for resource allocation by specifying an absolute, relative or no deadline. To avoid thwarting of resource owning tasks, *priority ceiling* within the resource concept temporarily increases the priority of the currently owning task up to the highest priority of all other tasks waiting for this particular resource. Only the owner task of a resource is liable and capable for releasing it.

*Smart*OS' *interrupt concept* supports hardware and software interrupt handling in kernel mode. Per default, interrupt cascading is disabled but can be reactivated if desired. As soon as a hardware interrupt occurs, a general IRQ dispatcher is executed. It stores the current system time with a latency between 9 and 10 cycles on a MSP430 MCU. When running at 8 MHz, this allows the calculation of a timestamp with precision of ≈ 0.8 µs regarding a time discretion of

1 µs for the timeline. Next, the dispatcher switches to the kernel stack for context saving and handler execution. After processing the specific handler function, the scheduler is executed again. For most architectures it is common practice to share hardware IRQs among several sources. This is a problem when developing modular software components independently from each other as a common IRQ handler must be adopted to meet the requirements of all modules. *Smart*OS addresses this issue by supporting independent software IRQ handlers for arbitrary sources. These are subordinate to hardware IRQ handlers and allow automatic demultiplexing of shared hardware interrupts (see Fig. 2).

For most hard real-time applications, performance can not be sacrificed to provide safety or convenience. To optimize speed and reactivity, *Smart*OS executes as little code as possible in kernel mode and disables hardware interrupts only within a very short section of a few assembler instructions. Apart from some architecture specific parts which use assembler, *Smart*OS is implemented in the C programming language. The reference implementation for TI's MSP430 MCU [5] consumes 1.4 kB program ROM and 90 byte RAM. Any additional task requires 46 byte RAM. Energy analysis on the sensor node SNoW^5 [6, 7] running at 8 MHz showed a current consumption of 6.5 mW in sleep mode (idle task) and 19 mW in active mode.

4. Example

For better understanding the following example will illustrate some *Smart*OS concepts from section 3. First, we will declare two tasks, each controlling the periodic flashing of a LED. The one with relative, the other with absolute delay.

```
Time_t delay = 3000000;          // delay in µs

// task with stack-size 10×16bit, priority 200
OS_DECLARE_TASK(tLED_Red, 10, 200);
OS_TASKENTRY(tLED_Red) {         // entry function
  while (1) {
    sleep(delay);                // relative delay
    LED_toggle(LED_RED);
  }
}

// task with stack-size 10×16bit, priority 200
OS_DECLARE_TASK(tLED_Blue, 10, 200);
OS_TASKENTRY(tLED_Blue) {        // entry function
  Time_t deadline;
  getCurrentTime(&deadline);

  while (1) {
    deadline += delay;
    sleepUntil(&deadline);       // absolute delay
    LED_toggle(LED_BLUE);
  }
}
```

Next, we will do some work concurrently to the LED tasks. Therefore, we declare one single interrupt handler to manage two channels of a DMA controller. It fills a buffer for subsequent digital signal processing. As soon as the buffer is full, an event will be triggered and a task waiting for this event will resume. Notice the automatic stack size estimation for the DSP task and its exclusive access on the data buffer.

```
// event and resource declaration
OS_DECLARE_EVENT(evDSP);
OS_DECLARE_RESOURCE(DataBuf);

// IRQ handler for DMA processing
void BufferHandler(int channel) {
  // ... some (channel dependent) code
  if (buffer_is_full)
    __syscall_setEvent(&evDSP);   // trigger event
}

// declare IRQ handler for channel 0
OS_DECLARE_IRQ_HANDLER(OS_IRQ_DMA0,
   &BufferHandler, 0);

// declare IRQ handler for channel 1
OS_DECLARE_IRQ_HANDLER(OS_IRQ_DMA1,
   &BufferHandler, 1);

// DSP task with automatic stack-size estimation
// and priority 50
OS_DECLARE_TASK(tDSP, __STACK__AUTO__, 50);
OS_TASKENTRY(tDSP) {   // entry function
  while (1) {
    waitEvent(&evDSP); // wait without timeout
    // get exclusive access on DataBuf,
    // process the buffer, and release it.
    getResource(&DataBuf);
    DSP();             // some DSP functionality
    releaseResource(&DataBuf);
  }
}
```

The final step is to properly launch the system at power-up. This is done by a never returning main-function as entry point for the whole application:

```
OS_MAIN {
  init_osc();              // init MCU clock
  os_init_environment();// init tasks,events,res.
  run_os();                // start SmartOS scheduler
}
```

5. Conclusion and outlook

In this paper we gave some benefits of operating systems for embedded systems and sensor/actor nodes in particular. Next, we introduced some of our reasons for implementing an operating system from scratch despite of other existing ones. The underlying techniques of *Smart*OS were explained and a meaningful example provided a deeper insight to complete this paper.

*Smart*OS was yet tested successfully within various projects like the ultrasonic localization system SNoW Bat [8]. Further applications based on *Smart*OS and extensions for the SNoW^5 sensor node include stepper motor control, digital compass implementation, GPS readout, CAN bus interfacing and a TCP/IP stack for ethernet connection. The actual main project is the real-world installation of a WSN based vehicle-to-infrastructure communication system comprising 70 SNoW^5 nodes. Right now, *Smart*OS is available for TI's MSP430 MCU family [5] and thus runs also on some other nodes like TelosB [9]. However, we are working on further ports for other 16-bit and 32-bit microcontrollers, e.g. Hitachi's SuperH family.

Our short term research objectives are the support for multi-CPU nodes, energy harvesting techniques and analysis of various wireless communication protocols (B-MAC [10], PEDAMACS [11], SCP-MAC [12], TRAMA [13]). Additionally, we are studying the possibilities for secure and remote software updates via radio, IrDA and ethernet. One long-term goal is the integration of well-known concepts from agent technologies like negotiation or auction into existing processes for data propagation or task scheduling.

References

[1] UC BERKELEY: *TinyOS*. `http://www.tinyos.net/`, 2004.

[2] BARRY, RICHARD: *FreeRTOS^{TM} homepage*. `http://www.freertos.org/`, 12. December 2005.

[3] SKYDAN, OLEG: *SOS - small operating system*. `http://skydan.in.ua/SOS/`, 25. February 2006.

[4] ETH ZURICH: *BTnut*. `http://www.btnode.ethz.ch/`, 2007.

[5] TEXAS INSTRUMENTS INC., Dallas (USA): *MSP430x1xx Family User's Guide*, 2006.

[6] KOLLA, REINER, MARCEL BAUNACH, and CLEMENS MÜHLBERGER: *Snow^5: a modular platform for sophisticated real-time wireless sensor networking*. Technical Report 399, Institut für Informatik, Universität Würzburg, January 2007.

[7] KOLLA, REINER, MARCEL BAUNACH, and CLEMENS MÜHLBERGER: *Snow^5: A versatile ultra low power modular node for wireless ad hoc sensor networking*. In MARRÓN, PEDRO JOSÉ (editor): *5. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, pages 55–59, Stuttgart, July 2006. Institut für Parallele und Verteilte Systeme.

[8] KOLLA, REINER, MARCEL BAUNACH, and CLEMENS MÜHLBERGER: *SNoW Bat: A high precise WSN based location system*. Technical Report 424, Institut für Informatik, Universität Würzburg, May 2007.

[9] POLASTRE, JOSEPH, ROBERT SZEWCZYK, and 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.

[10] POLASTRE, J., J. HILL, and D. CULLER: *Versatile low power media access for wireless sensor networks*. In *SenSys04*, pages 95–107, Baltimore, MD, November 2004. B-MAC.

[11] COLERI-ERGEN, S. and P. VARAIYA: *Pedamacs: Power efficient and delay aware medium access protocol for sensor networks*. IEEE Trans. on Mobile Computing, 5(7):920–930, July 2006. PEDAMACS.

[12] YE, W., F. SILVA, and J. HEIDEMANN: *Ultra-low duty cycle mac with scheduled channel polling*. In *SenSys06*, pages 321–334, Boulder, CO, November 2006. SCPMAC.

[13] RAJENDRAN, V., K. OBRACZKA, and J. GARCIA-LUNA-ACEVES: *Energy-efficient, collision-free medium access control for wireless sensor networks*. Wireless Networks, 12(1):63–78, February 2006. TRAMA.

WIP: Energy Container for Database-Oriented Sensor Networks

[Extended Abstract]

Simon Kellner
System Architecture Group
Universität Karlsruhe
simon.kellner@kit.edu

ABSTRACT

Energy remains the most critical resource in sensor networks. In static sensor-net applications where most events can be calculated a priori, energy management is often done implicitly and manually by application developers.

With the advent of database interfaces to sensor nets, this is no longer possible due to dynamically created queries. There are several scenarios in which it is desirable to get an accurate account of the resources a sensor net consumed on behalf of a certain query.

In this work, we propose to adapt the Resource Container concept from the desktop computing world to sensor networks in order to facilitate dynamic energy accounting.

1. INTRODUCTION

Energy still is the most critical resource in sensor networks. Current energy supplies already take up most of a sensor node's space, but can provide the desired node lifetimes of years only when sensor-net application designers give a high priority to a long sensor-net lifetime. Sensor-net Operating Systems (OSes) like TinyOS [2] encourage energy saving by not providing a convenient CPU-abstraction such as threads, which could, for example, tempt application developers into creating CPU-intensive waiting loops and thus into wasting energy.

Database interfaces to sensor nets like TinyDB [3] make it easier for users to retrieve sensor data: A sensor-net application is formulated as a request in an SQL-like language and interpreted by the sensor net until the request expires. The program on the sensor nodes only needs the ability to interpret and execute such requests. This eliminates the need to reprogram sensor nodes and allows multiple queries to be processed simultaneously.

Such dynamic systems can support multiple users in a sensor net, each with his own set of queries. In this scenario it is desirable to account the energy consumption of each query, e.g. to bill users based on their sensor net "usage", or to find the query with the highest energy consumption and cancel it before it wears down the energy supplies.

On traditional computers on the desktop or in the server room, Resource Containers (RCs) are used for this purpose. In this work, we investigate how this concept can be adapted to TinyOS, an OS widely used on sensor nodes. We consider RCs for energy accounting only, although the concept can be used for all resources an OS manages.

2. BACKGROUND

In this section, traditional Resource Containers as they are used in PCs are introduced as well as the properties of TinyOS that are relevant to this work.

2.1 Resource Containers

Resource Containers are an OS-abstraction introduced by Banga, Druschel and Mogul [1] in 1999 and consist basically of OS-provided storage for accounting data. The idea is to separate OS abstractions for CPU and resource accounting, because resource usage is independent of CPU abstractions.

Instead, it is dependent on *tasks*: A task is loosely defined as something a user wants the system to do (e.g., serving a web page or drawing a picture). In modern systems, such a task is no longer identical to a process: A web-server can use threads to serve different web pages simultaneously (one process working on several tasks), or several processes cooperate to accomplish one task (e.g., graphical application and X-server).

In an RC-enabled system, every process can create RCs, change its active RC and share an RC with another process. Continuing both examples above, a web-server can bind each of its threads to a separate RC, and a graphical application can share an RC with the X-server.

One such implementation of RCs for Linux is described in [4]. Here, when a process creates a new RC, the RC is bound to a file descriptor. This has the advantage that existing code used for user-land handles to kernel objects can be reused. The RCs are organized in a hierarchy in which each parent RC holds the accounting data aggregated over all its children. So a process cannot cheat the system by creating new RCs, since they will all have the same parent.

In summary, RCs give administrators and users the ability of accounting tasks, which usually has a higher significance than process-based accounting.

2.2 TinyOS

TinyOS, one of the most prevalent OSes for sensor nodes, is event-driven. It does not provide "convenient" CPU abstractions known from other OSes like processes or threads. All activities in TinyOS can be seen as responses to interrupts. These responses typically consist of few instructions and some commands to peripheral hardware that will trigger the next interrupt, allowing the processor to sleep in the meantime. This design does not tempt inexperienced programmers into creating energy-expensive polling routines.

Instead of high-level processor abstractions like threads or processes, TinyOS opts for a low-level abstraction in order to save stack memory, the TinyOS `task`s (not to be mistaken for the RC-related tasks). TinyOS `task`s run until completion and cannot be interrupted by other TinyOS `task`s. This is a strong incentive to keep them short, as other operations would suffer serious delays. Instead, a long-running TinyOS `task` should enqueue (`post`) itself in the run queue and quit, postponing its work in effect. In summary, a typical workflow of packet reception, sensor queries, a bit of computation and packet transmission on a sensor node is distributed across several of its devices and held together by TinyOS `task`s and interrupts, interspersed with sleep intervals.

The original RC concept associates processes or threads with one or more RCs on which the OS can account resource usage. In the absence of these processor abstractions the association of resource use with RCs is more difficult.

3. RESOURCE CONTAINERS IN TinyOS

The focus of this work is on how to attach RCs to queries executed by a TinyOS application, since dynamically created queries are the most interesting targets for on-line accounting. However, RCs can be used to account other tasks as well.

In the following we assume that the application can identify queries through an ID which is present in all packets related to that query.

3.1 Normal Resource Containers

A normal RC is associated with a query. As soon as an application learns the ID of the query currently being processed, it informs TinyOS that it wishes to switch to the RC associated with this query. The selected RC is then bound to the current TinyOS `task`.

The energy consumption of all further activities coming from this TinyOS `task` is accounted to the selected RC. If the TinyOS `task` posts a new TinyOS `task` or sets up a new timer, this binding can be stored by the scheduler or timer system, and can be used to switch back to the stored RC automatically on the corresponding wake-up call.

The OS here clearly depends on the application for correct accounting, but this is both feasible and necessary in a sensor-net application. It is necessary to prevent producing hard-to-maintain code, and it is feasible because there should be only few places where this RC-switching occurs, namely when a sensor-net application starts processing a query.

3.2 Anonymous Resource Containers

Since TinyOS applications spend most of their time sleeping and perform only minimal amounts of processing, energy consumed during interrupt handling is not negligible.

For example, a timer interrupt may cause the activation of a communication device, which is subsequently used to send stored sensor data to other nodes. The sensor node is not aware of the query ID until it accesses the packet it is about to send. In the meantime, the activation of the communication device can consume a substantial amount of energy that cannot be assigned to the correct RC at that moment.

As a solution, the interrupt handler can allocate a temporary, anonymous RC and use it to account both its own energy consumption and the device activation. Later, when the application becomes aware of the query ID, it can switch to the RC associated with the query, causing the temporary RC to be merged and released.

3.3 Special Resource Containers

It may be necessary to employ special RCs to provide additional information or to handle cases where the correct RC is not known.

3.3.1 Root Resource Container

One RC worth mentioning is the RC for the whole node. It is used to collect the amount of energy consumed by the whole node, regardless of queries. This information is of interest to the nodes themselves in order to estimate the amount of remaining energy. It can also be regarded as another data source and can itself be the target of a query.

3.3.2 Idle Resource Container

Some energy consumption simply can not be clearly accounted to a query, e.g., the energy spent during sleep (*idle energy*). We call the problem of accounting this energy consumption in a fair manner *accounting fairness*.

One way to address the issue of idle energy accounting is to distribute the accounted idle energy among all queries known to the sensor node. To achieve this, a special RC for this energy class is present in the system. At certain times, this RC is cleared and its content distributed among all existing normal RCs. This has to be done both periodically and on creation/expiration of a query:

- Periodically so that the accounting information remains recent,
- At query instantiation to avoid penalizing this query by accounting sleeping energy spent before its instantiation, and
- At query expiration to avoid losing accounted energy.

The fairness of this distribution is subject to discussion and thus should be handled by a project-dependent policy. Policy examples include equal distribution and partitioning according to duty-cycle or used energy.

So, one can picture the RCs in a 3-level hierarchy: the root RC for the node, named RCs for the queries and anonymous

RCs to account energy consumed for a (yet) unknown purpose. In this hierarchy the root RC contains the aggregated accounting data of the named RCs, while the anonymous RCs will eventually be merged with one of the named RCs.

3.4 Shared Data

Caching the acquired sensor data introduces another instance of the accounting-fairness problem. Without additional measures, the first query to sample data bears the cost of acquiring it, subsequent queries can use it at almost zero cost. If the accuracy of timing or accounting can be relaxed, some trade-offs between one of them and accounting fairness can be considered.

A trade-off between timing accuracy and accounting fairness can be implemented as a subscriber model for sensor data: The sensor data is sampled either on time-out after the first subscription or when enough parties subscribed to this sensor data. The energy is split among all of the subscribed parties.

Another trade-off between accounting accuracy and fairness can be implemented by assigning a value to the sampled sensor data that decays with every access. For example, the initial query bears 3/4 of the costs, the next query 3/4 of the remaining costs, and after a time-out, the rest is distributed across all queries that acquired this data.

3.5 Resource Container Aggregation

RCs lend themselves quite naturally to sensor nets with dynamically created queries. When receiving a new query, a sensor node allocates an RC for this query, accounts the query's energy costs to that RC and sends the accounted data back together with the responses to this query.

RC contents can easily be aggregated by summation over all RCs with the same query ID. The design of RCs to store all of the energy accounted to it since its creation makes it resilient to occasional packet loss. When accounting information is lost in the network due to temporary packet loss, the aggregated accounting information at the data sink may be incorrect, but it will be correct again once the temporary packet loss is over.

To allow the data sink to compare the collected aggregated RC values and to detect packet loss, a node should additionally send the number of sensor nodes involved in an aggregate, if this information is not already present in the aggregated sensor data.

4. CONCLUSION

Resource Containers are an elegant concept for resource accounting in traditional operating systems. This concept can be adapted to the field of sensor networks, where pure event-driven operating systems prevail.

The benefit of this concept is accurate accounting of sensor network tasks, which is useful information to developers, administrators and users.

5. ACKNOWLEDGMENTS

This work is done as part of the BW-FIT project ZeuS.

6. REFERENCES

[1] G. Banga, P. Druschel, and J. Mogul. Resource containers: A new facility for resource management in server systems. In *Proceedings of the Third Symposium on Operating System Design and Implementation (OSDI'99)*, Feb. 1999.

[2] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *ASPLOS-IX: Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 93–104, New York, NY, USA, 2000. ACM Press.

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. The design of an acquisitional query processor for sensor networks. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 491–502, New York, NY, USA, 2003. ACM Press.

[4] M. Waitz. Accounting and control of power consumption in energy-aware operating systems. Master's thesis, Department of Computer Science 4, University of Erlangen, Jan. 2003. SA-I4-2002-14.

Kostenaspekte beim Entwurf von Funksensorknoten

M. Niedermayer, J. Hefer, S. Guttowski
System Design & Integration
Fraunhofer Institut für Zuverlässigkeit und Mikrointegration,
Berlin

{niederm|hefer|guttowski}@izm.fraunhofer.de

R. Thomasius, H. Reichl
Design and Simulation Methods for System Integration
Technische Universität Berlin, FSP Mikroperipherik,
Berlin

{thomasius|reichl}@izm.fhg.de

Zusammenfassung

Drahtlose Sensornetze stellen einen wachsenden Markt dar. Viele Anwendungen erfordern dabei eine Vielzahl kostengünstiger Funksensorknoten, die in verschiedenste Objekte eingebettet werden können. In der Vergangenheit wurden Protokolle, Architekturen und Fertigungstechnologien zumeist getrennt optimiert. Beim kostenoptimierten Entwurf muss der Systementwurf jedoch zusammen mit den fertigungstechnischen Randbedingungen betrachtet werden. Dabei liegt der Schwerpunkt auf der Identifikation der Kostentreiber und deren Optimierung. Nach Einführung der wesentlichen Kostenaspekte drahtloser Sensorknoten wird eine Plattform zur Kostenoptimierung vorgestellt. Diese besteht aus einem Entwurfswerkzeug und einer Infrastruktur zur Verifikation von Funksensorsystemen, um Prototypen unterschiedlicher Architekturen mit verschiedenen Aufbautechniken zu testen und die Kostensenkungsmöglichkeiten praxisnah zu analysieren.

Schlüsselwörter

Drahtlose Sensorsysteme, Kostenmodelle, Entwurfsmethodik, Hetero-Systemintegration, 3D-Aufbau- und Verbindungstechnik

1. Einleitung

Ob per Telefon, Fernsehen oder Internet – ein wesentlicher Teil des Informationsaustausches geschieht in der virtuellen Welt. In der Vergangenheit waren Menschen die Brücke zu Netzwerken, um in der virtuellen Welt zu kommunizieren. Mit der technischen Machbarkeit von kleinen kostengünstigen Systemen zur Datenerfassung in der physikalischen Welt, die ebenfalls im Netzwerk per Funk kommunizieren können, ändert sich dieser Status Quo. Drahtlose Sensornetzwerke schließen somit die Lücke zwischen der physikalischen und der virtuellen Welt. Entsprechende Sensornetzwerke mit einer größeren Anzahl an Funksensorknoten können jedoch nur dann ein großes Marktpotential erreichen, sofern der einzelne Funksensorknoten sehr kostengünstig gefertigt werden kann. Anwendungen wie Zutrittskontrollen, Fernwartung und Zustandsüberwachung werden davon profitieren, wenn drahtlose Funksensorknoten die Preisspanne von 1 bis 10€ erreichen.

Funksensorknoten weisen besonders viele Entwurfsfreiheitsgrade auf. Der Grad der Entwurfsautomatisierung ist dabei recht gering, insbesondere bei den vielen fertigungsnahen Entscheidungen, welche kostenoptimale Lösungen mit sich bringen. Beim Protokoll- und Schaltungsentwurf können die Konsequenzen im Hinblick auf die Systemkosten oftmals erst am Ende des Entwurfsablaufs beurteilt werden.

Abschnitt 2 diskutiert zunächst die Marktentwicklung und die technologischen Trends. Anschließend werden in Abschnitt 3 generelle Kostenabhängigkeiten von kompakten Funksensorknoten erörtert. Die Vorstellung der implementierten Plattform zur angewandten Kostenanalyse erfolgt in Abschnitt 4. Schließlich gibt Abschnitt 5 einen Ausblick auf zukünftige Aktivitäten.

2. Kostenentwicklung und Trends

Bei der Beobachtung der Marktentwicklung von innovativen Produkten unterscheidet man den Labormarkt und den Massenmarkt. Mit der technologischen Machbarkeit erlaubt der Labormarkt zuerst Produkte mit sehr spezifischen Anforderungen. Die Kostenrestriktionen sind eher unkritisch. Dies ändert sich mit dem Eintritt in den Massenmarkt. Während beim Investitionsgütermarkt ein preislicher Unterschied von einigen 10€ über die Marktfähigkeit entscheidet, können im Konsumgütermarkt sogar wenige Cent von Bedeutung sein. So ergibt sich eine entsprechende Markterschließung schrittweise. Funksensorknoten befinden sich gerade an der Schwelle der Öffnung des Konsumgütermarktes (Abb. 1). Zu den ersten Massenprodukten zählen derzeit drahtlose Heizkostenzähler und Reifendrucksensoren.

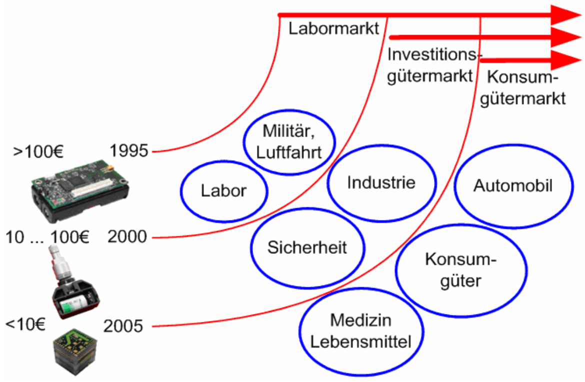


Abb. 1: Marktentwicklung von Funksensorknoten

Seit 1948 hat die Mikroelektronik ein enormes Wachstum erfahren. Die technische Entwicklung im Bereich der Mikrosystemtechnik ist ebenfalls beachtlich. Sie hinkt jedoch der Dynamik bei den integrierten Schaltkreisen deutlich hinterher. So fiel beispielsweise in den 25 Jahren von 1970 bis 1995 das Preis-Leistungsverhältnis bei Mikroprozessoren auf 0,01 Prozent, während die entsprechenden Kostenreduktionen bei Sensoren lediglich 33 und bei Aktoren 10 Prozent betrugen [1]. Auf Basis der Technologien der Mikroelektronik und Mikrosystemtechnik werden zukünftig besonders kleine und robuste Komponenten machbar. Die Kombination aus Miniaturisierung und geringeren Fertigungskosten eröffnet auch Märkte für neue Anwendungen.

Die dynamische Entwicklung bei den Komponententechnologien führte auch zu einer erheblichen Steigerung der Integrationsdichte im Bereich der Aufbau- und Verbindungstechnik. Verschiedene Trends begünstigen diese Entwicklung. So werden die Komponentengehäuse immer kleiner. Die Einführung von so genannten 3D-Gehäusen durch Stapelung mehrerer Komponenten ermöglicht deutlich steigende Packungsdichten. Auch die Entwicklung bei den Substrattechnologien erlaubt eine wirtschaftliche Komponentenintegration auf immer kleineren Raum. Besonders anschaulich ist die technologische Entwicklung im Bereich der Mobiltelefone, in dem bei einer Reduktion der belegten Leiterplattenfläche um 15 Prozent die Kosten jährlich um 25 Prozent sinken [2].

3. Kostenaspekte der Teilkomponenten

Energieautarke Funksensorknoten bestehen neben einer eigenen Energieversorgung aus den Analogkomponenten für die Messdatenerfassung und die Funkkommunikation sowie aus den digitalen Schaltungselementen für die Ablaufsteuerung und Datenverarbeitung [3]. Auch wenn es die Vision eines universellen Funksensorknotens gibt, werden zukünftig kostenoptimierte Varianten dominieren, deren Architektur hinsichtlich der wesentlichen Systemparameter (wie Rechengeschwindigkeit, Funkdatenrate, Reichweite, Messgenauigkeit, Betriebsdauer) auf die konkrete Anwendung angepasst ist. Daher sind am Fraunhofer Institut für Zuverlässigkeit und Mikrointegration Funksensorsysteme mit alternativen Architekturen und unterschiedlichen Fertigungstechnologien entwickelt worden [4]. Tabelle 1 listet drei exemplarische Funksensorknoten gleicher Größe (1cm³) auf, die hinsichtlich Reichweite, Datenrate, Leistungsaufnahme sowie Aufbau- und Verbindungstechnologie variieren. Basierend auf diesen Vorarbeiten sollten praxisnah die Stärken und Schwächen der einzelnen Architekturelemente untersucht und kostengünstige Technologiekombinationen je nach erforderlichem Formfaktor ermittelt werden.

Tab. 1: Prototypvergleich verschiedener Architekturen

			
Chip-satz	TI MSP430F149 Nordic nRF2401	Atmel ATMega128L Chipcon CC1100	Atmel ATMega128L Chipcon CC1000
Funk	0,5 – 3m (1Mbits) Substratantenne	4 – 75m (500kbits) Ext. Drahtantenne	2 – 30m (38kbits) Miniloop-Antenne
AVT	Modulstapel mit Kontaktrahmen ICs als COB / MLF	Modulstapel mit Seitenwandkontakten ICs als MLF	Gefaltetes Flex ICs als FlipChip
Abmaße	10x10x10 mm³	10x10x10 mm³	10x10x10 mm³

Auch wenn die Siliziumintegration vielfach Kostenvorteile bringt, führt einen reine Einchiplösung als System-on-Chip selbst bei großen Stückzahlen nicht zum kostengünstigsten System. Die Anpassung der Prozesstechnologien für die Fertigung der einzelnen Komponenten bedeutet oft einen Kompromiss hinsichtlich der funktionellen Komponentenparameter. Beim komplementären Ansatz, dem System-in-Package, werden daher optimierte Komponenten mit sehr verschiedenen Herstellungsprozessen gefertigt und anschließend mit Hilfe der Aufbau- und Verbindungstechnik integriert.

Hinsichtlich der Fertigungskosten stellen planare Leiterplattenbaugruppen sowie Multichipmodule auf laminiertem FR4-Substrat die günstigste Modulintegrationstechnologie dar. Während eine Verdrahtung mit Standardleiterplatten zu recht großflächigen Baugruppen führt, sind derzeit Feinstleiterplatten mit Halbleiterchips im MLF-Gehäuse sowie Passive in der 0402-Bauform in der Regel die preiswerteste Variante. Zwar sind die Substratkosten pro Fläche etwas höher, aber die Modulfläche halbiert sich oftmals, so dass sich die Ausbeute pro Fertigungsnutzen mit entsprechenden Kostenvorteilen verdoppelt. Aufgrund der schwierigeren Testbarkeit von ungehäusten Mikrochips sowie teuerer Ausrüstung zur Bestückung kleinerer Bauformen, wie 0201 und 01005 für die passiven Bauelemente, ist eine weitere Miniaturisierung mit Mehrkosten verbunden. Jedoch aufgrund der höheren Schockfestigkeit und kompakteren Bauform kann sich dieser Zusatzaufwand in der Anwendung insgesamt rechtfertigen.

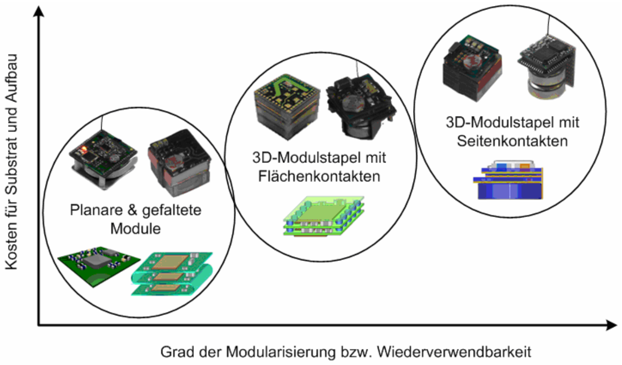


Abb. 2: Aufwand verschiedener Aufbauvarianten

Besonders kompakte Funksensorknoten werden ebenfalls durch die Technologien der 3D-Systemintegration auf Basis gestapelter bzw. gefalteter Module möglich. Die gefalteten Module weisen nahezu die gleichen Fertigungsprozesse auf wie die planaren Multichipmodule. Faltbare Substrate kosten jedoch gegenüber vergleichbaren FR4-Leiterplatten ungefähr das Doppelte. Fertigungstechnisch etwas aufwendiger sind Modulstapel mit flächigen Innenkontakten oder alternativ mit Seitenwandkontakten (Abb. 2). Durch Verwendung bereits getesteter Teilmodule stehen diesen Ansätzen dafür Kostenersparnisse gegenüber, so dass diese Modulstapel insbesondere bei moderaten Stückzahlen für spezifische Anwendungen vorteilhaft sind. Während bei den Modulstapeln mit flächigen Innenkontakten die Teilmodule enger aufeinander abgestimmt sein müssen, erlauben Funksensorknoten mit vertikalen Seitenwandkontakten den geringsten Abstimmungsaufwand. Die Teilmodule können dabei recht heterogene Abmaße aufweisen. Lediglich die Verdrahtung der Seitenwände muss beim Tausch von Teilmodulen neu festgelegt werden.

Zur Energieversorgung nutzen besonders kostengünstige Funksensorknoten meist Batterien, da sie eine vergleichsweise hohe Energiedichte besitzen. Für eine Lithium-Batterie (Bauform) mit einer Energiekapazität von 3Wh (Größe ca. 4cm³) bezahlt man 1 bis 3€. Die Verwendung von wieder aufladbaren Zellen in Kombination mit Energiewandlern (wie Solarzellen, Thermogenerator, Vibrationswandler) bietet für einige Anwendungen eine interessante Option, jedoch liegen die Kosten meist deutlich über denen einer Variante mit einer ausreichend großer Primärbatterie.

Die digitalen Funktionskomponenten werden in der Regel durch CMOS-Schaltungen als Mikrochip realisiert. Je nach Leistungsfähigkeit und erforderlichem Speicher wird eine Chipfläche von 10 mm² bis zu einigen cm² benötigt. Für einen Mikrocontroller mit integriertem Speicher und einer Chipfläche von 12mm² ergeben sich Kosten von ca. 72k€ für einen 12-Zoll-Wafer mit 10 000 ICs. Die anteiligen Maskenkosten betragen bei Standardkomponenten, die in Millionenstückzahlen gefertigt werden, weniger als 10 Prozent. Werden im Rahmen einer ASIC-Entwicklung jedoch nur geringere Stückzahlen benötigt, dann steigt dieser Anteil massiv, so dass der gleiche Mikrocontroller statt 0,72€ dann 1,08€ (Stückzahl 100k) bzw. 5,76€ (Stückzahl 10k) kostet. Die Entwicklungskosten wurden dabei noch nicht einmal berücksichtigt. Einen weiteren Kostenhebel bietet die Speichertechnologie insbesondere für den Programmspeicher der Prozessoren. In der Prototypentwicklung wird dazu gern vielfach beschreibbarer Flash-Speicher genutzt. Bei einer größeren Anzahl von Speicherzellen kann eine massive Reduktion der Chipfläche und damit die Stückkosten erreicht werden, wenn die Software als maskenprogrammierter ROM abgelegt wird. Dies erlaubt allerdings nachträglich keine Softwareänderungen mehr, was eine Herausforderung für die Planung der Entwicklungszeit ist.

Kostengünstige Funkarchitekturen werden ebenfalls größtenteils auf einem Mikrochip integriert. Je nach Schaltungskonzept und Prozesstechnologie variiert jedoch die Anzahl der Komponenten für die Außenbeschaltung. Superheterodyne Empfänger mit mehreren Zwischenfrequenzen stellen in der Relation zu ihren Kenndaten meist die energieeffizienteste Lösung dar. Dafür sind jedoch recht teure HF-Komponenten zur Filterung erforderlich, die sich besonders für Low-Cost-Sensorknoten ausschließen. Deutlich weniger externe Komponenten benötigen kostengünstigere Architekturen wie Low-IF- und Zero-IF-Empfänger, die das Funksignal auf sehr tiefe Frequenzen heruntermischen, so dass die Filterkomponenten auf einem IC integriert werden können. Solche Mikrochips erreichen Kosten in der Größenordnung von 1€. Die Anzahl der benötigten passiven Komponenten zur Störentkopplung, Impedanzanpassung und Filterung liegt zwischen 10 und 20, was zusätzliche Kosten von 15 bis 80 Cent erfordert. Die Realisierungsmöglichkeiten von Antennen sind recht vielfältig. Während im Substrat eingebettete Varianten wie Loop-Antennen nur wenige Cent kosten, können einige diskrete Aufbauten, wie beispielsweise kompakte Helixantennen, auch 0,25€ und mehr kosten.

Zur Erzeugung der Frequenzen ist in der Regel ein hermetisch gehäuster Quarz erforderlich, der eine Frequenzstabilität von mindestens 50ppm aufweisen muss. Aufgrund der aufwendigen Gehäusung kosten bei größeren Stückzahlen solche Komponenten ca. 0,50€, sofern der Automobil-Temperaturbereich von -40 bis 125°C erreicht werden muss. Ein weiterer Quarz kann für die Takterzeugung der Datenverarbeitung notwendig sein, wenn man den Synchronisationsaufwand, beispielsweise zur Koordination der Funkkommunikation im Adhoc-Netzwerk, verringern muss.

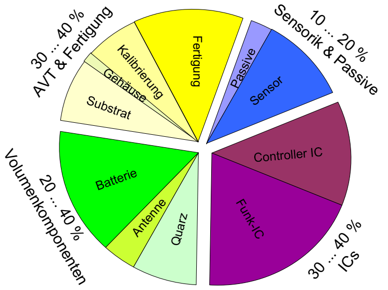


Abb. 3: Typische Kostenstruktur bei einer Massenproduktion

Eine typische Kostenverteilung für energieautarke Funksensoren, wie man sie derzeit bei den ersten Massenprodukten in Millionenstückzahl vorfindet, wurde in Abb. 3 dargestellt. Die Kosten für die Sensorik und den Kalibrierungsaufwand schwanken je nach Messgröße und -genauigkeit sehr stark. So können auf dem Chip integrierte Temperatursensoren mit einer Auflösung von 2°C den Bruchteil eines Cents kosten, während ein MEMS-Beschleunigungssensor in der Massenproduktion die 1€-Region erreicht. Für spezifische Anwendungen sind Sensorkosten im dreistelligen Euro-Bereich nicht ungewöhnlich, wie dies z.B. bei Beschleunigungssensoren mit einem Messbereich von 2000g gilt. Dabei sind die Entwicklungs- und Maskenkosten von einer deutlich kleineren Stückzahl zu tragen. Der Kalibrieraufwand ist auch vom Materialaufbau abhängig. So erfordern piezoresistive MEMS-Drucksensoren für die Reifendrucküberwachung eine aufwendige Kalibrierung bei einer höheren Temperatur, während dies bei kapazitiven MEMS-Drucksensoren nicht nötig ist [5]. Bei letzteren fallen jedoch höhere AVT-Kosten an, um der höheren thermomechanischen Stressempfindlichkeit zu begegnen.

4. Plattform zur Kostenanalyse

Zur Erarbeitung geeigneter Entwurfsmethoden wurde eine Prototypplattform entwickelt (Abb. 4), um die verschiedenen Technologien für Sensornetzwerke auf Netzwerk-, Architekur- und Prozessebene schnell und praxisnah verifizieren zu können. Ursprünglich diente die realisierte Infrastruktur zur parallelen Implementierung der Test- und Anwendungssoftware. Eine Auflistung der verschiedenen Teilmodule ist [6] zu entnehmen. Im Ergebnis können nun Funksensorknoten beschleunigt aufgebaut und für das konkrete Anwendungsszenario angepasst werden, um für eine gegebene Stückzahl das Kostenoptimum zu ermitteln. Das Spektrum reicht vom leistungsfähigen Prototypen, wie einer mobilen Funkkamera, bis zum minimalistischen Funktemperatursensor für Adhoc-Netzwerke, welcher derzeit hinsichtlich seiner Größe (Kantenlänge 6mm) einen Weltrekord darstellt [7].

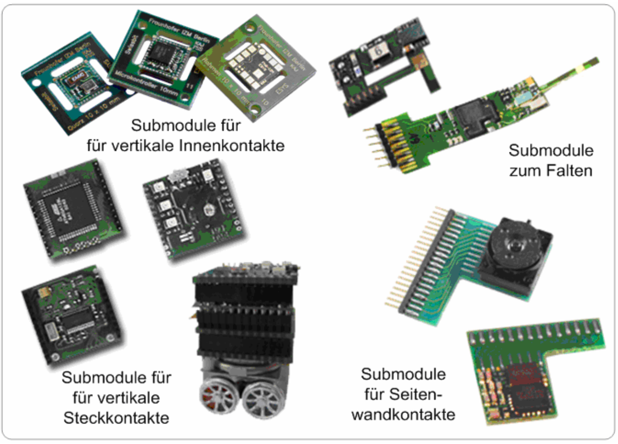


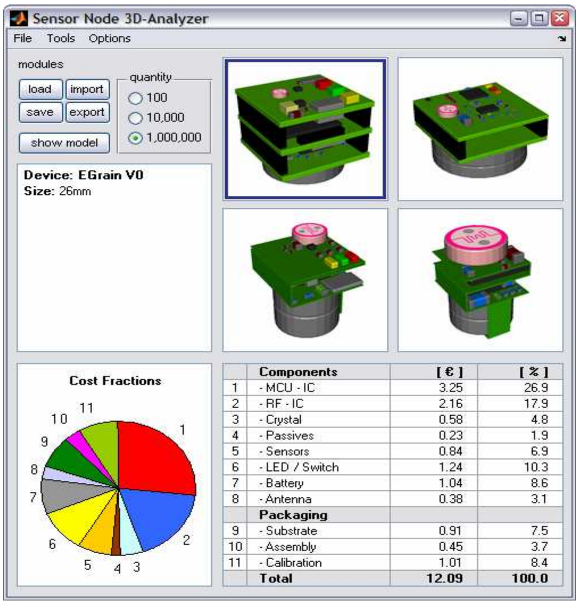
Abb. 4: Erarbeitete, modulare Prototypplattform

Für die physikalischen Entwurfsentscheidungen wurde ein Werkzeug implementiert, das die Partitionierung von Funktionsschichten und die räumliche Anordnung von Komponenten unterstützt. Die Kosten der Fertigungstechnologien werden dabei über partielle Verdrahtungsdichten eingebunden. Das Entwurfswerkzeug dient zur Erstellung von Kostenbilanzen. In der Phase der Entwurfskonzeption werden Komponentenkosten aus den Datensätzen bereits realisierter Funksensorknoten abgeschätzt. Im weiteren physikalischen Entwurf wird zunächst auf eine Festlegung der Verdrahtung verzichtet. Es wird lediglich eine Feinplatzierung aller Komponenten vorbereitet, um mit Hilfe von partiellen Verdrahtungsdichten eine Technologievorauswahl zu treffen. Diese basiert auf der Analyse der Netzliste für die betrachtete Komponentenanordnung. Zur Abschätzung von partiellen Verdrahtungsdichten wird ein 2,5D-Ansatz verwendet, wodurch zunächst die Anzahl der benötigten horizontalen und vertikalen Verdrahtungselemente ermittelt wird. Im Ergebnis werden die horizontalen Verdrahtungsdichten zur Auswahl der Substrattechnologien genutzt, um je nach konkretem Flächenbedarf die Substratkosten abzuschätzen (Tab. 2). Die Kostenmodelle für die Kontaktierungstechnologien – von der Oberflächenmontage diskreter Komponenten bis zur Einbettung von Mikrochips und Passiven in das Substrat – werden derzeit erarbeitet.

Tab. 2: Typische Substratkosten & Verdrahtungskapazitäten

Verbindungstechnologie	Flächenkosten [€/cm²]	Verdrahtungskapazität [1/mm²]
Dünnfilmsubstrate	1 ... 10	100 … 100 000
Dickfilmsubstrate	0,1 ... 2	10 … 100
Laminierte Substrate	0,01 ... 1	5 … 500

Daraus lassen sich die Strategien für die Grobverdrahtung ableiten. Durch eine geeignete Leiterbahnführung können partiell sehr hohe Verdrahtungsdichten ausgeglichen werden, um eine recht homogene Aufbau- und Verbindungstechnik einsetzen zu können. Alternativ kann man bei der Analyse der Verdrahtungskanäle aus Kosten- und Zuverlässigkeitsgründen versuchen, die Verdrahtungsdichte an den meisten Stellen deutlich abzusenken, um nur an wenigen kritischen Stellen die Technologien mit hoher Verdrahtungskapazität anzuwenden. Aus den entsprechenden Entwurfsentscheidungen hinsichtlich der Auswahl der Materialien und Prozessfolgen werden je nach Detailtiefe die Verdrahtungselemente geometrisch beschrieben und platziert. Im Ergebnis werden Kostenbilanzen erstellt, die auch die anteiligen Komponentenkosten ausweisen (Abb. 5).



○ 100
○ 10,000
● 1,000,000

Device: EGrain V0
Size: 26mm

	Components	[€]	[%]
1	- MCU - IC	3.25	26.9
2	- RF - IC	2.16	17.9
3	- Crystal	0.58	4.8
4	- Passives	0.23	1.9
5	- Sensors	0.84	6.9
6	- LED / Switch	1.24	10.3
7	- Battery	1.04	8.6
8	- Antenna	0.38	3.1
	Packaging		
9	- Substrate	0.91	7.5
10	- Assembly	0.45	3.7
11	- Calibration	1.01	8.4
	Total	**12.09**	**100.0**

Abb. 5: Entwurfswerkzeug für die Kostenanalyse

Zur Implementierung des Entwurfswerkzeuges wurde aufgrund der offenen und plattformunabhängigen Architektur die Interpretersprache Matlab^{TM} verwendet. Die Laufzeitumgebung von Matlab^{TM} stellt bereits viele Zusatzmodule mit Funktionen beispielsweise für die Kommunikation, die Datenerfassung und die Statistik zur Verfügung. Mittelfristig soll über eine Kostenmodellierung stark vereinfachter Prozesssequenzen eine teilweise automatisierte Erzeugung von Kostenbilanzen für verschiedene Zielparameter wie beispielsweise die Stückzahl erfolgen. Die entsprechende, regelmäßige Aktualisierung der Kostenanteile (Abb. 6) erlaubt eine geeignete Festlegung der Schwerpunkte für die weitere Kostenoptimierung, was das wesentliche Element einer ganzheitlichen, kostengetriebenen Entwurfsmethodik ist.

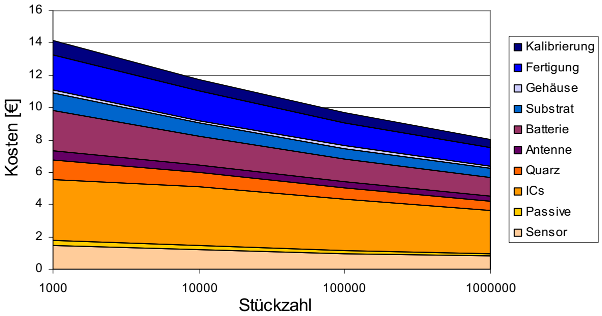


Abb. 6: Kostenanteile abhängig von der Stückzahl

5. Ausblick

Der Entwurf kostengünstiger Funksensorknoten erfordert die Analyse unterschiedlicher Zielkonflikte hinsichtlich der Systemfunktionalität, der Bauteilauswahl und der Fertigungstechnologien. Die vorgestellte Plattform zur Kostenanalyse, bestehend aus einer Testplattform für Funksensorknoten und einem Entwurfswerkzeug zur Technologieanalyse, soll zur Erarbeitung von Kostenmodellen für konkrete Anwendungsszenarien ausgebaut werden. Dazu wird eine Methodik zum kostenoptimierten Entwurf von Sensornetzwerken erarbeitet, die Kostenaspekte bereits im Vorfeld berücksichtigt und den Systementwurf zusammen mit den fertigungstechnischen Randbedingungen betrachtet.

Während der Schwerpunkt der bisherigen Arbeit weitestgehend auf den Modulintegrationstechnologien lag, soll die zukünftige Entwicklung von Funksensorknoten stärker das Potential der Waferintegrationstechnologien berücksichtigen. Zum Einen werden dadurch extrem kleine und robuste Funksensorknoten mit einer Kantenlänge von wenigen Millimetern möglich. Zum Anderen sind unter Beachtung der Ausbeute- und Testbarkeitsproblematik auch Kostenvorteile für größere Stückzahlen realistisch, um so die Preisregion von 2€ pro Funksensorknoten mittelfristig zu erreichen. Die erfordert jedoch noch Forschungsarbeiten zur Realisierung von Waferlevel-Spezialkomponenten insbesondere für Batterien, Quarze und Antennen.

6. Referenzen

[1] T. Scheiter: "Integration mikromechanischer Sensoren in einer CMOS/BICMOS - Prozeßumgebung", Dissertation, TU München, 1996

[2] K. Gilleo: "Area Array Package Design - Techniques in High-Density Electronics", McGraw-Hill, 2002, S.64 ff.

[3] C. Kallmayer et al.: "Packaging Challenges in Miniaturization" in Ambient Intelligence, Springer-Verlag, 2005

[4] L. Heinze et al.: „AVM Special Issue“ Zeitschrift für Telekommunikation Frequenz Band 58, Berlin, 2004

[5] R. Thomasius et al.: "Miniaturized Wireless Sensors for Automotive Applications”, 10th Int. Forum on Advanced Microsystems for Automotive Applications, Berlin, 2006

[6] M. Niedermayer et al.: "Miniaturization Platform of Wireless Sensor Nodes Based on 3D Packaging Technologies”, Proc. 5th Int. Conf. ISPN / SPOTS, Nashville, USA, 2006

[7] M. Niedermayer et al.: "Design for Miniaturization of Wireless Sensor Nodes Based on 3D-Packaging Technologies", 1st Int. Conf. on Smart Systems Integration, Paris, 2007

On Pursuit of Real-time and Reliability Guarantees in Wireless Sensor Networks

Thanikesavan Sivanthi
Institute for Communication Networks
Schwarzenbergstrasse 95, BA 4D
Hamburg, D-21073, Germany
thanikesavan.sivanthi@tuhh.de

Ulrich Killat
Institute for Communication Networks
Schwarzenbergstrasse 95, BA 4D
Hamburg, D-21073, Germany
killat@tuhh.de

ABSTRACT

Wireless sensor networks offer a distributed computing platform for many future applications, which have both timeliness and reliability requirements. Guaranteeing timeliness and reliability in a wireless sensor network is very challenging due to the unreliable and unpredictable characteristics of the network. This problem is addressed by very few research works, which focus on either of the requirements. This article presents a distributed problem solving perspective, which exploits the synergy between wireless sensor nodes, for achieving timeliness and reliability guarantees in wireless sensor networks.

1. INTRODUCTION

Wireless sensor networks offer a distributed computing platform for many emerging applications [23, 25], which require both timeliness and reliability requirements. Examples of such applications are traffic coordination, seismic monitoring, structural health monitoring etc. The timeliness guarantee signifies the timing constraint on a set of operations performed by the nodes of a wireless sensor network. The reliability guarantee implies the requirement on the probability of successful completion of operations, tolerating the failures in a wireless sensor network. Many recent works related to timeliness [3], [19], [12], [24], [17], [18] and reliability [2], [6] [10] [20], [11] requirements in wireless sensor networks consider either one of the requirements. A recent study for providing service differentiation and probabilistic guarantees for both timeliness and reliability is presented in [9]. The approach provides a multi-layer mechanism to differentiate flows based on their timeliness and reliability requirements. However, this approach requires the global knowledge of the probabilistic distribution of event locations and the required deadlines for finding the optimal configuration of the network parameter (SetSpeed [9]) during the off-line design phase. Unfortunately, in this approach the network parameter is static, whereas adaptable network parameters are desirable in order to opportunistically exploit the favorable network conditions and get a better service, as and when the opportunity presents itself. Further, the multi-layer mechanism is expensive due to the complex changes which should be implemented both at the routing and MAC layers. Most of the approaches listed above consider a bottom-up approach to provide timeliness and/or reliability guarantees, i.e. they first define the sub problems to be solved by the nodes, thereby expecting the nodes to educe a global solution. The appropriate way to approach the problem is to follow a top-down approach. In the top-down approach, the requirements are stated as a global problem, which can then be decomposed into sub problems that are solved by the nodes. The solution for the global problem is attained by solving the sub problems.

Due to shared wireless medium and the network failures [22], achieving timeliness and reliability guarantees in wireless sensor networks requires resolving the channel contention between the nodes and finding a route for the message from the source node to the sink node considering the failures and other messages which traverse along the vicinity of the route. This can be achieved by means of a joint scheduling and routing mechanism. The scheduling aspect deals with sharing of the available capacity between the nodes that are within the communication range. The routing aspect finds a route for the communication from a source node to a sink node, while taking care of the contending flows and failures along the route. The joint scheduling and routing in wireless sensor network to guarantee timeliness and reliability is a challenging problem. This is because the wireless channel is a shared medium and nodes can hear each other even though they might not be able to communicate. Consequently, interference might happen at a receiver which receives a message from a sender and if the receiver is within the carrier sense range of other active senders. However, there will be no interference if the receiver is outside the carrier sense range of the other active senders. This implies that at any time instant, some links can transmit messages in parallel without any interference. If the locations of the sensor nodes are known, it is possible to identify a set of cliques, where each clique consists of links that cannot transmit simultaneously, based on the signal to interference and noise ratio (SINR) [13] of the sensor nodes. If the SINR is above a certain threshold, then the message transmitted by an active link is not interfered by the other active link transmissions. In this case, the links belong to different cliques and can transmit messages simultaneously. Since, the links within the same clique share the capacity of the clique. The link flows in a clique should be scheduled, such that the sum of traffic demands of all link flows in a clique is less than or equal to the capacity of the clique. The traffic demand of a flow can be estimated as the ratio of its transmission time to its transmission deadline. A flow from a source to a destination node may traverse several hops or links. Subsequently, the flow should be routed along several cliques considering the current available capacity of the cliques. The scheduling and routing decisions should be made such that the timing and reliability requirements of all flows can be guaranteed. The following section presents the joint scheduling and routing in wireless sensor networks from a distributed problem solving perspective.

2. A DISTRIBUTED PROBLEM SOLVING PERSPECTIVE

Distributed problem solving is referred as the collective effort of semi-autonomous nodes to solve a problem in a distributed fashion [7]. In distributed problem solving, the global problem is decomposed into simple sub problems which are solved by a set of nodes. The solution for the global problem is obtained by solving the sub problems. The computing, communicating and storage abilities of the nodes of wireless sensor networks per se provide a platform for distributed problem solving. Consequently, the scheduling and routing problem can be solved in a distributed fashion by the nodes of a wireless sensor network. Each node solves a sub problem which collectively results in the required global behavior i.e. timeliness and reliability. The means to achieve sub problems from a global resource allocation problem is extensively studied in literature [21, 27], and finds its application in wired networks [15, 16, 8] and wireless networks [4, 26]. Unfortunately, none of these approaches have considered timeliness and reliability aspects in their distributed resource allocation. The basic idea of these approaches is to decompose the global problem into distributed sub problems, which are coordinated by means of some signaling [1]. The main challenge to solve the joint scheduling and routing problem in a decentralized fashion is the formalization of sub problems that can be solved by the individual nodes. The sub problem formalization should consider the different application and the network resource characteristics, some of which are described in the sequel.

2.1 Application Characteristics

A wireless sensor network application consists of a set of operations which are performed by the nodes of the network. The following are some of the application characteristics which should be considered when formulating the sub problem:

- **Release time and deadline:** These are the timing constraints of an application. The release time of the application is the earliest start time of the initial operation of the application and the deadline of the application is the latest completion time of the final operation of the application.

- **Application reliability:** An application reliability is defined as the minimum requirement on the probability of successful completion of the application, tolerating the different failures in the network.

- **Ordering:**: Some applications have strict precedence relationships between operations. The precedence constraints should be respected while making the scheduling and routing decision.

2.2 Resource Characteristics

The following are some of the network resource characteristics which should be considered when formulating the sub problem.

- **Node limitations:** The routing and scheduling decision should consider the limitations in the processing power, memory and energy of the sensor nodes.

- **Capacity limitation:** The channel contentions and interferences caused by simultaneous transmissions enforces limit on the bandwidth available to the sensor nodes.

- **Physical layer limitation:** At any time instant, a node can either be a sender or a receiver. Further, interference can happen at a receiver if it is within the carrier sense range of other active senders, which transmit messages simultaneously. The nodes which cannot transmit simultaneously form a clique.

- **Buffer size:** The buffer size has a significant impact on the end-to-end delay of an application. A proper dimensioning of buffer size is important to guarantee the timeliness and reliability of the application.

- **Number of MAC layer retransmissions:** The MAC layer provides a local reliability mechanism by retransmitting the failed packets until a maximum number of retransmissions is reached. The number of MAC layer retransmissions should be chosen such that it does not affect the end-to-end delay of the application, while not compromising the application reliability.

- **Failure rate of the nodes and channel:** The wireless channel is subject to errors due to different channel effects. Further, the nodes can fail due to random failures, power depletion and extreme environmental conditions. The failure rates of the node and the channel errors should be considered in order to find a reliable route from the source to the sink node.

The global problem can be formulated, considering the above application and resource characteristics, with an appropriate objective function, that reflect the timeliness and reliability requirements of the application, and a set of application and resource constraints. By means of a systematic decomposition of the global problem into sub problems, which are solved by the individual nodes, the joint scheduling and routing in wireless sensor network can be performed in a decentralized fashion. To that end techniques based on network utility maximization [14] or distributed constraint satisfaction [27] can be applied. In the former case, the convexity of the global objective and constraints should be ensured for the convergence of the solution to the global optimal solution. An appropriate decomposition principle, vertical or horizontal decomposition [5], should be chosen such that the solution derived by the sub problems converges to the optimal solution of global problem in limited time and with minimum signaling overhead. The objective, variable and constraints for a sub problem should be defined such that the collective behavior of all nodes should elicit the required global behavior. Each node solves the sub problem, assigns values for the variables based on the constraints, and communicates the values for the variables which appear in inter node constraints to the respective nodes. The collective behavior of all nodes ensure meeting the timeliness and reliability requirements of the application. Such a joint scheduling and routing mechanism based on the distributed problem solving paradigm, offers the promise for a scalable and adaptable distributed resource management, by means of which timeliness and reliability guarantees can be provided in wireless sensor networks.

3. CONCLUSIONS

This article presented a perspective to provide real-time and reliability guarantees in wireless sensor network. The underlying idea for solving the scheduling and routing problem in a distributed fashion by the nodes of the wireless sensor network is proposed.

4. REFERENCES

[1] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods.* Prentice Hall, 1989.

[2] S. Bhatnagar, B. Deb, and B. Nath. Service differentiation in sensor networks. In *Proceedings of the 4th International Symposium on Wireless Personal Multimedia Communications*, September 2001.

[3] M. Caccamo, L. Y. Zhang, L. Sha, and G. Buttazzo. An Implicit Prioritized Access Protocol for Wireless Sensor Networks. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*, December 2002.

[4] L. Chen, S. Low, and J. Doyle. Joint Congestion Control and Media Access control design for Ad Hoc Wireless Networks. In *Proceedings of the 24th Annual Conference of the IEEE INFOCOM*, March 2005.

[5] M. Chiang, S. Low, A. Calderbank, and J. Doyle. Layering as Optimization Decomposition: A Mathematical Theory of Network Architecture. In *Proceedings of the IEEE*, January 2007.

[6] B. Deb, S. Bhatnagar, and B. Nath. Reinform: Reliable information forwarding using multiple paths in sensor networks. In *Proceedings of the 28th IEEE International Conference on Local Computer Networks*, October 2003.

[7] E. H. Durfee. Distributed Problem Solving and Planning. In *Multi-Agent Systems and Applications*, July 2001.

[8] K. Eger and U. Killat. Resource Pricing in Peer-to-Peer Networks. *IEEE Communications Letters*, 11(1):82–84, 2007.

[9] E. Felemban, C.-G. Lee, and E. Ekici. MMSPEED: Multipath Multi-SPEED Protocol for QoS Guarantee of Reliability and Timeliness in Wireless Sensor Networks. *IEEE Transactions on Mobile Computing*, 5(6):738–754, 2006.

[10] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin. Highly-resilient, energy-efficient multipath routing in wireless sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(4):11–25, 2001.

[11] Q. Han, I. Lazaridis, S. Mehrotra, and N. Venkatasubramanian. Sensor Data Collection with Expected Reliability Guarantees. In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications Workshops*, March 2005.

[12] T. He, J. A. Stankovic, C. Lu, and T. F. Abdelzaher. SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, May 2003.

[13] H. Karl and A. Willig. *Protocols and Architecture for Wireless Sensor Networks.* Wiley, 2005.

[14] E. P. Kelly. Charging and Rate Control for Elastic Traffic. *European Transactions on Telecommunication*, 8(1):33–37, 1997.

[15] F. P. Kelly, A. Maulloo, and D. Tan. Rate control in communication networks: shadow prices, proportional fairness and stability. *Journal of the Operational Research Society*, 49(3):237–252, 1998.

[16] R. J. La and V. Anatharaman. Utility-based rate control in the Internet for elastic traffic. *IEEE/ACM Transactions on Networking*, 10(2):272–286, 2002.

[17] H. Li, P. J. Shenoy, and K. Ramamritham. Scheduling Messages with Deadlines in Multi-Hop Real-Time Sensor Networks. In *Proceedings of the 11th IEEE Real-Time and Embedded Technology and Applications Symposium*, March 2005.

[18] K. Liu, N. Abu-Ghazaleh, and K.-D. Kang. JiTS: Just-in-time scheduling for real-time sensor data dissemination. In *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications*, March 2006.

[19] C. Lu, B. Blum, T. Abdelzaher, J. Stankovic, and T. He. RAP: A Real-Time Communication Architecture for Large-Scale Wireless Sensor Networks. In *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium*, September 2002.

[20] M. Marina and S.Das. On Demand Multipath Distance Vector Routing in Ad Hoc Networks. In *Proceedings of the 9th IEEE International Conference on Network Protocols*, November 2001.

[21] D. P. Palomar and M. Chiang. A Tutorial on Decomposition Methods for Network Utility Maximization. *IEEE Journal Selected Areas in Communication*, 24(8):1439–1451, 2006.

[22] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou. Real-Time Communication and Coordination in Embedded Sensor Networks. In *Proceedings of the IEEE*, July 2003.

[23] T. Arampatzis and J. Lygeros and S. Manesis. A Survey of Applications of Wireless Sensors and Wireless Sensor Networks. In *Proceedings of the 2005 IEEE International Symposium on Mediterrean Conference on Control and Automation*, June 2005.

[24] S. Wang, R. Nathuji, R. Bettati, and W. Zhao. Providing Statistical Delay Guarantees in Wireless Networks. In *Proceedings of the 24th International Conference on Distributed Computing Systems*, March 2004.

[25] A. Wheeler. Commercial Applications of Wireless Sensor Networks Using ZigBee. *IEEE Communication Magazine*, 45(4):70–77, 2007.

[26] Y. Xue, B. Li, and K. Nahrstedt. Optimal Resource Allocation in Wireless Ad Hoc Networks: A Price-Based Approach. *IEEE Transactions on Mobile Computing*, 5(4):347–364, 2006.

[27] K. Yokoo, E. H. Durfee, T. Ishida, and K. Kuwabara. The Distributed Constraint Satisfaction Problem: Formalization and Algorithms. *IEEE Transactions on Knowledge and DATA Engineering*, 10(5):673–685, 1998.

iSense: A Modular Hardware and Software Platform for Wireless Sensor Networks

Carsten Buschmann
coalesenses GmbH
Röntgenstraße 28
23562 Lübeck, Germany
buschmann@coalesenses.com

Dennis Pfisterer
coalesenses GmbH
Röntgenstraße 28
23562 Lübeck, Germany
pfisterer@coalesenses.com

ABSTRACT

We present iSense, a modular hardware and software platform for wireless networks that is intended for both industry and research applications. The hardware is arranged around a core module with an IEEE 802.15.4 compliant radio, a 32-bit RISC controller running at 16MHz, 96kbytes of memory, a highly accurate clock and a switchable power regulator. It can be combined with a number of sensor modules, different power sources, a gateway and I/O module and various others. The hardware is supplemented with a modular operating and networking firmware that is based on object oriented programming. A comprehensive development environment is available free of charge.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design; H.4 [**Information Systems Applications**]: Miscellaneous; C.3 [**Special Purpose And Application-Based Systems**]: Real-Time and embedded systems; C.4 [**Performance of Systems**]: Design Studies

General Terms

Design, Experimentation, Performance

Keywords

WSN, sensor network, modular software and hardware platform

1. INTRODUCTION

During the last years, wireless sensor networks have attracted ongoing research attention. Recently, efforts increasingly move from fundamentals to research in how applications could be constructed.

This shift can be interpreted as a sign of maturity: wireless sensor networks seem to become ready for real life applications. However, these demand for commercial platforms. Most of the hardware developed for research purposes is not appropriate for commercial applications for a number of reasons. They lack robustness and reliability and are not really extensible. They come without declarations of conformity or guaranteed properties and are provided just “as is”. All in all, still only few commercial platforms [2, 11, 10, 9] are available.

The operation domain of wireless sensor networks is extremely broad. Applications range from ubiquitous computing with its demand for compact systems or fixed installations profiting from wireless communication driven by wall mount adapters to long-time ad-hoc installations that need high-capacity batteries.

Hence a wireless sensor networking platform should

- first of all be flexible and extensible in software and hardware to meet as many of the diverse application requirements,
- be robust and compact,
- offer a due proportion of performance and storage capacity on the one hand and current consumption on the other, combined with ultra-low power sleep modes,
- provide a reliable and standard compliant radio and
- offer exact timing and an accurate clock.

We introduce iSense, a modular hardware and software platform that meets all these requirements.

2. MODULAR HARDWARE...

In order to fit a wide variety of application demands the iSense hardware platform is made up of a number of a number of modules that can be combined in various ways. Like this, functionality can be easily rearranged, and new features can be added by appending new modules.

Currently, a core module comprising computation and wireless communication, different energy modules, a gateway module for interfacing with computers and an IO-module are available. A number of sensor modules are under development. This module structure is visualized in Figure 1.

The heart of the hardware platform is the *core module.* It accommodates the wireless micro controller Jennic JN5139, a chip that combines the controller and the wireless communication transceiver in a single housing.

The controller provides 32 bit RISC computation and is running at 16 MHz. It comprises 96kbytes of memory that are shared by program code and data. The advantage of this choice is that memory consumption of program code and data can be traded. Opposite to other controllers where the

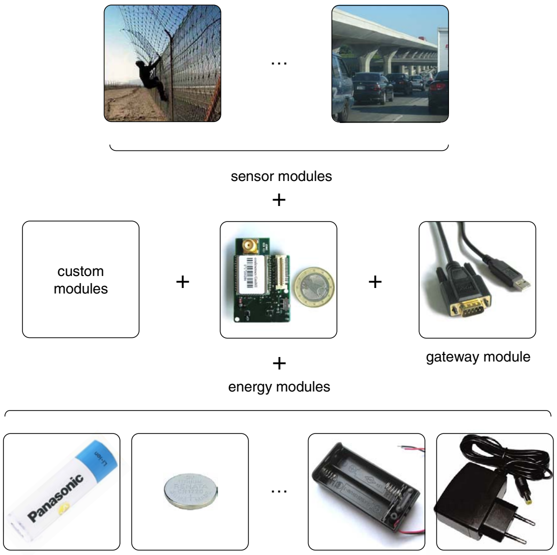


Figure 1: Hardware modules of the iSense platform.

user is limited to a certain amount of data and code memory, free choices that are only bounded by the sum of both become possible here.

The radio part complies with the IEEE 802.15.4 standard [7]. It achieves a data rate of 250kBits/s and provides hardware AES encryption. With a receive sensitivity of -97dBm and a transmit power tunable between -60dBm and +3dBm it reaches ranges of up to 500m. Apart from this standard version that is equipped with an SMA antenna connector, derivatives with an integrated antenna for especially compact systems or with an additional power amplifier for ranges of up to 2km are available. All three systems are ZigBee-ready.

A common quandary in design is whether or not to use a voltage regulator. It has the advantage that operation with voltages lower than the required one is possible, but the regulator inherently wastes energy. This is especially bad as it also wastes current if the voltage would be high enough and the regulator would not be required. To resolve this problem, we decided to combine the measurement of the supply voltage with the possibility to bypass the regulator by a software switch. Like this, the regulator usage can be omitted when not required but is available when the supply voltage drops.

To enable long, but still synchronous sleep and wakeup cycles, the module is equipped with a high precision clock (error < 20ppm). It features a software switchable LED for debugging purposes.

There is a 34 pin connector on both sides of the module where other modules can be attached to the core module. It can supply up to 500mA to other modules.

The controller can be programmed in various ways. While over-the-air programming (OTAP) is possible and considered to be the standard procedure, the program can also be transfered via the gateway module (discussed later) or using a special programming adapter that mates with corresponding pads on the module.

The core modules' sleep current goes as low as 10μA. In full operation the micro controller uses about 9mA, the radio part 29mA. The module can be powered by a wall mount adapter or a standard battery holder, by one of the power modules or via the USB interface of the gateway module.

Two different *power modules* are available. The *lithium-ion module* combines a high capacity rechargeable battery with a charge controller and a battery monitor that tracks the voltage as well as the current flows from and to the battery. The integrated charge controller enables in-system-charging using the core module's wall mount adapter or the gateway module's USB power. The *coin cell module* is intended for particularly compact systems. It holds one CR2477 battery and features a battery monitor for exact battery level information, too.

The *gateway module* is intended for debugging and interfacing data to other networks. Apart from LEDs, buttons and a user adjustable potentiometer, it features a USB and a RS232 interface. An additional *I/O module* offers convenient access to the compact bus connector via 2.54mm spaced pins and hence enables fast mock up of adapter boards.

Apart from the aforementioned central modules, a number of sensor boards are under development. The *security sensor module* comprises a passive infrared (PIR) sensor and 3-axis accelerometer. The former features a range of approx. 10m at an angle of 130 degrees while the latter is adjustable to ranges of ±2g and ±6g. The possibility to attach a camera is planned. The *vehicle detection sensor module* accommodates 2-axis anisotropic magneto-resistive (AMR) sensors. They perceive changes in the earth's magnetic field induced by large ferro-electric masses and can hence detect vehicles at distances of up to approximately 5m. Other modules such as a board with a temperature and light sensor are in preparation.

3. ... AND MODULAR SOFTWARE

The extremely flexible and modular hardware design of the iSense platform requires the same flexibility of the software that drives the individual sensor nodes. The iSense software has been designed with maximal flexibility in mind while allowing for a professional, industry-grade development experience.

One of the fundamental design guidelines is to use state of the art programming methods that are well understood by a large user community. Advanced techniques such as object oriented C++ programming and dynamic memory allocation, that are usually not available in sensor network environments, allow for a rapid and error-avoiding development process. In addition, run-time memory allocaton allows for appropriately sized buffers etc. and hence increases memory efficiency. iSense ships with a tiny and lean STL-like implementation that relieves application developers from dealing with recurring and error-prone tasks. It provides implementations for standard containers such as lists, sets and maps.

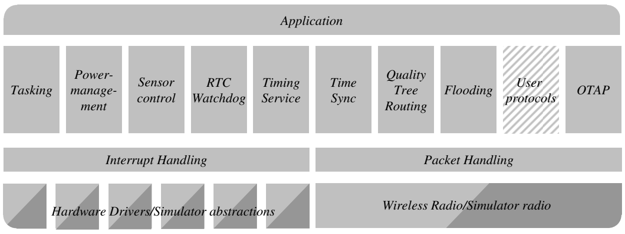


Figure 2: Modular software structure of iSense.

As a result, the development of applications for the iSense platform is completely based on well-known technologies and does not require any proprietary extensions. Hence, the extremely flat learning curve enables a rapid application development that benefits from existing domain expertise of the developers and a plethora of available tools.

Just like its hardware counterpart, the software platform is organized in a set of modules where each of these functional entities provides a highly specialized service to the application. When developing an application, users assemble a subset of the available modules to a lean operating system that contains exactly the required functionalities. A web-based configuration dialog operated by the coalesenses GmbH allows for an easy, user-friendly selection of these functionalities and subsequently delivers the custom-tailored operating system instance to the user. Figure 2 depicts the overall architecture of the iSense software platform. It is comprised of four distinct building blocks: a hardware abstraction layer on the bottom, operating system functionality and networking support in the middle and the actual user-defined application at the top of the figure.

The hardware abstraction layer (HAL) encapsulates hardware functionality and hides intricate details of the underlying hardware by providing a focused and straightforward application programmer's interface to the upper layers. Abstractions for interacting with A/D & D/A converters and I/O interfaces (e.g., serial UARTs, I^2C and SPI) are available as well as for timers, permanent storage and the wireless interface. A typical usage scenario of the HAL is the integration of the sensor modules described above. These are usually connected to one of the I/O-pins or bus systems and are easily integrated into the iSense software using the HAL functionality. Using this architecture, all modules above the HAL are independent of a particular hardware platform. Application code developed inside this framework is ready-to-run on any platform that provides an implementation of the iSense-API. Currently, it is available in two flavors: one for the iSense hardware platform and one for the simulation framework Shawn [8, 4]. This allows developers to test their implemented functionality inside a simulation framework before the application is actually deployed on an iSense node thus significantly increasing the development speed.

On top of the hardware abstractions, the iSense framework provides operating system functionalities that ease application development through an event-driven model. Applications receive call-backs whenever events occur for which the application has registered itself. These events occur either application-driven (e.g., when timers elapse) or hardware-driven (e.g., when input signal of A/D converters change or data is received on one of the I/O-systems). For the application-driven events, the iSense operating system offers two distinct choices. Whenever high timing precision is required, a timing service allows callbacks to be handled uninterruptible and with minimal delay in the interrupt context. Functionality that is not time-critical can register with the tasking service that can be interrupted by the timing service. Finally, the operating system is responsible for conserving the scarce energy resources of a sensor node whenever possible. If desired by the user, the power management infrastructure can put the device into one of the different low power modes.

Besides the functionality that operates strictly local on each single sensor node, one key ingredient of WSNs is wireless communication and consequently a subset of the iSense-modules tackles especially this issue. The HAL already contains convenient abstractions from the details of the wireless interface on top of which the networking support of iSense provides a number of powerful, sophisticated services. It is comprised of routing, time synchronization and over-the-air programming modules. Typical WSN applications require that data is communicated well beyond the communication range of a single sensor node and iSense offers two routing modules that cover a large portion of the design space for sensor network applications. First, a controlled flooding implementation that provides an error-resilient, robust method for conveying data to a set of nodes that is within an n-hop neighborhood of the sending node. Second, a tree-routing module that enables data transfer from the network to one or more sinks. The metric for the link choices is based on packet losses in order to maintain routes with high delivery rates and hence increase network robustness.

It is often vital for WSN applications that the clocks of all nodes in a sensor networks run synchronized, e.g. for data aggregation or sleep/wake-scheduling. This integral feature is integrated as a module in iSense and developers use this functionality and can rely on accurate clocks with a deviation of less than 1ms over 10 hops. Another particularly important module provides the ability to re-program an already deployed sensor network wirelessly. This over-the-air programming (OTAP) provides for a flexible development and operation of sensor networks as wired connections are superfluous and no manual mass-programming is necessary.

Apart from the features of the iSense hardware and software, a comprehensive and accepted development environment is a vital property for successful application development. The iSense software and the development tool-chain are available free of charge and use widely accepted and popular tools such as the Gnu Compiler Collection [5] (GCC) and the Eclipse [3] development framework. Furthermore, iSense provides iShell, a convenient means to interact with the sensor network. It combines the functionality of a serial terminal, serial and over-the-air programming of sensor nodes as well as a flexible plug-in system for integrating user-defined functionality such as data analysis or wireless monitoring. iSense and iShell provide an optional (de)-multiplexing service on the serial link. This enables applications to use a number of different, independent data streams, e.g. separating debugging output from different application data streams.

4. CONCLUSION

In this paper we present our modular hardware and software platform called iSense. It features a number of unique, industry-grade properties. Its modular approach allows creating custom-tailored instances of the hardware as well as the software that comprise exactly the required functionality for a particular application. The small form factor and the available outdoor-capable housings allows for robust and reliable deployments. A low-power design enables a long autonomous operation that – combined with the ability for wireless reprogramming – results in an easy to operate and maintain sensor network. The IEEE 802.15.4 standard-compliant core features a ZigBee-ready radio, hardware encryption and high data rates. The flexible software API with rich a variety of software modules and the use of well-known development tools enables rapid development of market-ready applications. Future work will include support for TinyOS [6] (expected this year) and the integration of the network visualizer SpyGlass [1] into iShell.

5. REFERENCES

[1] C. Buschmann, D. Pfisterer, S. Fischer, S. P. Fekete, and A. Kröller. SpyGlass: A wireless sensor network visualizer. *ACM SIGBED Review*, 2005.

[2] Crossbow Technology Inc. MICAz wireless measurement system. http://www.xbow.com, 2004.

[3] Eclipse Foundation. Eclipse - an open development platform, 2001. http://www.eclipse.org/.

[4] S. P. Fekete, A. Kröller, S. Fischer, and D. Pfisterer. Shawn: The fast, highly customizable sensor network simulator. In *Proceedings of the Fourth International Conference on Networked Sensing Systems (INSS 2007)*, June 2007.

[5] Free Software Foundation, Inc. Gnu Compiler Collection (GCC), 1984. http://gcc.gnu.org/.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *SIGOPS Oper. Syst. Rev.*, 34(5):93–104, 2000.

[7] IEEE 802.15 Working Group. IEEE 802.15 WPAN Task Group 4 (TG4). http://www.ieee802.org/15/pub/TG4.html.

[8] A. Kröller, D. Pfisterer, C. Buschmann, S. P. Fekete, and S. Fischer. Shawn: A new approach to simulating wireless sensor networks. In *Design, Analysis, and Simulation of Distributed Systems 2005, Part of the SpringSim 2005*, pages 117–124, April 2005.

[9] Moteiv Corporation. Tmote Sky. http://www.moteiv.com/products-tmotesky.php, 2005.

[10] J. Polastre, R. Szewczyk, and D. E. Culler. Telos: enabling ultra-low power wireless research. In *IPSN '05: Proceedings of the 4th international symposium on Information processing in sensor networks*, pages 364–369, 2005.

[11] ScatterWeb GmbH. ScatterNode. http://www.scatterweb.com/, 2005.

Accurate Timing in Sensor Network Simulation

Muhammad Hamad Alizai, Olaf Landsiedel, Klaus Wehrle
Distributed Systems Group
RWTH Aachen
{hamad.alizai, olaf.landsiedel, klaus.wehrle}@rwth-aachen.de

ABSTRACT

Accuracy, speed and scalability are the basic requirements of sensor network simulation. To comprehend the accurate behavior of resource constrained embedded systems such as sensor nodes it is important in simulations to model the time-dependent behavior of the system. In this paper we present our extensions of TOSSIM [2] – a widely used event-driven simulation environment for sensor networks – to enable its simulation models to capture the time-accurate behavior of sensor networks by exhibiting timing and interrupt properties of the platform dependent source-code. By mapping the device specific code with the simulation model, we can derive the timing of functional code blocks. As result of such a mapping it is possible to determine the time when a certain code block gets executed and the time the execution takes, eliminating the need of expensive cycle-accurate instruction level simulators with limited speed and restricted scalability.

1. INTRODUCTION

Simulation indisputably remains one of the most important tools for analyzing, evaluating and validating system design. The importance of simulation is further aggravated for systems having an embedded nature, high deployment costs, or possessing unobservable fast interactions yet important to validate the system design. Sensor Networks with their distributed behavior, strenuous deployment requirements, constrained resources, and invisible and unpredictable interaction between the sensor nodes poses additional demands on their simulation.

In the past few years a great deal of effort has been invested in the design and development of simulators for sensor networks to embrace the special requirements imposed by the highly distributed and dynamic nature of sensor networks. Unfortunately all these efforts have made compromises over different attributes of the simulation, for example, accuracy has been compromised over scalability and vice versa. SWAN [4], SensorSim [5], and SENS [6] are examples of sensor network simulators which compromise scalability over accuracy by using nonfigurative models of the sensor nodes. Such simulation models only contribute to quantify network delays, throughputs, packet collisions, power usage and the effect of several power management schemes [3]. However, these models do not reveal the timing and interrupt properties of applications, the operating systems, and hardware components.

ATEMU [7] and Avrora [3] on the other hand are cycle-accurate instruction level simulators for sensor networks with the most expressive simulation models. Nevertheless, they compromise the scalability and performance/speed. ATEMU is 30 times slower than TOSSIM [3], and its poor performance limits its scalability to 120 nodes. Avrora shows better performance measures than ATEMU with reasonably good speed for small number of sensor network nodes but it is still 50% slower than TOSSIM. The performance measures of Avrora have been calculated on a 16 processor machine, not easily accessible to normal end-users and developers. Avrora exhibits typical performance bottlenecks of instruction level simulators when run on customary end-user machines, especially, when several avrora-monitors are enabled for detailed analysis of the sensor network behavior.

Our goal is to provide time accurate simulation for sensor networks at the basic-block granularity (i.e. sequence of instructions with a single entry point, single exit point, and no internal branches) of the source code without compromising the speed and scalability, and hence eliminating the need to use expensive instruction level simulators. We extend TOSSIM to exhibit the timing and interrupt properties of sensor network code without destroying its performance and scalability advantages.

2. TOSSIM

TOSSIM is an extremely fast sensor network simulator scalable to thousands of sensor network nodes. It compiles directly from the TinyOS source code into the simulation environment by adding an alternative compilation target. The fact that it compiles directly from the platform dependent source-code makes it more expressive than SensorSim, SWAN, and SENS. TOSSIM only requires to model the low level components responsible for hardware interaction such as low level access to timers, communication channels, sensors, and the radio. These low level components expose the real hardware and are placed at the Hardware Presentation Layer (HPL) of the TinyOS-2.0's platform abstraction model [8]. TOSSIM also benefits from the event-based, component oriented programming model of TinyOS by translating the asynchronous-events and hardware interrupts into discrete simulator events which drive the simulation.

TOSSIM's level of detail was sufficient to measure packet losses, packet CRC failure rates, and the length of the send queue for up to 8,192 nodes [3]. However, TOSSIM's compilation steps lose the fine-grained timing and interrupt properties of the code that are extremely important for a time-accurate simulation [3].

We address these problems by exploiting the fact that TinyOS runs the same code (except the small platform dependent HPL layer) in simulation and on the sensor network hardware. This feature of TOSSIM enables to create a mapping between the platform dependent binary and the simulation code. We use Mica-2 as our target platform. Our method is to (1) analyze the platform dependent assembly program and compute the cycle count corresponding to each basic-block; (2) assign a priority number to every simulator event to enable TOSSIM to model the interrupt and preemption behavior of the real hardware; (3) extend the C-source code generated by TOSSIM to (a) increment the simulation clock at the start of every source-code line by the cycle count information obtained in the first step, hence, enabling the TOSSIM to exhibit the timing properties of the code. (b) Re-schedule the TOSSIM event queue at the start of every basic-block on the basis of new timing information obtained, and also on basis of the assigned interrupt priority of each event in the simulation queue to model the masking and preemption properties of the hardware interrupts in the simulation infrastructure.

Our approach is different from CPU-profiling approach in PowerTOSSIM[1] – an extension of TOSSIM for simulating the power consumption of sensor networks, which does offline processing to obtain the cycle counts for CPU power profiling. We, on the other hand embed TOSSIM with the information obtained from the assembly of Mica-2 motes to perform online adjustments in the simulation clock and event queue.

3. TIME ACCURATE SIMULATION

This section describes the details of the time accuracy related problems in TOSSIM and our approach to address these problems.

3.1 Timing Discrepancy

TOSSIM captures the TinyOS event-driven concurrency model at interrupt and task granularity [9], and it has a single queue both for the tasks and the events. The simulation is triggered by the events and the tasks in the TOSSIM event-queue which is sorted in the increasing time order. TOSSIM adjusts its simulation clock at the start of the execution of every event by assigning the time stamp of the recently popped event from the queue to the simulation clock. Events and tasks take zero execution time in TOSSIM as the simulation clock remains unadjusted during the course of execution; hence, TOSSIM loses the fine-grained time accuracy of the code. This imperfection of TOSSIM introduces even more problems, for example, TOSSIM is unable to differentiate between a task requiring a large number clock cycles to transmit several bytes over the radio from a task requiring few clock cycles just to blink an LED attached to the microcontroller pin or to report a timer fire.

The execution time of an event or task may also affect the timing of next events or tasks in the queue as shown in Figure-1. For example, if TOSSIM is currently executing an event associated with high priority interrupt and there is an immediately scheduled task or event representing a low priority interrupt, then its execution time should be delayed – timestamp should be readjusted, at least until the execution of current event is finished. TOSSIM, because of its imperfection to track the system time during execution of an event, is unable to capture this priority based interrupt behavior of the hardware which masks the less priority interrupt or delays the execution of tasks while handling a high priority interrupt. Similarly, in TOSSIM the simulator events run atomically one after another, therefore, unlike on real hardware, interrupts cannot preempt one another [9]. On the other hand, long tasks – tasks requiring several clock cycles to execute, delay the execution of other tasks and can be preempted by events, but TOSSIM is unable to model such behaviors as show in Figure-2.

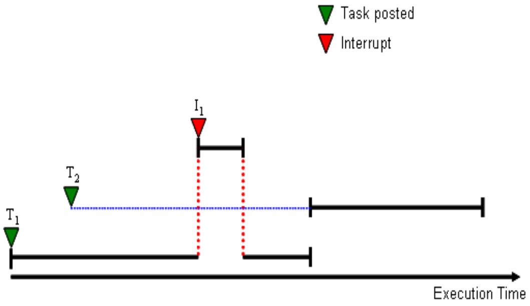


Figure 1. TinyOS event handling and execution flow

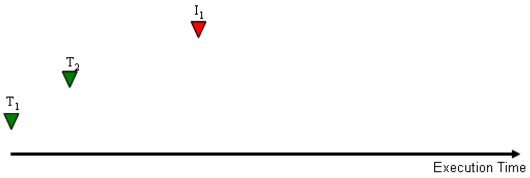


Figure 2. TOSSIM execution flow

3.2 Our Solution

Our approach to solve this timing discrepancy involves three steps.

3.2.1 Basic-block Mapping

We address the timing discrepancy of TOSSIM by enabling it to exhibit the timing properties of the code at the basic-block granularity. We achieve this by creating a mapping between the TOSSIM's C-source code and the assembly of platform dependent code (Mica-2 in our case). Our mapping technique is similar to PowerTOSSIM.

TinyOS uses the NesC compiler to compile the TinyOS component graph to a single C-source file, which in effect is then compiled into the binary for the specified target platform through appropriate C-compiler (i.e. gcc for TOSSIM and avr-gcc for Mica-2). We use the avr-objdump utility with appropriate options to obtain the assembly of Mica-2 platform which also contains a mapping of the assembly instructions to the original nesC source-code. We parse this assembly file to obtain the cycle counts corresponding to the basic-blocks of the source-code. On the other hand, we use the C-source file generated by the nesC compiler for the TOSSIM platform. The C-source file of TOSSIM also provides the mapping between C-source code and the original nesC source code, thus, enabling the mapping between the platform dependent assembly and the TOSSIM's C-source file.

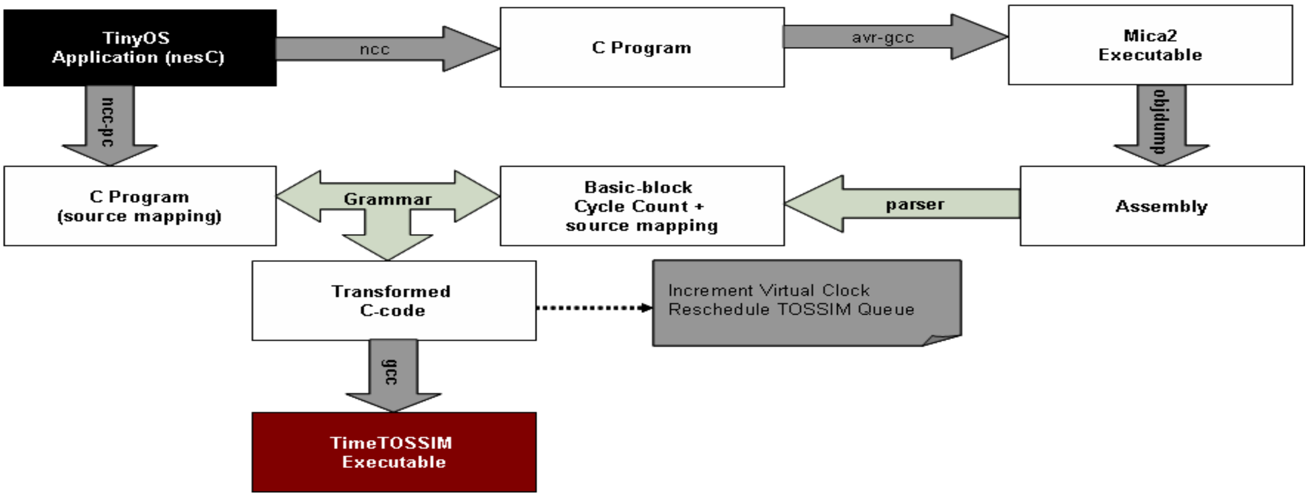


Figure 3. Block Diagram: Extending TOSSIM to capture time-accurate behavior of the system

We parse the C-source file of TOSSIM using ANTLR's [10] GNU-C grammar to perform source-to-source transformation. Our transformation includes (1) extending the C-source file by adding functions that increment the simulation clock and perform online adjustments in the TOSSIM Queue; (2) adding a call to these functions at the start of every basic-block. These transformations enable TOSSIM to exhibit the timing properties of application at the basic-block granularity. The whole process of extending the TOSSIM is shown in Figure-3.

3.2.2 Rescheduling the TOSSIM Event Queue

By extending TOSSIM to incorporate the timing properties of the system at basic-block granularity also enables us to reschedule the TOSSIM queue and intensify TOSSIM even further to exhibit the interrupt properties of the hardware. We do this by rescheduling every event and task in the TOSSIM queue (hereinafter referred to as target event) whose time-stamp is less than the simulation clock time. Additionally, we assign interrupt priority numbers to every event in the TOSSIM Queue. Tasks are assigned zero interrupt priority. Rescheduling the event queue introduces two possibilities; (1) either the target event in the event-queue has an interrupt priority less than or equal to the current event or task being executed. In this case we increment the time-stamp of the target event by the amount of time needed to execute the current basic-block; (2) or the target event represents a high priority interrupt. In this case we interleave the execution of the current event or task (i.e. at the start of the basic-block) and start the execution of the target event in the queue with high priority.

3.2.3 Hardware Component Profiling

The NesC compiler, when compiling for TOSSIM, replaces the components at the HPL of platform abstraction architecture with their corresponding reimplementation for TOSSIM. Our transformations work very well when the TOSSIM is executing the platform independent part of the application code (i.e .common for TOSSIM and Mica-2 platform), and we achieve 100% basic-block mapping. But this basic-block mapping fails and we loose our granularity once the TOSSIM enters the execution of its own reimplementation of hardware related components.

We address this problem by profiling the hardware related components. We observed that the behavior of these low level components, that expose the hardware, is static. For example, it always takes the same amount of cycles to turn an LED On or Off. It is also possible to do some manual mapping between the components that share the same algorithmic properties and execution flow but their execution time is not static. For example, TOSSIM has its own scheduler but its execution flow is analogous to the TinyOS Scheduler, nonetheless, execution time of the scheduler is not static because it performs some context switching as well as processes long queues of tasks. We do manual mapping between the TinyOS scheduler and the TOSSIM Scheduler to maintain the same basic-block granularity and timing resolutions that we desire to achieve.

4. CONCLUSION AND FUTURE WORK

In this paper we discussed the importance of timing properties of the source code in simulations. We showcased a distinct technique and demonstrated how time-accurate simulation can be achieved using this approach as described in section-3.2. It enables to model the time-accurate behavior of the system at the basic-block granularity without using the non scalable and low performance instruction level simulators.

We are still in the active development phase of our work. Intense evaluation is yet to be performed, though the initial results are very promising. We achieve a beyond 99% time accuracy with basic prototype applications like Blink and TestScheduler. We plan to rectify TOSSIM's hardware models including timers and radio to model the original hardware accurately. TOSSIM is also unable to model the behavior of atomic statements – block of statement that run uninterrupted. Access to the application code at the basic-block level can also help in accurately modeling the atomic statement blocks in the code.

5. REFERENCES

[1] Victor Schnayder, Mark Hampstead, Bor-rong Chen, Geoff Werner Allen, and Matt Welsh. *Simulating the power consumption of large-scale sensor network applications*. In Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys) 2003, Nov. 2003.

[2] P. Levis, N. Lee, M. Welsh, and D. Culler. *TOSSIM:Accurate and scalable simulation of entire TinyOS applications*. In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys) 2003, Nov. 2003.

[3] Ben Titzer, Daniel Lee, and Jens Palsberg. *Avrora: Scalable Sensor Network Simulation with Precise Timing*. In Proceedings of IPSN'05, Fourth International Conference on Information Processing in Sensor Networks, Los Angeles, 2005.

[4] J. Liu, D. Nicol, F. Perrone, M. Liljenstam, C. Elliot, and D. Pearson. *Simulation modeling of large-scale ad-hoc sensor networks*. In Proc. European Interoperability Workshop 2001, London, England, June 2001.

[5] S. Park, A. Savvides, and M. B. Srivastava. *SensorSim: A simulation framework for sensor networks.* In Proc. MSWIM 2000, Boston, MA, August 2000.

[6] S. Sundresh, W.-Y. Kim, and G. Agha. *SENS: A sensor, environment and network simulator*. In Proc. 37th Annual Simulation Symposium (ANSS '04), 2004.

[7] J. Polley, D. Blazakis, J. McGee, D. Rusk, J. S. Baras, and M. Karir. *ATEMU: A fine-grained sensor network simulator*. In Proceedings of SECON'04, First IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004.

[8] Vlado Handziski, Joseph Polastre, Jan-Hinrich Hauer, Cory Sharp, Adam Wolisz, David Culler, David Gay. *TinyOS 2.0 Enhancement Proposal (TEP – 2).* http://www.tinyos.net/tinyos-2.x/doc/html/tep2.html

[9] Philip Levis and Nelson Lee. *TOSSIM: A Simulator for TinyOS Networks* http://www.cs.berkeley.edu/~pal/research/../pubs/nido.pdf.

[10] Terence Parr. *ANTLR Parser Generator*. http://www.antlr.org/.

A Quantitative Evaluation of the Simulation Accuracy of Wireless Sensor Networks

Georg Wittenburg
wittenbu@inf.fu-berlin.de

Jochen Schiller
schiller@inf.fu-berlin.de

Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

ABSTRACT

In the field of wireless sensor networks, network simulators are commonly used to evaluate properties of software components or the network as a whole. Their advantages in reduced experimental overhead, flexibility, and repeatability come at the expense of questionable credibility of the results. In order to quantify the simulation accuracy of wireless sensor networks, we have conducted a field test measuring the packet loss rate and compared the data with the results obtained from a carefully configured simulation of the same scenario. Our evaluation gives insight into how much trust can be put into the results of simulations of comparable scenarios.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*wireless communication*; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis; D.2.8 [**Software Engineering**]: Metrics—*performance measures*

Keywords

Wireless Sensor Networks, Simulation, Accuracy, ScatterWeb, ns-2

1. INTRODUCTION

Simulating a Wireless Sensor Network (WSN) provides distinct advantages over a full-scale, real-world deployment when it comes to evaluating new software components: A simulation can be set up in less time, is more flexible with regard to network layout and communication parameters, and allows for different algorithms to be run under exactly the same conditions. These advantages of reduced experimental overhead, flexibility, and repeatability come at the expense of questionable credibility of the results. For most simulations it is unknown how closely the results obtained resemble those from a similar real-world deployment.

The contribution of this paper is to quantitatively evaluate the inaccuracy incurred by relying on a simulation rather than a real deployment. Our approach is to first measure certain metrics in a field test, and then recreate the exact conditions of this test as closely as possible in a simulation with the goal of comparing the measurements taken. While doing so, we pay special care not to use our knowledge of the results of the field test to over-optimize the simulation parameters with regard to reducing the discrepancy between simulative and real-world measurements. Instead, our intention is for this simulation to be just as accurate as any other simulation that is configured carefully following the recommendations from the literature.

For our field test, we used ScatterWeb ESB sensor nodes based on the Texas Instruments MSP430 ultra-low power microcontroller with 60 KB Flash and 2 KB RAM [12]. Inter-node radio communication takes place at 868 MHz on the license-free ISM band using the RF Monolithics TR1001 radio transceiver at a data transfer rate of 19.2 kbps [11]. For the simulations, we relied on our previous work [14] that allows us to run the same software components on both real sensor nodes and the `ns-2` network simulator. This work is briefly summarized in Section 2.1.

The remainder of this paper is structured as follows: Section 2 summarizes key aspects of preliminary work that led up to the current experiment. Section 3 describes in detail the experiment, which consists of both a field test using ScatterWeb sensor nodes and a corresponding simulation. Section 4 presents and evaluates the results. Section 5 gives a brief overview of current research in the area of simulation accuracy, and Section 6 concludes.

2. PRELIMINARIES

In this section, we briefly recapitulate the most important aspects of preliminary work.

2.1 ScatterWeb on ns-2

As pointed out above, simulations offer several advantages over regular deployments when it comes to evaluating new software components for WSNs. Therefore, we have developed an approach to run the same software components both on ScatterWeb sensor nodes and the `ns-2` network simulator [3]. In a nutshell, this was achieved by porting the C API provided by the ScatterWeb firmware to `ns-2`, which was chosen as a simulation platform due to its architectural compatibility with existing ScatterWeb software components, its wide-spread use in research, and in order to avoid the pitfalls of implementing a network simulator from scratch. The key advantage of our simulation approach is that – except for the effects of program execution speed and energy consumption which we intend to address in future work – it leaves the higher-layered software components oblivious to whether they are being executed on a real sensor node or as part of a simulation.

In our previous work [14], we concentrated on API compatibility and the transparent integration of ScatterWeb software components into `ns-2`, while leaving questions regarding the simulation accuracy for future work. In this paper, we add to these results by evaluating the simulation accuracy with regard to packet transmissions over the wireless interface.

2.2 Network Metrics

Table 1: Metrics for Different Network Layers as Applicable in the Field of WSNs

Layer	Metrics
PHY	Bit Error Rate (BER) Radio Signal Strength (RSS)
DLL	Packet Loss Rate (PLR) Packet Collision Rate (PCR)
NET	Packet Delivery Rate (PDR) Hop Count, Latency Overhead Traffic
APP	Application QoS Parameters

Several options are available when deciding which metric to use for comparing simulation and reality. The choice depends on the ISO/OSI layer that we intend to look at. Table 1 lists commonly used metrics sorted by the layers they correspond to.[1] While it would certainly be interesting to compare simulation and reality for measurements of all of these metrics and explore how inaccuracies at lower layers interact with those on the upper layers, this is beyond the scope of this work. Instead, we focus on the Packet Loss Rate (PLR) as seen by the network layer. This choice is motivated by the fact that layer 2 packets are well supported as a networking concept in virtually all simulation tools on one side (bit errors, for instance, are not), and because we want to avoid tying our results to any particular routing protocol on the other side.

2.3 Radio Propagation Models

Given the experimental setup proposed above, the most crucial component of the simulator with regard to the expected results is the radio propagation model. `ns-2` implements three radio propagation models: free space, two-ray ground reflection, and shadowing [3]. The first two are variations of the unit disc graph model, i.e. within a certain radius of the sender all nodes always have perfect reception. These models are known to resemble reality quite poorly [9]. The shadowing model is the only one to include a probabilistic term as part of the calculation of the received signal power:

$$\left[\frac{P_r(d)}{P_r(d_0)}\right]_{dB} = -10\beta log\left(\frac{d}{d_0}\right) + X_{dB}$$

where $P_r(d)$ is the mean received power at distance d as computed relative to a reference power $P_r(d_0)$ at distance d_0. β is the path loss exponent, and X_{dB} is a Gaussian random variable with zero mean and standard deviation σ_{dB}, called the shadowing deviation. For an accurate simulation, both β and σ_{dB} need to be measured at the site of the planned deployment. Typical values for an urban outdoor area range between 2.7 dB and 5 dB for β and between 4 dB and 12 dB for σ_{dB} [10].

We also considered using more sophisticated radio propagation models, e.g. the Radio Irregularity Model (RIM) proposed by Zhou *et al.* [15]. However, the drawback of more recent models is that less data and recommendations exist for a realistic choice of parameter values. Hence, we decided to only consider well-established models for this experiment.

3. EXPERIMENTAL SETUP

This section describes the field test using ScatterWeb sensor nodes as well as the corresponding simulation. In both tests, we transmitted several packets from one sending sensor node to another receiving sensor node and varied both the distance between the nodes as well as the transmission power setting while observing the PLR.

3.1 Field Test

We conducted the field test using two ScatterWeb ESB sensor nodes placed in an urban outdoor environment at a height of 60 cm without any obstructions in their direct line of sight. The distance between the nodes was varied in the range from 5 m to 90 m in steps of 5 m. At each of the distances, we varied the transmission power setting on the firmware API between 0 and 100 (which corresponds to the full range of the TR1001 as connected on the ESB) in steps of 10. For all these combinations of distance and transmission power setting, we sent 20 128-byte packets from one node, counted the correctly received packets on the other node, and calculated the PLR.

3.2 Simulation

For the simulated sensor nodes, we mapped the information from the datasheet [11] and as extracted by inspecting the implementation of the low-level ScatterWeb firmware as closely as possible to the simulation. We tried to reuse existing `ns-2` components wherever possible by adapting their parameters to match the characteristics of the real sensor nodes. As the version of `ns-2` used in our experiments had no support for changing the transmission power for individual packets during the simulation, we modified the simulator to add this feature. Further, there was no information available on how the transmission power setting between 0 and 100 maps to the actual transmission power in milliwatts, so we separately measured these values and configured the simulated sensor nodes accordingly. Due to space constraints we omit the details of these measurements.

Finding good recommendations for the path loss exponent β and the shadowing deviation σ_{dB} in the literature was challenging. In most cases, the focus is on communication distances one order of magnitude higher than the one commonly found in current WSNs. The most suitable values we found are due to Seidel *et al.* [13], who measured β = 2.7 and σ_{dB} = 11.8 dB for a frequency of 900 MHz. Further, ITU-R P.1546 [4] recommends σ_{dB} = 9.5 dB for a frequency of 600 MHz. The only measurements that target WSNs directly are due to Darbari *et al.* [1], whose results are not applicable to our experiment because they were measured at

---

[1] Some of these metrics may also be applicable at other layers than those listed. For example, one could argue that PLR and PDR can also be observed in the NET and DLL layers respectively.

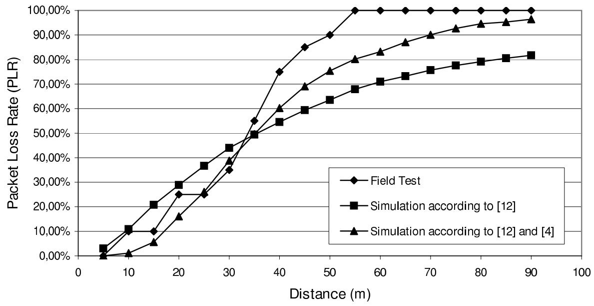


Figure 1: PLR Against Distance with Fixed Transmission Power Setting of 60

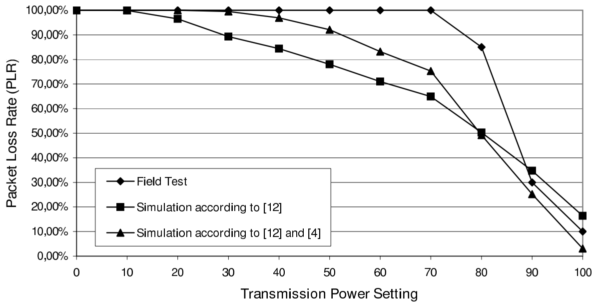


Figure 2: PLR Against Transmission Power Setting with Fixed Distance of 60 m

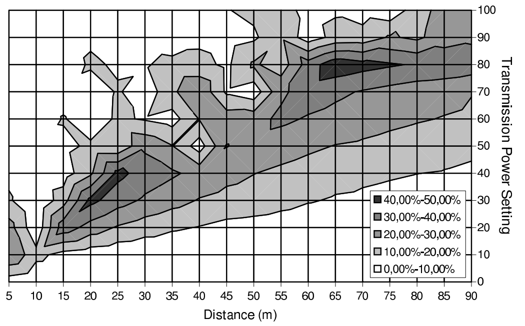


Figure 3: PLR Differences Between Simulation and Reality with Parameters According to [12]

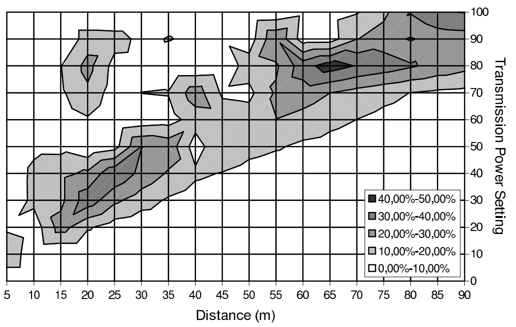


Figure 4: PLR Differences Between Simulation and Reality with Parameters According to [12] and [4]

distances below 1 m and at a frequency of 2.4 GHz. For our experiment, we decided to use the recommended values from [13] and, in a second simulation run, the path loss exponent from [13] combined with the shadowing deviation from [4].

4. RESULTS AND DISCUSSION

Given the experimental setup described in the previous section, we now proceed to present and evaluate the results obtained.

In Figure 1, the PLR is plotted against the distance between two sensor nodes. With the transmission power setting fixed at 60, the diagram representatively illustrates the effects observed and allows us to omit diagrams for the remaining transmission power settings for brevity. The three curves in the diagram correspond to the measurements from the real-world field test and two simulations with different parameters for the radio propagation model. As expected, the PLR increases with larger distances for all three curves and the rate at which it increases differs as a result of inaccuracies in the simulation. Similarly, Figure 2 shows the PLR, but this time plotted against the transmission power setting at a fixed distance of 60 m. Analogous to the observation above, the PLR decreases with higher transmission power settings and simulation inaccuracies can be observed in the different rates of decrease. Once again we omit diagrams for the remaining transmission power settings for brevity.

We now proceed to compare the complete results from the field test with each of the two simulation runs in Figures 3 and 4. These diagrams show the differences between PLRs from the field test and from one simulation respectively. In both diagrams the differences are low for both low transmission power settings at large distances and high transmission power settings at short distances. This is due to the fact that these scenarios are comparatively easy to describe correctly in the radio propagation model and hence the simulation is quite accurate. In contrast, this is not true for values along the diagonal of the diagram as both distance and transmission power setting increase. In these situations, larger differences in PLRs can be observed as the radio propagation model shows its weakness at correctly predicting the characteristics of the signal at the border of the transmission range. Outliers in the diagram, such as the local maximum in the top left quadrant of Figure 4, can be attributed to multi-path signal propagation and additive or subtractive interference at the receiver which are not part of the simulation due to lacking information about the surroundings.

For a quantitative evaluation of the simulation accuracy, one would have to estimate how likely it is for each combination of distance and transmission power setting to occur during a test run, weight the data points accordingly, and then calculate the average difference. For our particular experiment, all combination of distance and transmission power setting occurred equally often, hence no weights are necessary. The average difference between the field test and the simulation according to [13] is 12.3%. The simulation with combined values from [13] and [4] is slightly more accurate with an average difference of 8,2%. It is important to keep in mind that these numbers are closely tied to the metric used for the measurements, which in this case is the PLR, and hence one should not think of them as the one comprehensive quantifier for simulation accuracy. Still, these results give a good indication on how much confidence can be put into the results from a carefully configured simulation.

5. RELATED WORK

The accuracy of network simulators has been studied by Johnson [7] and more recently by Jansen and McGregor [6]. Both works differ from ours in that they use application and transport layer metrics, and for this reason are not directly concerned with the problems arising from transmitting over a wireless medium. Furthermore, they do not compare their results with data from field tests, and generally focus more on the validation of networking algorithms and simulation methodology. [6] is similar to our work in that we share the advantages of integrating existing implementations of software components directly into the simulator.

Liu *et al.* [8] compares data of network layer metrics from a large field test using up to 40 laptop computers communicating over IEEE 802.11 with corresponding simulations. They use different radio propagation models for their comparisons including a "generic model" that is similar to the shadowing model used in our simulations. In their evaluation they observe effects similar to those depicted in Figure 1, however they neither try to quantify the simulation accuracy nor do they elaborate on their choice of parameters.

Ivanov *et al.* [5] discusses the accuracy of `ns-2`-based simulations and emulations with regard to packet delivery ratio, the network connectivity graph, and packet latencies. They partly fine-tuned the parameters of the radio propagation model to match the observed real network topology. Hence, their results correspond to the optimal values for a given scenario, which we intentionally avoid in our approach.

Finally, Newport *et al.* [9] evaluates the impact of commonly made assumptions on simulation accuracy in general.

6. CONCLUSION AND FUTURE WORK

In this paper, we have evaluated the accuracy of a carefully configured simulation with regard to the Packet Loss Rate (PLR) by comparing the results with data from a field test using two ScatterWeb ESB sensor nodes. The average difference between simulation and reality for this metric is 12.3% or 8,2%, depending on which recommendation is followed for the choice of parameters for the radio propagation model. These results allow us to judge the credibility of other simulations for similar deployments.

As a next step we intent to undertake similar experiments for other metrics and analyze how the inaccuracies of the simulation interact over different network layers. Further experiments relying on ray-tracing-based radio propagation models such as proposed in [2] and [16] may complement this work for indoor scenarios. Finally, we are planning to improve our simulations by adding the notions of program execution speed of the simulated software components and correct modelling of energy consumption.

7. REFERENCES

[1] F. Darbari, I. McGregor, G. Whyte, R. W. Stewart, and I. Thayne. Channel Estimation for Short Range Wireless Sensor Network. In *Proceedings of the IEE Conference on DSPenabledRadio*, Southampton, United Kingdom, Sept. 2005.

[2] J.-M. Dricot and P. D. Doncker. High-accuracy Physical Layer Model for Wireless Network Simulations in NS-2. In *Proceedings of the International Workshop on Wireless Ad-Hoc Networks (IWWAN '04)*, pages 249–253, Oulu, Finland, May 2004.

[3] K. Fall and K. Varadhan. *The ns Manual*, May 2007.

[4] International Telecommunication Union. Recommendation ITU-R P.1546-2: Method for Point-to-area Predictions for Terrestrial Services in the Frequency Range 30 MHz to 3.000 MHz, Aug. 2005.

[5] S. Ivanov, A. Herms, and G. Lukas. Experimental Validation of the ns-2 Wireless Model using Simulation, Emulation, and Real Network. In *Proceedings of the 4th Workshop on Mobile Ad-Hoc Networks (WMAN 2007)*, pages 433–444, Bern, Switzerland, Feb. 2007.

[6] S. Jansen and A. McGregor. Performance, Validation and Testing with the Network Simulation Cradle. In *Proceedings of MASCOT 2006*, Monterey, CA, U.S.A., Sept. 2006.

[7] D. B. Johnson. Validation of Wireless and Mobile Network Models and Simulation. In *Proceedings of the DARPA/NIST Workshop on Validation of Large-Scale Network Models and Simulation*, Fairfax, VA, U.S.A., May 1999.

[8] J. Liu, Y. Yuan, D. M. Nicol, R. S. Gray, C. C. Newport, D. Kotz, and L. F. Perrone. Empirical Validation of Wireless Models in Simulations of Ad Hoc Routing Protocols. *Simulation: Transactions of The Society for Modeling and Simulation International*, 81(4):307–323, Apr. 2005.

[9] C. Newport, D. Kotz, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *Simulation: Transactions of The Society for Modeling and Simulation International*, 2007 (accepted for publication).

[10] T. S. Rappaport. *Wireless Communications: Principles and Practice*. Prentice Hall, Dec. 2001.

[11] RF Monolithics Inc. *TR1001 868.35 MHz Hybrid Transceiver Data Sheet*, Aug. 2001.

[12] J. Schiller, A. Liers, and H. Ritter. ScatterWeb: A Wireless Sensornet Platform for Research and Teaching. *Computer Communications*, 28:1545–1551, Apr. 2005.

[13] S. Y. Seidel, T. S. Rappaport, and R. Singh. Path Loss and Multipath Delay Statistics in Four European Cities for 900 MHz Cellular and Microcellular Communications. *IEE Electronics Letters*, 26(20):1713–1715, Sept. 1990.

[14] G. Wittenburg and J. Schiller. Running Real-World Software on Simulated Wireless Sensor Nodes. In *Proceedings of the ACM Workshop on Real-World Wireless Sensor Networks (REALWSN'06)*, pages 7–11, Uppsala, Sweden, June 2006.

[15] G. Zhou, T. He, S. Krishnamurthy, and J. A. Stankovic. Impact of Radio Irregularity on Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Mobile Systems, Applications, and Services (MobySys '04)*, 2004.

[16] F. Österlind. A Ray-Tracing Based Radio Medium in COOJA. Dec. 2006.

Model Checking for Energy Efficient Scheduling in Wireless Sensor Networks[*]

Peter H. Schmitt, Frank Werner[†]
Universität Karlsruhe (TH)
Institut für Theoretische Informatik
{pschmitt,werner}@ira.uka.de

ABSTRACT

Networking and power management of wireless energy - conscious sensor networks is an important area of current research and considering the competitive constraints on the global energy market it will even gain importance. We investigate in the present work a network of MicaZ sensor motes using the Zigbee protocol for communication, and provide a model using Timed Safety Automata. The TA model will comprise a full functional scenario set-up consisting of a network controller, network routers, and sensor devices collecting information. Along with the model sending and receiving mechanisms included, a realistic need-specific investigation is feasible.

Our analysis' focus is on estimating energy consumption by model checking in different scenarios within a fixed but variable topology using the UPPAAL[7] tool. Special interest is devoted to the energy use in marginal situations that rarely occur and are consequently not fully covered by simulation.

1. INTRODUCTION

The technique of model checking has been successfully used in many application areas and has proved particularly useful in very early design stages when only a model or a blueprint of the product is available. Using model checking tools flaws and errors have been revealed early, reducing costly changes in the later product design cycle. The technique has been widely used for the analysis of security protocols, and concurrent, distributed algorithms.

In this paper we investigate the question whether the success story of model checking can be repeated in the area of low-energy sensor networks. We want to gain experience how these networks can be modelled, and what kind of analysis should and can be performed.

Common safety and liveness properties will certainly still play a role, although we rather focus on questions related to energy consumption. What is the minimal energy needed to reach a state for a given property? Can we formulate conditions that will guarantee that the lifetime of a sensor node is at least three month or at least contribute to their solution? Furthermore we want to evaluate to which extent rarely occurring situations can be identified.

The plan of this paper is as follows. In the rest of this introduction we briefly introduce the theoretical framework of model checking. Section 2 describes our model for sensor networks, in Section 3 we present our results, and conclude with the usual wrap-up and suggestions for future research in Section 4.

What Model Checking is. Model checking is a formal method for automatically verifying system designs which has been applied to an impressive variety of areas. The method requires a model $\mathcal{M}$ of the system under investigation, a property ϕ that the system should have, and an algorithm to check whether $\mathcal{M}$ indeed satisfies ϕ. The method is very flexible since it is applicable to all systems that can be modelled as some kind of finite state machine. Investigations into application areas related to wireless sensor networks are just starting [6, 5, 9].

Due to several considerations we decided to choose for modelling the sensor network UPPAAL[7], an integrated tool environment for the design, simulation and verification of real-time systems. The tool is fairly efficient, and adequate for systems that can be modelled as a collection of non-deterministic processes, communicating through binary actions, broadcasting channels, or shared variables, and having a finite control structure, and real-valued clocks. Especially the natural use of clocks in UPPAAL is promising an adequate and realistic modelling of the sensor network.

2. MODEL OF SENSOR NETWORK

For being comparable to other studies we chose the MicaZ sensor mote manufactured by Crossbow since it provides a versatile platform in particular for low-energy sensor networks [8]. The protocol for communication between the sensors is the Zigbee protocol[10] because it can be beneficially used, combining low transmission rates while fulfilling the criteria of being energy-conscious.

Out of the different topologies that exist we choose the *Mesh-Network* to be the most appropriate one in our scenario. What we pursue is a Peer-to-Peer network consisting of only FFDs (full functional devices) since we aim on using a beacon disabled network. Device-to-device, and device-to-router communication is established using the CSMA/CA access on the common medium. As such devices wake up in certain intervals send their recently gathered information, and fall back to sleep. The communication medium is represented as one channel on which all devices communicate, which is fixed over the analysis. This is intentionally done since it introduces collisions and related situations of inter-

---

[*] This work has previously been published as a report available at `http://www.ubka.uni-karlsruhe.de/cgi-bin/psview?document=ira/2007/1`.

[†] This research position is funded by the BW-FIT Project ZeuS (Zuverlässige Informationsbereitstellung in energiebewussten ubiquitären Systemen).

est. The more, it allows us to neglect the overhead arising from network maintenance like active-, passive scans, and channel changes because of high traffic with close-by networks. Apart from having bidirectional communication we restrict the model on passing information from the sensor devices to the network coordinator, and explicitly forbid a vice versa communication flow.
Whenever a sensor end device (ZED) is waking up, it is forwarding the recently collected data packets to the router (ZR), and the going back to sleep again. The routers in turn pass the packet along the network link - possibly using other routers - to the Zigbee network controller (ZC) which is the root of each Zigbee network and unique. Routers are the only devices in our model that embody a sending and receiving side.

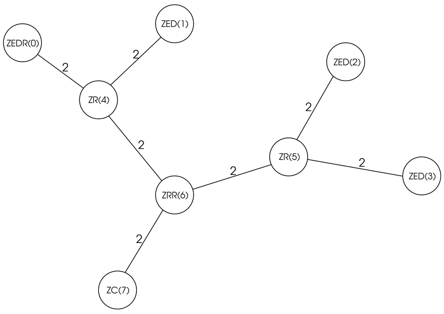
Uppaal **Model.** The sensor nodes are the only devices which have the capability of collecting sensor values and in turn have no means to receive packets from the rest of the network. Routers can receive and forward packets, and finally the network controller has a mere capability of receiving packets. For the measure of energy consumed, a reference sensor, and a reference router are modelled. Due to this design issues the state space is kept small, retaining the essential functions for each device.
The power draw of the reference devices [1] is incorporated into the model, using values as shown in the table below (Tab. 1), and costs are accumulated whenever state changes occur.
Although the model makes no use of changing transmission rates during execution due to complexity considerations, the MicaZ transmission can be changed in advance accordingly to the distance matrix from Fig. 1(b) in concrete steps of $-10dBm$, $-5dBm$, and $0dBm$. To account for a very restricted state space, the models transmission rate is constant in the model but changed over different properties to imitate different scenarios.

Sensor Nodes and Routers. Modelling the sensor nodes, routers, and controllers shown in Fig. 2 we singly use the MicaZ motes, although heterogeneous networks with more powerful routers can be desirable for other applications. In our approach Zigbee Controller (ZC), Router (ZR), and End Devices (ZED) are modelled in a similar fashion with only subliminal differences, corresponding to the requirements of a homogeneous networking scenario. The costs i.e. the energy use of the individual sensor devices, is incorporated into the model to allow even distinct transmission rates for a realistic scenario. By using this approach the required energy to reach a specific state can be immediately derived from the verification.

Network Controller. is intentionally modelled without an energy function. This is mainly because we assume it to be attached to some persistent power supply since it is most critical to the network.

Table 1: MicaZ's energy measures in $[\mu A]$ for processor (proc) and transmission unit (trans).

State	proc	trans	Remarks
PDown	15	1	down mode
PSleep	8 000	1	proc up, trans down
PIdle	8 000	20	proc and trans up
PSnd1	8 000	11 000	sending at $-10dBm$
PSnd2	8 000	14 000	sending at $-5dBm$
PSnd3	8 000	17 400	sending at $0dBm$
PRcv	8 000	19 700	receiving mode



(a) Scenario with reference sensor (ZEDR), sensors (ZED), routers (ZR), reference router (ZRR) with cost function, and Zigbee Controller (ZC). Edges are labelled with distances and numbers in parenthesis denoted process.

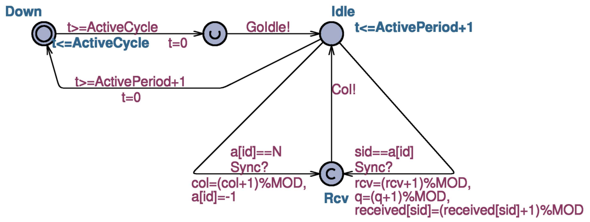
$$\begin{pmatrix} 0&4&8&8&2&6&4&6\\ 4&0&8&8&2&6&4&6\\ 8&8&0&4&6&2&4&6\\ 8&8&4&0&6&2&4&6\\ 2&2&6&6&0&4&2&4\\ 6&6&2&2&4&0&2&4\\ 4&4&4&4&2&2&0&2\\ 6&6&6&6&4&4&2&0 \end{pmatrix}$$

(b) Distance matrix for the selected scenario.

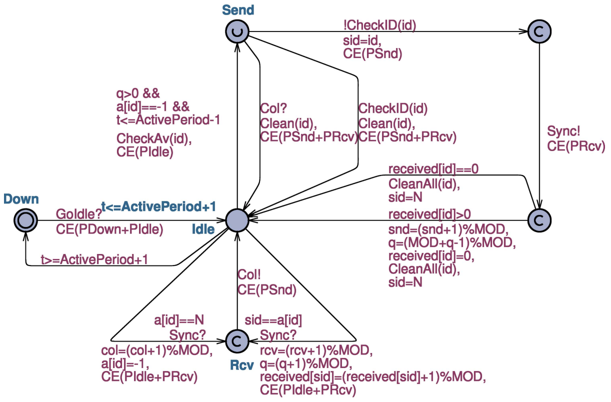
Figure 1: Scenario settings underlying the analysed sensor network.

Zigbee Protocol. For being comparable to results gained through practical experiments the model is designed as proposed in the Zigbee specification [10] for homogeneous networks underlying a tree topology with devices as shown in Fig. 1(a). Digits in parenthesis indicate the process number used for identification later on in the verification part. Distances between respective entities are modelled using the distance matrix from Fig. 1(b) to determine communication flow within the sensor network as well as the number of hops a packet as to undergo until arriving at its destination.
Since most energy is preserved in sleep mode - where processor, and on-board transmission unit are shut down - we target an average duty cycle of 1%[8] by defining *ActivePeriod* 1 and *BeaconInterval* 100 in our model in Fig. 2. The Zigbee protocol is designed to handle most of the communication within the contention free period collision free, although we consider the contention period in which the CSMA/CA feature is used to be an interesting point to investigate.

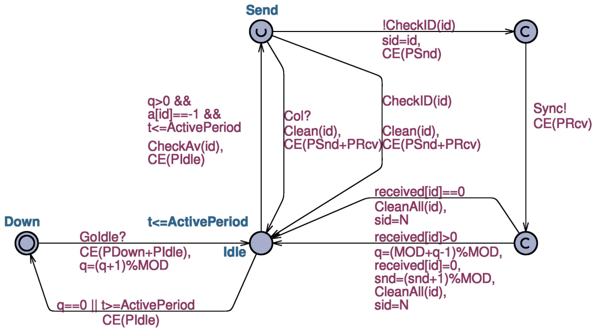
Cost Estimation. Since special interest is on the estimation of costs, a reference sensor node cf. Fig. 2(c) and a reference router cf. Fig. 2(b) are equipped with a *cost estimation* function $CE()$ where energy costs are accumulated along the paths, avoiding a gratuitous blow-up of the state space. Consequently, costs from Tab. 1 are used according to state changes by the automata, giving some observables at hand. Respectively adopted to this design is the cost of leaving state Down. Using the cost *PDown* multiplied



(a) The Zigbee Network Controller in charge of the network beacon maintenance.



(b) The Zigbee reference Router with cost function $CE()$ forwarding packets along the network links.



(c) The reference Zigbee sensor node with cost function $CE()$ delivering sensor values to the network.

Figure 2: UPPAAL timed automata models of the Zigbee sensor network.

by 99 time units gives an accurate energy consumption for the time spent in this state, totalling the transition cost to $PDown' = 1\,548\mu A$.

3. VERIFICATION RESULTS

Before the outcomes of different properties are tested using the new model, the scenario is tested for deadlock freeness ($A\Box\neg deadlock$) to assure plausible sound modelling and sanity. For all experiments the state space is bound by introducing variable MOD which is set to appropriate values, and computing all variables modulo this upper bound. The transmission rate is increased over different scenarios from $-10dBm$, $-5dBm$, to $0dBm$, augmenting the theoretical coverage. These energy draws correspond to ranges of two, four, and six units of the distance matrix shown in figure 1(b) and have a merely theoretic character. Properties are verified by using a hash table size of 512MB for state hashing in the UPPAAL tool setting, and giving the shortest path for some satisfying property, needed for finding the lowest cost. Using a bigger hashtable does only fractionally increase the modelling size since the state space grows exponentially with the number of variables used (assuming no optimisation at all is done).

Table 2: Energy consumed by the sensor device for different ranges of two, four, and six units of coverage.

range	Property	energy $[\mu A]$
2	$E\Diamond\, ZC.rcv = 1 \& ZEDR.snd = 1$	64 300
4	$E\Diamond\, ZC.rcv = 1 \& ZEDR.snd = 1$	67 300
6	$E\Diamond\, ZC.rcv = 1 \& ZEDR.snd = 1$	70 700

3.1 Energy Considerations

Starting with a deadlock free model, energy considerations are obtained by searching the state space spanned by the model for properties as specified by the user. Whenever a satisfying state is found, a path is generated that shows the transitions taken until the state is reached. All experiments conducted here investigate the use of energy of the reference sensor and router under different scenarios. The desired properties are checked by definition of CTL[4, 2] formulae.

Sensor Devices. The first experiment conducted is observing the power drawn by the reference MicaZ sensor under the following property:

$$E\Diamond\, ZC(7).rcv = 1 \& ZEDR(0).snd = 1$$

This CTL property expresses in a verbalised form, questions: *"What is the energy used by the reference node if it transmits one packet ($ZEDR(0).snd = 1$) which is routed through the network and finally received by the Zigbee Controller ($ZC(7).rcv = 1$)?"*
By use of the temporal operator $E\Diamond\,\phi$ ("Does there exist a path such that ϕ does eventually hold in the future"), the shortest path is returned as defined by the appropriate strategy. Tab. 2 captures the energy drawn for the above property while varying the theoretic transmission range.

Zigbee Routers. After having studied the energy drawn by Zigbee sensor gadgets, a further step is to investigate the cost that occur at the routing devices, since they need more power due to higher activity. For this scenario the reference Zigbee router ZRR from Fig. 2(b) has been chosen, since it interlinks the Zigbee controller with the network, and is hence most critical to energy constraints.
The analysis observes the energy consumption by the router using different transmission rates, and increasing collisions occurring at the router over the experiments. Results are illustrated in the table 3 below. As expected, by increasing the transmission rates more devices in the network are capable of over-leaping the reference router node, thus preserving the ZRR's energy, and enable a faster delivery of packets to the ZC at the root.

Table 3: Energy in $[mA]$ consumed by the reference Zigbee router with varying transmission range to account for an packet successfully send and received with varying collisions.

range	Property	Energy
2	$E\Diamond ZC(0).rcv = 1$	167
4	$E\Diamond ZC(0).rcv = 1$	138
6	$E\Diamond ZC(0).rcv = 1$	91
2	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 1$	214
4	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 1$	128
6	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 1$	144
2	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 2$	261
4	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 2$	138
6	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 2$	197
2	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 3$	355
4	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 3$	138
6	$E\Diamond ZC(0).rcv = 1\&ZRR(6).col = 3$	250

4. CONCLUSION

Our experiments showed that the timed automata model presented in Section 2 is a good basis for the analysis of energy consumption of sensor motes within an arbitrary scenario. Special emphasis is hereby on the investigation of marginal or borderline situations which rarely occur in simulation and can thus be exhaustively analysed using the here presented approach.

Furthermore we were able to determine cost optimal timings for specific schedules. The features provided by the UPPAAL tool proved to be flexible enough to formulate queries involving the estimation of energy consumption. Especially the fact that UPPAAL offers a built-in concept of multi-cast and a notion of timing were very useful during the modelling phase. It seems notable, that although the model consists of eight devices - and modern simulation tools can do up to thousands - the analysis presented here is more exhaustive. As such situations which rarely occur in simulation can be identified by the current approach using timed automata theory and hereby help to investigate these border-line scenarios.

As can be seen from Table 3 higher transmission rates impose a higher energy consumption on each device, but simultaneously enable packets to reach the ZC using less hops along the network links. On the other side packets transmitted from other motes might collide more often in this case, and consequently the number of retransmissions before a packet's successful delivery to its destination is increased.

Although the verification is restricted due to limitations of UPPAAL, we believe that the model can be adopted to suit an even deeper analysis than shown here. To tackle the size of the state space, several attempts techniques are introduced that allow an increasing model complexity.

By accompanying the verification techniques pursued here with more realistic data, a deeper understanding of routing, contentions, and the hereby related energy consumption can be obtained. So far we have not made a serious attempt to use the counter measures recommended in the UPPAAL tutorial[3] to curb state explosion. It will also be promising to explore the potentials of other tools, and to look into infinite model checkers.

5. REFERENCES

[1] Micaz data sheet - wireless sensor networks. `www.xbow.com`.

[2] Rajeev Alur, Costas Courcoubetis, and David L. Dill. Model-checking for real-time systems. 1990.

[3] Gerd Behrmann, Alexandre David, and Kim G. Larsen. A tutorial on uppaal. Technical report, Department of Computer Science, Aalborg University, Denmark, November 2004.

[4] E. Clarke and I.A. Draghicescu. Expressibility results for linear-time and branching-time logics. In J.W. deBakker, W.P. deRoever, and G. Rozenberg, editors, *Proc. Workshop on Linear Time, Branching Time, and Order in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 257–268. Springer, 1988.

[5] Sinem Coleri, Mustafa Ergen, and T. John Koo. Lifetime analysis of a sensor network with hybrid automata modelling. In *WSNA '02: Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications*, pages 98–104, New York, NY, USA, 2002. ACM Press.

[6] YoungMin Kwon and Gul Agha. Performance evaluation of sensor networks: A statistical modeling and probabilistic model checking approach. In *ACM Transactions on Embedded Computing Systems (ACM TECS)*, 2006.

[7] Kim G. Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1997.

[8] Ciaran Lynch and Fergus O'Reilly. Processor choice for wireless sensor networks. In *Proc. 1st Workshop on Real-World Wireless Sensor Networks REALWSN*, number T2005:09 in SICS Technical Reports, pages 58–62. SICS, Stockholm, Sweden, 2005.

[9] G. Norman, D. Parker, M. Kwiatkowska, S. Shukla, and R. Gupta. Using probabilistic model checking for dynamic power management. *Formal Aspects of Computing*, 17(2):160–176, August 2005.

[10] ZigBee specification. Zig-Bee Document 053474r06, Version 1.0, June 2005.

Simulation von plattformunabhängigen TinyOS-Applikationen mit ns-2

Juri Saragazki, Olaf Landsiedel, Klaus Wehrle
Computer Science Department, Distributed Systems Group
RWTH Aachen
{juri.saragazki, olaf.landsiedel, klaus.wehrle}@
rwth-aachen.de

Kurzfassung

Die Entwicklung von verteilten Algorithmen, Betriebssystemen und Applikationen stellt aufgrund der außerordentlich beschränkten Ressourcen der einzelnen Sensorknoten und der hohen Dynamik, der drahtlose Sensornetzwerke unterliegen, eine besondere Herausforderung in der Forschung dar. Die Simulation von verteilten Applikationen und Betriebssystemen mit schlecht überschaubaren, komplizierten Vorgängen ist ein wichtiger Schritt in Bezug auf die Evaluierung von Algorithmen und ihren Implementierungen. In dieser Arbeit wird ein Rahmenwerk zur Simulation von TinyOS-Anwendungen vorgestellt. Es ermöglicht Anwendungen, die ohne Modifikationen auf Sensorknoten lauffähig sind, sowie deren Interaktion miteinander zu untersuchen und somit genaue und detaillierte Vorhersagen mit Hilfe des weit verbreiteten Netzwerksimulators ns-2 über ihr Verhalten in einer natürlichen Umgebung zu machen.

Schlüsselbegriffe

Drahtlose Sensornetzwerke, Simulation von Applikationen, TinyOS 2.x, Network Simulator 2, ns-2

1. MOTIVATION

Beim Aufbau von drahtlosen Sensornetzwerken mit autonomen Sensorknoten existiert eine Reihe von verschiedenartigen Herausforderungen. Für Elektroingenieure bringt die kompakte Größe der einzelnen Knoten einige Schwierigkeiten mit sich, wohingegen Informatiker bei der Entwicklung von Applikationen und Betriebssystemen mit der Limitierung der Ressourcen zurechtkommen müssen.

Aufgrund des hohen Komplexitätsgrads von verteilten Anwendungen, der Beschränkungen hinsichtlich der Hardwareressourcen und der hohen Anschaffungskosten der Komponenten ist ein Verzicht auf eine ausführliche Evaluierungsphase in einer realen Umgebung oder in einer Simulationsumgebung vor der Inbetriebnahme nicht denkbar. Der Aufbau einer realistischen Testumgebung mit einer hohen Anzahl von Knoten erweist sich jedoch wegen mehrerer Gründe oft als unmöglich bzw. nicht zweckmäßig. Es wird eine große Fläche benötigt und viele der natürlichen Phänomene können in einer solchen Umgebung nicht reproduziert werden. Die Kosten für einen zeitintensiven und materialaufwendigen Aufbau würden dabei realistische Grenzen übersteigen.

Um Applikationen und Netzwerke dennoch zu evaluieren, werden Netzwerksimulatoren eingesetzt. Je nach Wahl des Simulators muss die Anwendung, die später auf dem Sensorknoten ausgeführt wird, an den Simulatorkern angepasst werden, was wiederum einen hohen Zeitaufwand und eine Verschlechterung der Übertragbarkeit der Ergebnisse bedeutet.

Die rasante Entwicklung im Bereich der Miniaturisierung hat zu einer hohen Anzahl von verfügbaren Sensorknotenplattformen geführt. So unterscheiden sich die Geräte in Prozessorleistung, Speichergröße, Funkübertragungsbandbreite und -reichweite, Energieverbrauch, usw. Die hohe Vielfalt an Hardware bringt einerseits viele Vorteile mit sich, führt jedoch bei der Entwicklung von zuverlässigen Anwendungen zu einer Reihe von Schwierigkeiten.

TinyOS 2.x [3, 7] ist ein weit verbreitetes und frei verfügbares Betriebssystem für Sensornetzwerke der zweiten Generation, das eine plattformunabhängige Softwareentwicklung ermöglicht und eine Vielzahl von unterschiedlichen Hardwareplattformen unterstützt. TOSSIM [5, 6] ist der einzige den Autoren bekannte Simulator, der eine Simulation von unmodifizierten TinyOS 2.0-Anwendungen ermöglicht. Jedoch hat dieser Simulator einige Nachteile. So ist es nicht möglich unterschiedliche Applikationen in einem Netzwerk zu simulieren und die Anzahl der verfügbaren Modelle bzgl. Funkwellenausbreitung, sowie Umgebungssimulation ist begrenzt. NesCT [8] ist ein Sprachenübersetzer, der aus *NesC*-Quelltext *C++*-Klassen für den Simulator OMNeT++ [1] erstellt.

„Network Simulator 2" [2, 9] ist ein weit verbreiteter Netzwerksimulator mit einer breiten Palette von unterschiedlichen Modellen. In dieser Arbeit wird ein Rahmenwerk zur Simulation von unmodifizierten TinyOS-Anwendungen mit ns-2 vorgestellt. Mit Hilfe dieses Werkzeugs kann der Entwicklungsprozess von TinyOS-Applikationen erleichtert und beschleunigt werden. Eine zeitnahe Evaluierungsphase kann zur Verbesserung der Qualität der Software führen. Der Entwickler ist in der Lage Anwendungen in einer simulierten Umgebung zu testen und zu bewerten, eventuell vorhandene Fehler zu beseitigen und die Leistungsfähigkeit bzw. die Netzwerktopologie zu optimieren.

2. SYSTEMDESIGN

In dieser Arbeit wird ein System zur Evaluierung von unmodifizierten Anwendungen für Sensorknotennetzwerke vorgestellt. Der Fokus liegt dabei auf der Portierbarkeit der Anwendungen. Bei der Realisierung des Rahmenwerks stand der Faktor Benutzerfreundlichkeit im Vordergrund. Der Pro-

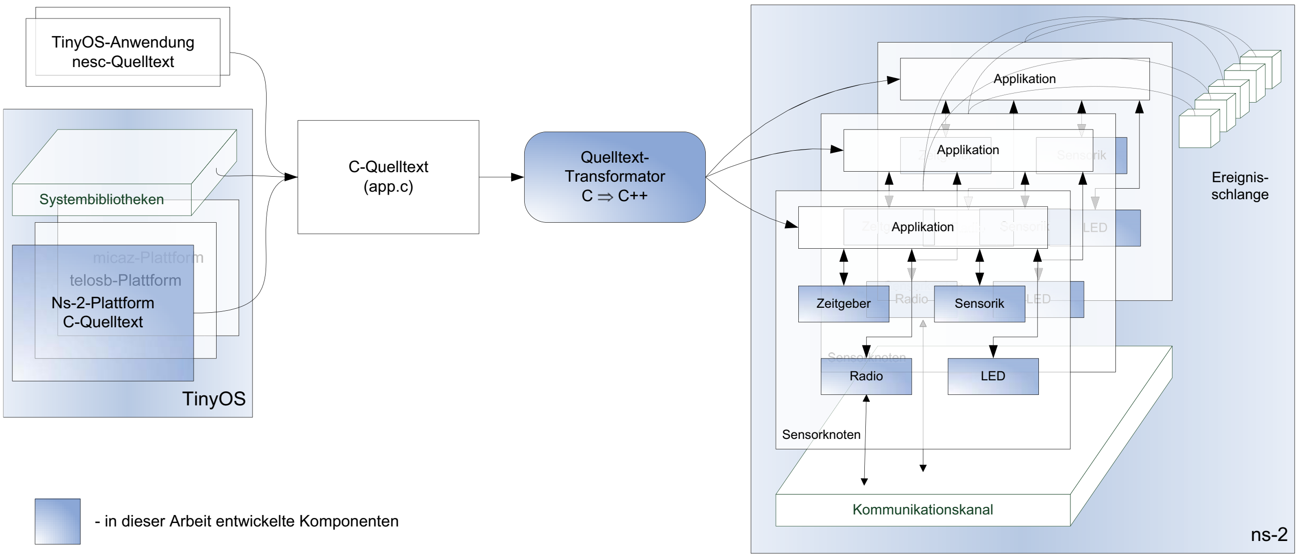


Abbildung 1: Adaption von TinyOS-Anwendungen an ns-2

zess bei der Erstellung einer Simulation von TinyOS-Anwendungen sollte daher automatisch bzw. mit möglichst wenigen manuellen Modifikationen ablaufen.

Um eine schnelle Portierung der Applikation zu erreichen, ist die Modifikation des Betriebssystems TinyOS 2.x ein angemessener Weg. Eine neue Plattform dient als Grundlage für eine Applikationsabstraktion bei der Simulation. Ein Vorteil einer abstrakten Plattform gegenüber einer hardwarenahen Plattform ist eine höhere Leistungsfähigkeit, da weniger Quelltext entsteht und so auch weniger Speicher benötigt wird. Diese neue Plattform bildet die Grundlage für die Verknüpfung zwischen der Applikation und dem Simulatorkern von ns-2. Sie beinhaltet die wichtigsten abstrakten Hardwarekomponenten auf der HIL [4] in der Abstraktionsarchitektur von TinyOS 2.x, wie z.B. Zeitgeber, Radio, Sensorik und Speicher. Der *nesC*-Compiler übersetzt den Quelltext der Applikation sowie der benötigten Komponenten in einen ausführbaren Quelltext der Sprache *C*. So entsteht eine große Systemdatei, die alle benötigten Deklarationen und Komponenten beinhaltet, wobei die originäre Strukturierung des ursprünglichen Quelltextes nicht erhalten bleibt. Diese Datei besteht aus vielen tausend Zeilen Quelltext und enthält eine Ansammlung von Prozedurdefinitionen, die aufgrund der modularen Betriebssystemarchitektur von TinyOS umfangreich ineinander verschachtelt sind.

Weiterhin wurde im Rahmen dieser Arbeit ein Werkzeug entwickelt, das aus der Systemdatei eine Klasse, die bei der Simulation zur Abstraktion der Applikation benutzt wird, generiert. Durch die Anwendung dieses Tools entsteht aus einer prozedurbasierten Sensorknotenanwendung, implementiert in der Programmiersprache *C*, eine abstrakte *C++*-Applikationsklasse. Bei der Transformation erfolgt eine Aufsplittung der großen Systemdatei in eine syntaktisch *C++*-typische Strukturierung mit einer Definitionsdatei mit Typ-, Variablen- und Funktionsdeklarationen und einer Hauptdatei, die die Methodenimplementierungen enthält. Das Werkzeug wurde in der Programmiersprache *Java* realisiert, weil diese umfangreiche Möglichkeiten zur Bearbeitung von regulären Ausdrücken bietet. Der Erstellungsprozess der Plattform wurde dabei so geändert, dass die Quelltexttransformation automatisch eingeleitet wird.

Der Simulator ns-2 wurde durch Komponenten erweitert, die die Hardware eines Sensorknotens abstrahieren. Virtuelle Sensorknoten führen Anwendungen aus, kommunizieren miteinander, lesen mit Hilfe von Sensoreinheiten Daten aus der Umgebung aus und verarbeiten diese. Jeder Knoten enthält eine Instanz der Applikationsklasse. Während der Simulation der Applikation greift die Applikationsinstanz auf die Funktionalitäten der ns-2-Klassen zu. Dieser Aufbau ermöglicht die Simulation von unterschiedlichen Applikationen auf verschiedenen Sensorknoten.

Da ns-2 ereignisgesteuert abläuft, muss jede Komponente mit Hilfe eines Zeitgebers ein Ereignis in die Warteschlange der Ablaufkontrolle des Simulators einfügen. Die Instanz der Applikation ruft die Methode der benötigten Hardware-Instanz, z.B. des Radiosenders auf. Nach Ablauf des Zeitgebers wird ein Ereignis ausgelöst, das für eine Rückführung zur Applikation sorgt. Die benötigte Instanz ruft die Methode der Applikation auf, die der Anwendung mitteilt, dass die Funktionalität im Simulator durchgeführt wurde. Nachfolgend wird die Hauptschleife der Ablaufkontrolle der Applikation aufgerufen, um evtl. anstehende Aufträge zu bearbeiten. Abbildung 2 skizziert das Zusammenspiel der Komponenten eines Sensorknotens in ns-2.

Die Aufrufkette beinhaltet dabei folgende Schritte:

1. Aufruf zur Ausführung einer Funktionalität aus der Applikation an die Simulatorkomponente.

2. Einordnung des Ereignisses in die Ereigniswarteschlange des Simulators.

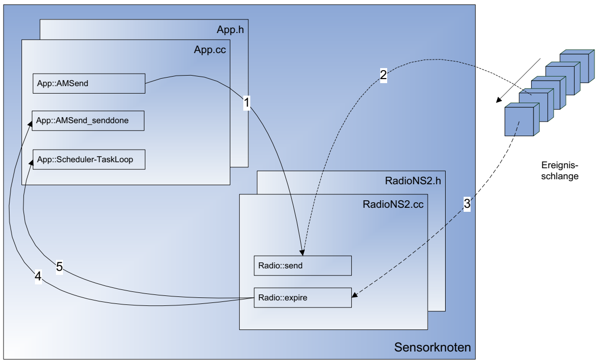


Abbildung 2: Aufrufkette bei der Simulation der Applikationen

3. Abarbeitung des Ereignisses.
4. Rückführung zur Applikation.
5. Anstoßen der Hauptschleife der Ablaufsteuerung der Applikation.

Detaillierte Beschreibung der einzelnen Schritte und Komponenten kann in [10] eingesehen werden.

3. EVALUIERUNG

Es wurden zahlreiche Funktionalitätstests und Simulationsläufe zur Messung der Leistungscharakteristiken durchgeführt. Bei den Tests wurde die Funktionalität der neu erstellten TinyOS-Plattform sowie der Funktionsumfang des Quelltexttransformators bei der Adaption an den Simulator ns-2 geprüft. Dabei wurden Standardanwendungen, die im Umfang des Betriebssystems enthalten sind, getestet. Bei nahezu 70% aller getesteten Applikationen war die Ausführung des originären TinyOS-Quelltextes im Simulator möglich. In den restlichen Fällen, z.B. bei der Verwendung einer Funktionalität, die nicht in der ns-2-Plattform implementiert wurde, war eine Kompilierung der Applikation nicht möglich. Dabei wurde die Anpassung an den Simulatorkern in den meisten Fällen automatisch durchgeführt, jedoch waren in einigen Fällen kleine manuelle Änderungen notwendig. Betriebssystemkomponenten von TinyOS greifen bei der Simulation auf die Funktionalitäten der virtuellen Knoten zu. Daher ist ihre Implementierung in TinyOS nur dann zweckmäßig, wenn die entsprechende Funktionalität im Simulator vorhanden ist. So wird z.B. im Rahmen dieser Arbeit der Energieverbrauch der Sensorknoten nicht betrachtet. Aus diesem Grund wurden die Konfigurationen, die unter die HIL [4] greifen, nicht realisiert.

Um die Leistungsfähigkeit des Simulators zu testen wurde eine Vielzahl von unterschiedlichen Messungen durchgeführt. Dabei wurden der Speicherverbrauch und die zur Simulation benötigte Zeit als Leistungsmerkmale protokolliert, da diese beiden Faktoren oft eine Beschränkung bei komplexen Berechnungsaufgaben darstellen.

Bei der Bewertung der Leistungsfähigkeit des Simulatorkerns existiert eine Vielzahl von Parametern, die einen Einfluss auf die Resultate haben. So spielen die Anzahl der Sensorknoten, die Knotendichte, die Simulationsanwendung, insbesondere die Anzahl der gesendeten und der empfangenen Nachrichten, die virtuelle Simulationsdauer, die Wahl des Funkausbreitungsmodells und die Netzwerktopologie eine bedeutende Rolle und beeinflussen die Ergebnisse bei der Messung der Ressourcen. Bei einer solch hohen Anzahl von Einfussfaktoren sind alles einschließende Aussagen schwierig. Es lassen sich jedoch Tendenzen erkennen. Bei den Simulationsmessungen hat sich herausgestellt, dass die Berechnung des Funkausbreitungsmodells die benötigte Zeit bei einer Simulation beschränkt. Der exponentielle Anstieg der benötigten Zeit (siehe Abbildung 4) im Verhältnis zur Erhöhung der Anzahl der Sensorknoten sorgt für einen rasanten Anstieg der Rechenzeit, da die Simulation der Radioübertragung aufwendig und rechenintensiv ist. Der Speicherverbrauch (siehe Abbildung 3) entwickelte sich bei den Tests linear.

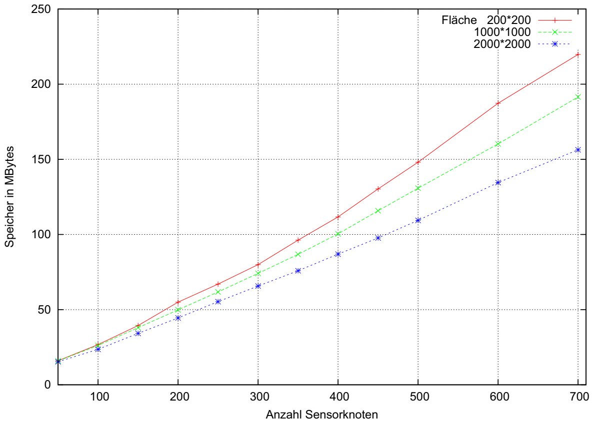


Abbildung 3: Speicherverbrauch bei der Variation der Anzahl der Sensorknoten

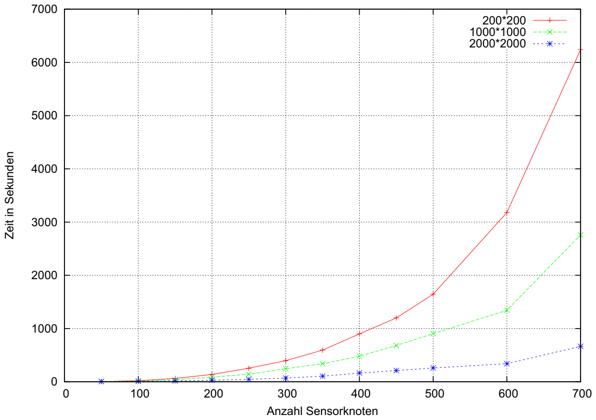


Abbildung 4: Benötigte Zeit bei der Variation der Anzahl der Sensorknoten

4. FAZIT UND AUSBLICK

Zusammenfassend stellt das Rahmenwerk gemeinsam mit dem Simulator ns-2 ein mächtiges und effizientes Werkzeug bei der Entwicklung von TinyOS-Anwendungen dar und ermöglicht eine schnelle Analyse der umgesetzten Ideen. Diese

Arbeit stellt eine Grundlage für die Simulation von unmodifizierten TinyOS-Applikationen, wobei weitere Optimierungen zur Verbesserung der Simulationsergebnisse beitragen könnten.

Ein wichtiger Aspekt bei der Simulation von Sensornetzwerken ist der Energieverbrauch. Sensorknoten sollen über eine längere Zeit in einer natürlichen Umgebung wartungsfrei im Einsatz sein können. Die Möglichkeit der Simulation des Energieverbrauchs von TinyOS-Applikationen würde zu weiteren Optimierungsoptionen bei der Erstellung der Software führen. Eine Erweiterung der Abstraktion des Sensorknotens in ns-2, die den Energieverbrauch simuliert, würde auch zu detaillierten Simulationsergebnissen über das gesamte Netzwerk beitragen. Knoten, die aufgrund eines erhöhten Energieverbrauchs ausfallen würden, könnten schneller identifiziert werden.

Eine genaue zeitliche Abbildung zwischen der Applikation und der Ablaufkontrolle des Simulators könnte eine weitere Optimierung der Ergebnisse hervorbringen. Die Simulation läuft ereignisbasiert ab. Das hat zur Folge, dass eine Rechenoperation ein Ereignis im Simulator auslöst. Die Warteschlange, in der sich Ereignisse befinden, wird von der Ablaufkontrolle des Simulators abgearbeitet. Jedoch ist die zur Abarbeitung benötigte Zeit auf einem PC aufgrund der unterschiedlichen Hardwarearchitektur nicht auf Sensorknoten übertragbar. Für eine genaue Abbildung der realen und virtuellen Zeit ist eine exakte hardwarebezogene, plattformspezifische Modellierung notwendig.

5. QUELLENANGABEN

[1] OMNeT++ Community Site. URL: `http://www.omnetpp.org/` [zuletzt besucht am 25.06.2007].

[2] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, Satish Kumar, Steven McCanne, Reza Rejaie, Puneet Sharma, Kannan Varadhan, Ya Xu, Haobo Yu und Daniel Zappala. Improving Simulation for Network Research. Technical Report 99-702b, University of Southern California, März 1999.

[3] TinyOS Research Group. TinyOS 2.0 Dokumentation. URL: `http://www.tinyos.net/tinyos-2.x/doc/` [zuletzt besucht am 08.05.2005].

[4] Vlado Handziski, Joseph Polastre, Jan-Hinrich Hauer, Cory Sharp, Adam Wolisz, David Culler und David Gay. TinyOS-Dokumentation TEP 2 - Hardware Abstraction Architecture. URL: `http://www.tinyos.net/tinyos-2.x/doc/html/tep2.html` [zuletzt besucht am 27.05.2007].

[5] Philip Levis. TOSSIM: A Simulator for TinyOS Networks. URL: `citeseer.ist.psu.edu/642439.html`.

[6] Philip Levis. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications, 2003. URL: `citeseer.ist.psu.edu/levis03tossim.html`.

[7] Philip Levis. T2: A Second Generation OS For Embedded Sensor Networks. Technical report, University of California, Berkeley, 2005.

[8] Offizielle Internetpräsenz. NesCT: A Language Translator. URL: `http://nesct.sourceforge.net/index.html` [zuletzt besucht am 21.06.2007].

[9] Offizielle Internetpräsenz. The Network Simulator - ns-2. URL: `http://www.isi.edu/nsnam/ns/` [zuletzt besucht am 20.05.2007].

[10] Juri Saragazki. Entwurf und Realisierung eines plattformunabhängigen Kommunikationsubsystems für Sensornetzwerke. Diplomarbeit, RWTH Aachen, Fachbereich Informatik, LuFGI 4 Verteilte Systeme, 2007.

A Simulation Model of IEEE 802.15.4 in OMNeT++

Feng Chen
Falko Dressler

Computer Networks and Communication Systems
University of Erlangen-Nuremberg
91058 Erlangen, Germany
{feng.chen,dressler}@informatik.uni-erlangen.de

ABSTRACT

IEEE 802.15.4 defines the physical and MAC layer specifications for low-rate wireless personal area networks (WPANs). In order to evaluate its performance, we develop a simulation model of IEEE 802.15.4 in OMNeT++, which is a popular simulation platform especially suitable for the simulation of communication networks. The model consists of two modules for PHY and MAC layers respectively and supports star and cluster tree topologies. Our model is built conforming to the latest version - IEEE Std 802.15.4-2006 and has the extendibility to the ZigBee protocol stack.

Keywords

IEEE 802.15.4, simulation model, OMNeT++

1. INTRODUCTION

IEEE 802.15.4 [1] is a standard designed for low-rate wireless personal area networks (LR-WPAN) and defines the specifications at the physical layer (PHY) and medium access control (MAC) sublayer. In contrast to wireless local area network (WLAN), which is standardized by IEEE 802.11 family, LR-WPAN stresses short-range operation, low-data-rate, energy-efficiency and low-cost. Thus, LR-WPAN has become one of the most foreseen technologies enabling WSNs. An example is ZigBee [4], which is an open specification built on the LR-WPAN standard and targeted at low-cost, low-data-rate and low-power wireless networking.

OMNeT++ [2] is a public-source, component-based and discrete event simulation environment and is becoming a very popular simulation platform especially in communication and networking community. Its primary application area covers the simulation of communication networks, IT systems, queuing networks and business processes as well. [5] has shown that OMNeT++ is very suitable for simulating wireless sensor networks owing to its modular structure and using NED language for ease of simulation configuration.

In this paper, we present a simulation model of IEEE 802.15.4 in OMNeT++. It consists of two modules for PHY and MAC layers respectively and supports simulations of star and cluster tree topologies. The PHY model implements the complete functions defined in the specifications. The MAC model implements three data transfer modes (direct, indirect and GTS), beacon transmission and synchronization, complete CSMA-CA mechanism and partial PAN management functions, like association. To measure energy consumption of nodes. we implement an energy model in the MAC module. All parameters defined in the specification except for those for security functions are implemented in the model and adjustable in the OMNeT++ configuration file omnetpp.ini or in a C++ source file where the default values for the majority protocol parameters are stored.

The rest of the paper is organized as follows. In section 2, we give a brief description of 802.15.4. In section 3, the model of IEEE 802.15.4 in OMNeT++ will be introduced in detail. Section 4 concludes the paper and give a vision to the future work.

2. A BRIEF OVERVIEW OF IEEE 802.15.4

In this section, we give a brief overview of IEEE 802.15.4. Only those parts related to our model are introduced.

The IEEE 802.15.4 network can work in one of three ISM frequency bands and choose from a total of 27 channels. Two different types of devices are defined in an LR-WPAN, a full function device (FFD) and a reduced function device (RFD). An FFD can talk to any other device and serves as a PAN coordinator, a coordinator or a device. An RFD can only talk to an FFD node. The standard supports two network topologies, star and peer-to-peer. In the star network, the communication occurs only between devices and a single central controller, called the PAN coordinator, which manages the whole PAN. The peer-to-peer topology also has a PAN coordinator, however is differs from the star topology in that any devices can communicate with any other one as long as they are in range of one another. A special case of peer-to-peer topology is cluster tree, in which a node talks only to its parent or children nodes.

To achieve better energy-efficiency, the IEEE 802.15.4 can operate on beacon-enabled mode, for which a superframe structure is utilized. A super frame is bounded by periodically transmitted beacon frames, which allow nodes to associate with and synchronize to their coordinators. It con-

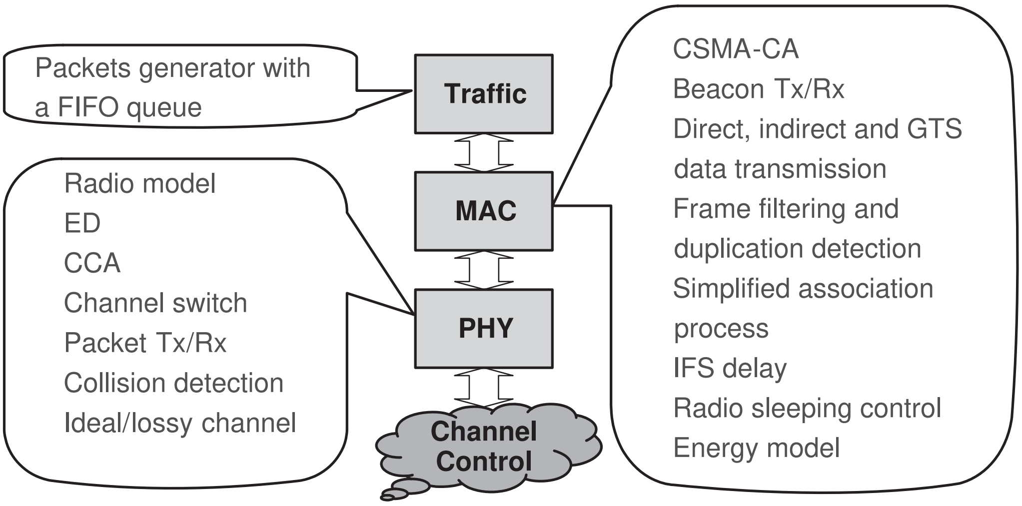


Figure 1: The structure and components of the 802.15.4 model

sists of two parts, active and inactive period. An active portion is divided into 16 contiguous time slots that form three parts: the beacon, contention access period (CAP) and contention-free period (CFP). In CAP, all data transmission should follow a successful execution of the slotted CSMA-CA algorithm. There are two data transfer modes defined in CAP, the indirect transmission for downlink data and the direct transmission for uplink data. In CFP, a device can communicate with the PAN coordinator directly in the called guaranteed time slots (GTS) without contending for the channel using CSMA-CA mechanism. The GTS are allocated by the PAN coordinator, therefor GTS transfer mode is only applicable in the star network.

3. DESCRIPTION OF IEEE 802.15.4 MODEL IN OMNET++

The IEEE 802.15.4 model is developed in the INET framework, which is an open-source communication networks simulation package for the OMNeT++ simulation environment and suited for simulations of wired, wireless and ad-hoc networks. The architecture of the 802.15.4 model is shown in Fig. 1. There are three sub models, traffic, MAC and PHY, each of which is a independent module and inherited from the basic C++ class *cSimpleModule* in OMNeT++. The modules are connected with each other via gates and communicate via messages. A snapshot of the model in the graphical interface of OMNeT++ called Tkenv is shown in Fig. 2. In the following sub sections, each of these modules is introduced in detail.

3.1 PHY Module

In the INET framework, a general radio module called *AbstractRadio* implements the common functionality of the radio, like packet transmission and reception with collision detection, channel switch, etc.. We adapt this module to create a proper radio model conforming to IEEE 802.15.4, by changing or adding the following contents:

- Redefine radio states: according to the specification, the radio is modeled with three states: transceiver disabled (*TRX_OFF*), transmitter enabled (*TX_ON*) and receiver enabled (*RX_ON*). The MAC module completely controls radio operation and can set the radio into different states via a request primitive. The PHY module reports the setting result back to the MAC via a confirm primitive.

- Clear channel access (CCA): the MAC module requires the PHY to perform CCA via the PLME-CCA.request primitive. The channel state (busy or idle) is determined in the model by checking two flags, *isRxing* and *isTxing*, which indicate whether the radio is currently receiving or transmitting packets. The PHY will set the flag *isRxing* by either of the following two facts: a packet (not a noise) is currently being received or the current noise level is above the sensitivity value. For a CCA request, checking flags will take place twice at both the beginning and the end of a period of 8 symbols. The CCA result will be reported back to the MAC via a corresponding confirm primitive.

In addition, the PHY module can be configured to work in any one of a total 27 channels and transmit packets at three different data rates, as defined in the specification. Dynamical channel switch during the simulation is also supported.

3.2 MAC Module

This is the main module in the whole model and contains the following three main parts:

3.2.1 Channel access

Channel access is the core function for any MAC protocol. We have implemented the majority functions and primitives for channel access defined in the specifications, including three data transfer modes (direct, indirect and GTS), MAC frame filtering, detection for duplicate received packets and the most important part - CSMA-CA mechanism

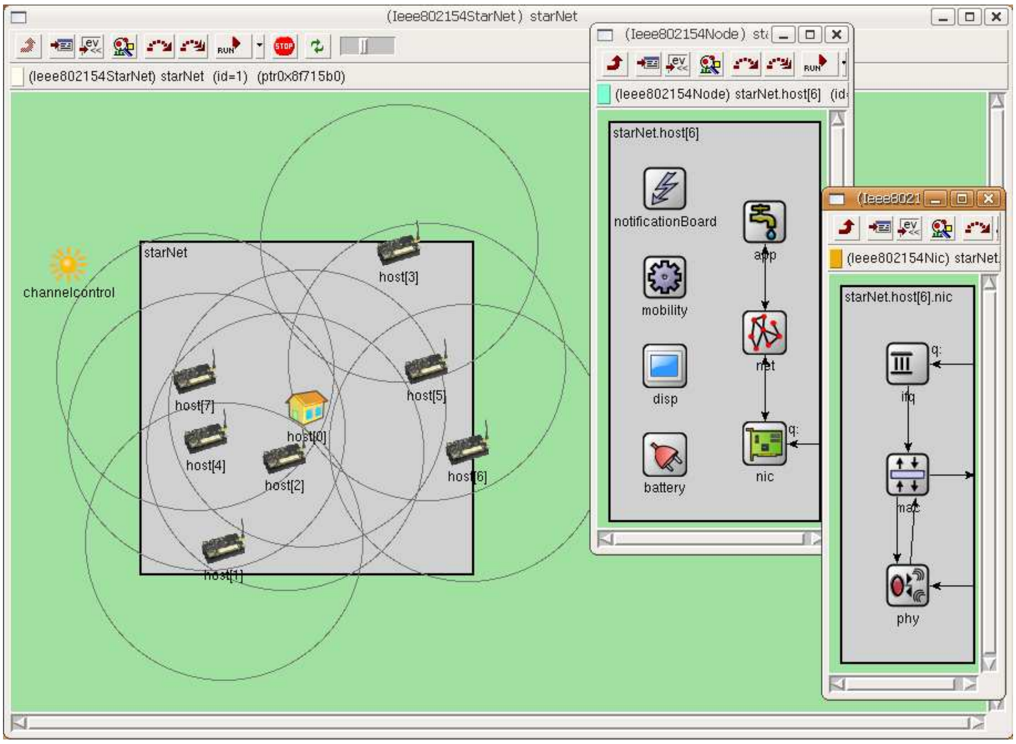


Figure 2: The 802.15.4 model in TKenv, the graphical interface of OMNeT++

(slotted and unslotted). To make the model more accurate, we also consider the IFS delay, which is defined as the required process time for a frame received from the PHY by the MAC.

3.2.2 Beacon mechanism and PAN management

IEEE 802.15.4 achieves good energy-efficiency owing to its beaconing mechanism. We implements in our model the complete beacon transmission and reception for nodes forming star and cluster tree network automatically. For example, to simulate a cluster tree topology, user can specify in the simulation setup phase which node to be the PAN coordinator and when to broadcast its first beacon. During the simulation running, when those nodes being placed one hop away from the PAN coordinator receive their first beacon from the PAN coordinator, they initiate the association process. After successful association, if those nodes are not leaf nodes, they will act as a coordinator and start transmitting beacons to those nodes placed at the next level in the cluster tree. When all leaf nodes are associated, the network forming phase is complete. With respect to PAN management functions, a simplified association process is modeled, that is, the node will associate with the coordinator, from which it receives the first beacon. Furthermore, loss of synchronization due to time drift (e.g. use of oscillator with $\pm 20ppm$) can be simulated with our model. Other PAN management functions, like channel scan and dissociation, are not considered in our model.

3.2.3 Energy model

Energy consumption is always a major interest in studying MAC protocols like IEEE 802.15.4 designed for energy-restricted wireless applications. In our model, we measure only the energy consumed at the radio. To achieve this in the MAC module, we let the MAC module keep tracking the current radio state in the PHY module via a message passing module defined in the INET framework called *NotificationBoard*. Therefor we only need to count the total time that the radio has spent in each state during the simulation. As long as the radio power for each state is known, the total energy consumption can be easily calculated. Although in our model we use a single state RX_ON to represent two possible working state in a real radio, idle listening and receiving, it is reasonable to assume that radios in these two states consume approximately the same power.

3.3 Traffic Module

The traffic module is not a part of IEEE 802.15.4, however we need a traffic generator to run the simulation. This module is modeled as the upper layers above the IEEE 802.15.4 model and acts as both a traffic source generating packets for the MAC module and a traffic sink collecting and analyzing received packets. The traffic module is inherited from the existing traffic generator class, which is implemented by Dietrich Isabel [3] and supports generating traffic with various types, including CBR, Exponential, and On-off. It uses a flexible XML-based parameter structure and supports dynamical change of traffic pattern during the simulation.

Table 1: Model Parameters

transmitterPower	power for sending packets (mW)
sensitivity	carrier sense threshold
thermalNoise	base noise level (dBm)
snirThreshold	signal/noise threshold
isPANCoor	is a PAN coordinator or not
BO	beacon order
SO	superframe order
startTime	when PAN starts beaconing

3.4 Model Parameters

Our model provides a vast amount of parameters adjustable for conducting a comprehensive study of IEEE 802.15.4. The default values for the majority protocol parameters are stored in a single C++ header file. We also put some parameters, which are most likely to be modified during the simulation, into the corresponding NED file for each simple module, so that user can change them conveniently in the simulation configuration file *omnetpp.ini*. Those parameters are listed in Table 1.

4. CONCLUSIONS AND FUTURE WORK

In this paper, we present a simulation model for IEEE Std. 802.15.4 developed in the popular simulation environment OMNeT++. Except for the PAN management and security functions, the model implements the majority parameters and functions defined in the specifications. Compared with the existing IEEE 802.15.4 model in ns-2, our model is built conforming to the latest IEEE Std. 802.15.4-2006 and implements the GTS data transfer mode, as well as an energy model. In the future work, more PAN management functions will be added to support simulating more complicated scenarios, e.g. mesh topology with ZigBee routing. A series of simulations for evaluating the performance of IEEE 802.15.4 in the QoS aspect are running. We also plan to integrate the security functions into our model and investigate how the performances are affected by those security mechanisms.

5. REFERENCES

[1] IEEE std. 802.15.4-2006, wireless medium access control (MAC) and physical layer (PHY) specifications for low-rate wireless personal area networks (WPANs).

[2] OMNeT++ homepage. [online]. available: http://www.omnetpp.org.

[3] OMNeT++ page at Isabel Dietrich's homepage. [online]. available: http://www7.informatik.uni-erlangen.de/ isabel/omnet/.

[4] ZigBee Alliance homepage. [online]. available: http://www.zigbee.org.

[5] C. Mallanda, A. Suri, V. Kunchakarra, S. S. Iyengar, R. Kannan, A. Durresi, and S. Sastry. Simulating wireless sensor networks with omnet++.

ServiceCast: Eine Architektur zur dienstorientierten Kommunikation in selbstorganisierenden Sensor-Aktor-Netzen

Andreas L. Kuntz
Institut für Telematik
Universität Karlsruhe (TH)
akuntz@tm.uka.de

Martina Zitterbart
Institut für Telematik
Universität Karlsruhe (TH)
zit@tm.uka.de

ABSTRACT

ServiceCast ist eine Architektur zur dienstorientierten Kommunikation in selbstorganisierenden Sensor-Aktor-Netzen (SSAN). Ziel ist es Methoden der Dienstorientierug auf niederen Protokollschichten verfügbar zu machen, um im gesamten Stack einem einheitlichen Kommunikations-Paradigma folgen zu können. Als zentrales Merkmal werden *Dienste* statt Knoten als Grundlage der Adressierung und des Routings betrachtet. Die folgenden von ServiceCast behandelten Aspekte sind für die Kommunikation in Selbstorganisierenden Sensor-Aktor-Netzen essentiell und daher von besonderem Interesse: Dienstorientierte Adressierung, Auffinden von Diensten, Verwaltung von Verbindungen sowie die verteilte Konfiguration und Initialisierung der Dienstinstanzen im Netz.

1. EINLEITUNG

Sensornetze werden üblicherweise als aus vielen kleinen, ressourcenarmen Knoten bestehend angenommen. Die Knoten erbringen dabei Dienste, welche sie anderen Knoten zur Verfügung stellen. Aufgrund der häufig beschränkten Energieressourcen, wird in vielen Arbeiten davon ausgegangen, dass Knoten temporär in Energiesparmodi wechseln, wodurch auch die vom Knoten erbrachten Dienste zeitweise nicht verfügbar sind. Daneben sind zahlreiche weitere Ursachen denkbar, welche zu Dynamik und damit zu ähnlichen Auswirkungen führen. Wir gehen davon aus, dass ein Großteil der Dienste redundant im Netz vorhanden ist, also mehrere Instanzen desselben Dienstes auf unterschiedlichen Knoten zur Verfügung stehen. Durch geeignete Abstraktion der Adressierung der Kommunikationspartner möchten wir diese Redundanz ausnutzen, um Nachrichten transparent an aktive Diensterbringer auszuliefern und damit die Verfügbarkeit der erbrachten Dienste zu erhöhen. Wir halten Dienstorientierung auf niederen Protokollschichten für einen vielversprechenden Ansatz, um eine geeignete Abstraktion zu realisieren.

Aufgrund der Annahme, dass Dienste durch mehrere Instanzen redundant vertreten sind, kann der Dienstbezeichner alleine den Kommunikationspartner nicht eindeutig identifizieren. Daher führt ServiceCast als orthogonales Merkmal Kontexte ein, welche Sensorknoten ihrer Umgebung zuordnen. Kontexte erlauben, das „wo“ in einer vom Benutzer beeinflussbaren Granularität zu definieren (z. B. „Lokation innerhalb eines Rechtecks mit den Eckpunkten $(\vec{x}, \vec{y})$“), während der Dienstbezeichner das „was“ definiert (z. B. „Temperaturdienst“).

Anstatt Knoten und Dienstinstanzen direkt zu adressieren, werden Nachrichten mit einem Ziel-Kontextbereich und einem Ziel-Dienstbezeichner versehen. Aufsuchen des Kontextes und Auswahl der Dienstinstanz werden vom Netz übernommen. So ist weder explizit das Übersetzen von Dienstbezeichner auf die ID des diensterbringenden Knotens notwendig, noch muss auf temporäre Nicht-Erreichbarkeit eines Dienstes Rücksicht genommen werden.

Architekturen zur Dienstfindung wie [3, 4, 5, 9] stützen sich auf Dienstverzeichnisse. Das Dienstverzeichnis bildet Dienstbezeichner auf die Adressen der Wirtsknoten ab, welche als Grundlage der späteren Kommunikationsbeziehung dienen. ServiceCast kommt ohne ein solches Verzeichnis aus. D. h. Dienstbezeichner müssen nicht in eine netzweit gültige Knotenadresse umgesetzt werden.

Im Folgenden wird der Ansatz anhand eines kurzen Beispiels erläutert. Auf Kontexte, Dienstfindung und Instanzauswahl wird anschließend eingegangen. Aspekte des Routings bezüglich der Kontexte sollen hier nicht weiter vertieft werden, da für viele mögliche Kontexträume bereits geeignete, spezielle Ansätze existieren [1, 7, 8]. Das Anycast-ähnliche Adressierungsschema erfordert besonderes Augenmerk bei der Zustellung von Nachrichtenfolgen, wenn mehrere, zueinander in Beziehung stehende Nachrichten an dieselbe Dienstinstanz übermittelt werden müssen. Auf diesen Punkt wird in Abschnitt 2.4 kurz eingegangen. Abschließend werden Aspekte der verteilten Konfiguration, der Dynamik und der Sicherheit kurz angerissen. Eine umfassende Behandlung derselben ist für zukünftige Arbeiten vorgesehen.

2. KONZEPT

ServiceCast adressiert Kommunikationspartner durch die Kombination von Ziel-Kontextbereich und Ziel-Dienstbezeichner. So werden durch dieselbe Adresse potentiell alle gleichwertigen Dienstinstanzen innerhalb des angegebenen Kontextbereiches angesprochen. Die Instanzauswahl wird transparent vom Netz vorgenommen, was die effektive Verfügbarkeit des Dienstes verbessert. ServiceCast setzt keine netzweit eindeutigen Knoten-Identifikatoren voraus; einzig lokal eindeutige Identifikatoren sind notwendig, um Nachbarknoten voneinander unterscheiden zu können.

Nachrichten werden in zwei Schritten zur Ziel-Dienstinstanz

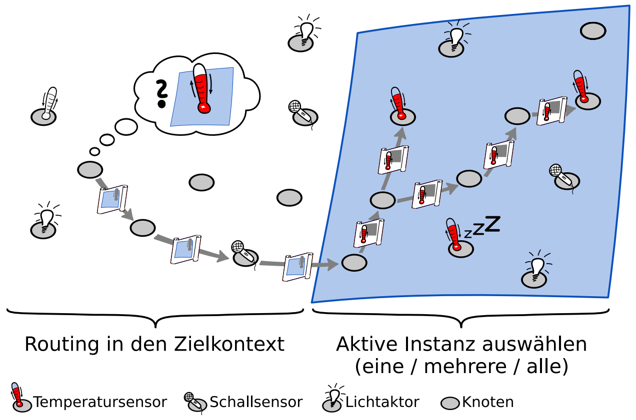


Figure 1: Adressierung und Routing mit ServiceCast

geleitet: (1) Routing der Nachricht in den Ziel-Kontext, (2) Auswahl der Ziel-Dienstinstanz. Abbildung 1 zeigt ein Beispielszenario: Sensorknoten sind durch Ovale angedeutet, Dienste durch die Piktogramme auf den Knoten. Das Beispiel-Netz bietet Temperatur- und Schallsensoren, sowie Lampen als Aktoren. Sensoren und Aktoren werden jeweils durch die zugehörigen Dienste gekapselt. Als Kontextraum soll hier beispielhaft die Lokation der Knoten in einem zweidimensionalen Koordinatensystem dienen.

Möchte beispielsweise ein Knoten die Temperatursensoren innerhalb eines bestimmten Bereiches abfragen, versendet er eine Nachricht, in welcher der gewünschte Bereich als Ziel-Kontextbereich codiert ist. In der Abbildung ist dieser durch das graue Feld rechts gekennzeichnet. Als Ziel-Dienstbezeichner trägt der Knoten „Temperaturdienst" in die Nachricht ein. Die Nachricht wird zunächst — ohne Berücksichtigung des Zieldienstes — in den „grauen" Kontextbereich geroutet (Abbildung 1, links). Erst innerhalb des Ziel-Kontextbereiches wird der Ziel-Dienstbezeichner „Temperaturdienst" ausgewertet und die Nachricht an die aktiven Instanzen des Temperaturdienstes geleitet (Abbildung 1, rechts). So kann z. B. die temporäre Nicht-Verfügbarkeit eines Dienstes, verursacht durch einen schlafenden Knoten, kompensiert werden. Ein Parameter in der Nachricht gibt an, wieviele der Instanzen die Nachricht erhalten sollen. Derzeit sind „genau eine", „mehrere" und „alle" als mögliche Werte vorgesehen.

2.1 Kontexte

Sensoren beobachten Phänomene ihrer Umwelt. Gemessene Sensorwerte sind an die Lokation ihrer Messung gebunden und verlieren ohne diese Information an Aussagekraft. Ein Kontext ordnet einen Sensorknoten seiner Umgebung zu und soll den Begriff der Lokation eines Sensorknotens verallgemeinern bzw. von ihm abstrahieren. Kontexte können symbolisch, diskreter Natur sein (z. B. die Nummer eines Raumes innerhalb eines Gebäudes oder die symbolische Identität eines Patienten) oder auch kontinuierlich (z. B. eine geographische Koordinate). Kontexte sollen als abstraktes Konstrukt gesehen werden und sind nicht auf die hier angeführten Beispiele beschränkt. Anforderungen an Kontexträume sind im diskreten Fall zwei Kontexte auf Gleichheit prüfen zu können, im kontinuierlichen Fall der Vergleich bezüglich einer zum Kontextraum gehörenden Ordnungsrelation. Kontexte erhöhen einerseits die Spezifität einer Anfrage indem sie die Menge der adressierten Knoten einschränken. Andererseits erlauben sie die Kompensation der Auswirkungen von Dynamik, z. B. im Falle von schlafenden Knoten.

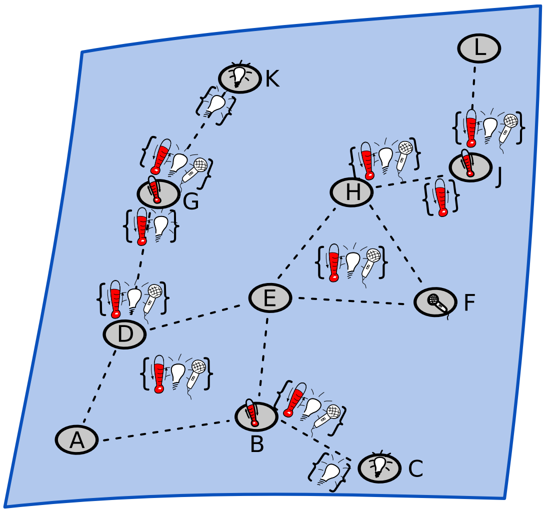


Figure 2: Trans Service Sets

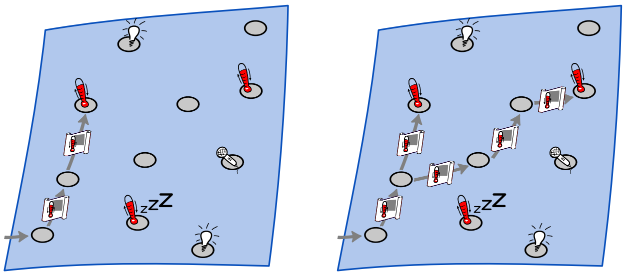
2.2 Auffinden von Diensten

Da ServiceCast Dienste als Grundlage des Routings verwendet, wird kein explizites Verzeichnis benötigt, welches Dienstbezeichner auf die IDs der diensterbringenden Knoten abbildet. Stattdessen werden Dienstinstanzen anhand von Dienstverfügbarkeitsinformationen gefunden. Diese werden periodisch zwischen benachbarten Knoten ausgetauscht. Während ein Knoten initial nur seine lokal angebotenen Dienste, das „Local Service Set" (LSS) kennt, lernt er mit der Zeit welche Dienstmengen über seine Nachbarknoten direkt und indirekt verfügbar sind. Eine solche Dienstmenge wird als „Trans Service Set" (TSS) bezeichnet. TSSs lassen sich als Erreichbarkeitsrelation wie folgt definieren: „Das $TSS(x, y)$ ist die Menge aller über einen Knoten x erreichbaren Dienste aus Sicht des Nachbarknotens y." TSSs sind also immer für ein Paar von benachbarten Knoten definiert. Genauer:

$$TSS(x, y) := \left(\bigcup_{\substack{n\in\mathcal{N}(x)\\ n\neq y}} TSS(n, x) \right) \cup LSS(x),$$

wobei $\mathcal{N}(x)$ die Menge aller zu x benachbarten Knoten ist.

Abbildung 2 verdeutlicht dies an einem Beispiel. Die Kanten zwischen den Knoten deuten die Nachbarschaftsbeziehungen an. Die Dienstsymbole in geschweiften Klammern repräsentieren das jeweilige TSS, also die Menge der Dienste, welche der Knoten über die entsprechende Verbindung seinem Nachbar verfügbar macht. In diesem Beispiel wurden bidirektionale Kanten angenommen. Ferner wurden die Knoten durch Buchstaben mit Namen versehen. Es sei hier noch einmal ausdrücklich erwähnt, dass die eindeutige Unterscheidbarkeit der Knoten keine für ServiceCast notwendige Voraussetzungen ist, sondern hier nur der einfacheren



(a) eine aktive (b) alle aktiven

Figure 3: Auswahl (a) einer bzw. (b) aller aktiven Instanzen des Temperaturdienstes

Beschreibung dient. ServiceCast kommt mit lokal eindeutigen Knoten-Identifikatoren aus. Innerhalb von Zyklen (z. B. ABDE oder EFH) steht an jeder Kante das selbe TSS. Vereinfachend ist dieses nur einmal in der Mitte des entsprechenden Zyklus eingetragen. Anders verhalten sich die TSSs am Rand des Netzes. Knoten K ist beispielsweise über Knoten G mit dem restlichen Netz verbunden. So enthält hier das TSS(G, K) alle im Netz verfügbaren Dienste, da K jeden einzelnen über G erreichen kann. Umgekehrt enthält TSS(G, D) nur die von G und K angebotenen Dienste.

Das Beispiel macht deutlich, dass in stark vermaschten Netzen TSSs alleine nur wenig Information zur Routingentscheidung beitragen können. Denn TSSs spiegeln die Erreichbarkeit von Diensten wider und codieren inhärent die Menge aller alternativen Pfade zu jedem der Dienste. TSSs repräsentieren also die eingangs erwähnte Redundanz der Dienste. Um sinnvolle Entscheidungen bezüglich des Routings treffen zu können, wird das Einbeziehen zusätzlicher Informationen wie z. B. die „Entfernung zur nächsten Dienst-Instanz“, Informationen über „Multiplizitäten“ oder über die Energiereserven des Wirtsknoten notwendig. Untersuchung und Evaluation der Einsatzfähigkeit und des Nutzens solcher Zusatzinformationen, sowie die Integration in ServiceCast ist für weiterführende Arbeiten geplant.

TSSs lassen sich z. B. mittels Bloomfilter [2] kompakt, und mit konstantem Speicherbedarf verwalten. Da für jeden Nachbarknoten exakt ein TSS gepflegt werden muss, wächst der gesamte Speicherbedarf linear mit der Anzahl der direkten Nachbarn pro Knoten (bzw. mit der Dichte der Knoten) und mit der maximalen Anzahl der Dienste. Wir erwarten, diese Anzahl für viele Applikationen einheitlich nach oben abschätzen zu können.

2.3 Dienstinstanz Auswahl

Innerhalb des Ziel-Kontextbereiches sind gleichwertige Dienstinstanzen austauschbar, falls (1) die Kommunikation mit einem Dienst aus einer einzigen Nachricht besteht oder (2) während einer längeren Verbindung zu einem Dienst kein Zustand innerhalb des Dienstes aufgebaut wird oder (3) der Zustand vollständig auf andere Dinstinstanzen repliziert werden kann. Die Behandlung von Nachrichtenfolgen, in welchen die einzelnen Nachrichten zueinander in Beziehung stehen, wird im nächsten Abschnitt besprochen. Ein Parameter in der Nachricht gibt an, wieviele der aktiven Instanzen die Nachricht erhalten sollen. Derzeit vorgesehen sind die Werte: „genau eine“ (=1), „mehrere“ ($\geq$ 0) und „alle“ (all). Der Wert „genau eine“ realisiert einen dienst- und kontextspezifischen Anycast, d. h. es wird eine beliebige, aktive Instanz des Ziel-Dienstes angesprochen, welche sich im Ziel-Kontextbereich befindet (siehe Abbildung 3(a)). Da hier implizit eine Garantie („genau eine“) ausgesprochen wurde, ist es notwendig für diese Art der Zustellung in den weiterleitenden Knoten temporär Zustand zu halten. Der Wert „mehrere“ spricht beliebige aktive Instanzen des Ziel-Dienstes innerhalb des Ziel-Kontextbereiches an. Undefiniert viele weitere Instanzen im Zielkontext können die Nachricht ebenfalls erhalten. Ein dienst- und kontextspezifischer Broadcast wird durch den Wert „alle“ bereit gestellt, welcher die Nachricht an alle aktiven Instanzen des Ziel-Dienstes innerhalb des Kontextbereiches übermitteln lässt (siehe Abbildung 3(b)).

2.4 Verbindungsorientierte Kommunikation

Obwohl angenommen werden kann, dass ein großer Teil des Datenaufkommens in Sensornetzen in oben beschriebener Weise behandelbar ist, sind Fälle denkbar in denen es notwendig sein wird den Kommunikationspartner temporär zu fixieren, also eine länger dauernde Kommunikationsverbindung mit derselben Instanz eines Dienstes (oder Gruppe von Instanzen) zu ermöglichen. Insbesondere im Umgang mit Aktoren kann dies wichtig sein. Für diesen Fall sollen — ähnlich der in [6] vorgeschlagenen Architektur — sogenannte Pseudo-Dienste eingeführt werden (in [6] „Transaction Identifier“ genannt). Pseudo-Dienste tragen probabilistisch und temporär netzweit eindeutige Dienstbezeichner. Sie werden bei Bedarf dynamisch instantiiert und markieren den Pfad, welchen die Nachricht durch das Netz genommen hat. Folgenachrichten enthalten den Pseudo-Dienst als Zieldienst, wodurch sie zu den selben Ziel-Dienstinstanzen wie die inititale Nachricht gelangen. So kann ein einzelner oder eine Gruppe von Kommunikationspartnern temporär fixiert werden. Die Pseudo-Dienste etablieren einen Softstate in den Zwischenknoten, welcher nach Nichtbenutzung der Verbindung wieder abgebaut wird.

2.5 Verteilte Konfiguration

Knoten sollen zunächst in die Zielregion ausgebracht und erst anschließend konfiguriert bzw. programmiert werden. Aus energietechnischen Gründen kann es sinnvoll sein, Sensorknoten von schon initial mit den applikationsspezifischen, zur Hardware passenden Diensten auszustatten, aber erst während der Konfigurationsphase zu entscheiden, welche von ihnen gestartet werden. Während der Konfigurationsphase soll durch lokale Kommunikation benachbarter Knoten entschieden werden, welche Dienste auf einem Knoten zur Verfügung gestellt werden. Dabei sollen Parameter wie die Anzahl der auf (möglicherweise indirekt) benachbarten Knoten verfügbaren Instanzen eines Dienstes oder die Lage des Knotens innerhalb des Kontextraumes eine Rolle spielen. Denkbar wäre z. B. eine gleichmäßige Verteilung von Dienstinstanzen bei einer vorgegebenen mittleren Dichte. Genauso sind andersartige Verteilungen (z. B. Gauss, etc.) als Vorgabe vorstellbar.

2.6 Dynamik

Neben Knoten-Mobilität sind Reprogrammierung oder Rekonfiguration von Knoten, Ausbringen neuer Knoten bzw. deren Wegfall, Kommunikationsstörungen und temporäre Unerreichbarkeit aufgrund von Energiesparmaßnahmen wichtige Ursachen für Dynamik. Allen Ursachen gemeinsam ist eine Veränderung der Topologie des Netzes, welche Auswirkungen auf Routing, das Auffinden von Diensten und andere Funktionalitäten haben kann. Durch die mit ServiceCast eingeführte abstrakte, topologiefreie Adressierung durch Dienstbezeichner und Kontexte, lassen sich Auswirkungen von Dynamik transparent kompensieren. Denn anstatt eine spezielle Dienstinstanz (oder einen Knoten) als Kommunikationspartner zu wählen, wird durch den Dienstbezeichner nur die Art des Dienstes spezifiziert, die Auswahl der Instanz(en) bleibt dem Netz überlassen.

2.7 Sicherheit

Zur Realisierung von Sicherheitszielen wie Integrität, Vertraulichkeit, Authentizität, Autorisation, etc. sind Sicherheitsprimitive wie Hashing, Verschlüsselung und ein Vertrauensanker notwendig. Kritisch ist in diesem Falle der Vertrauensanker zu betrachten. Ist er klassisch fest einem Knoten zugeordnet, muss für ServiceCast ein neuer Ansatz gefunden werden, da Knoten hier als austauschbar, anonym und frei von jeglichen fixen IDs angenommen werden. Darüber hinaus sollen Dienste unabhängig vom Wirtsknoten, also verlagerbar sein, was ebenfalls gegen einen an den Knoten gebundenen Vertrauensanker spricht.

Der Weg, welcher mit ServiceCast beschritten werden soll, ist Sicherheit als dienstbasiertes Konzept zu realisieren. Anstatt z. B. einen Knoten zu authentifizieren, gilt es die Integrität eines Dienstes zu überprüfen. Andere Ziele sind analog auf Dienste zu übertragen.

3. ZUSAMMENFASSUNG

Mit ServiceCast wurde ein Ansatz vorgestellt, welcher Methoden der Dienstorientierung auf niederen Protokollschichten verfügbar macht. Die Adressierung über Dienste, anstatt durch knotenspezifische Adressen, erlaubt eine Abstraktion der Kommunikation von der Topologie des Netzes. Dienstinstanzen werden innerhalb des Zielkontextbereiches lokal vom Netz ausgewählt, wodurch Dynamik-Auswirkungen für den Benutzer transparent kompensiert werden. Dadurch erwarten wir ein verbessertes Verhalten bezüglich der Verfügbarkeit von Diensten in dynamischen Netzen. ServiceCast kommt ohne explizites Dienstverzeichnis aus, was einen im Vergleich zu anderen Architekturen geringeren Protokolloverhead erwarten lässt. Darüber hinaus verlangt ServiceCast keine fixen, netzweit eindeutigen Knotenidentifikatoren, wodurch Knoten austauschbar und anonym werden.

4. DANKSAGUNG

Dank gilt es meinem geduldigen Kollegen Hans-Joachim Hof zu sagen für viele fruchtbare Diskussionen.
Diese Arbeit wurde im Rahmen des DFG-Graduiertenkollegs 1194 „Selbstorganisierende Sensor-Aktor-Netzwerke“ durch die Deutsche Forschungsgemeinschaft gefördert.

5. REFERENCES

[1] Brad Karp and H. T. Kung. GPSR: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254, New York, NY, USA, 2000. ACM Press.

[2] Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, 1970.

[3] K. Edwards and T. Rodden. *Jini Example by Example.* Prentice Hall PTR Upper Saddle River, NJ, USA, 2001.

[4] Hans-Joachim Hof and Martina Zitterbart. SCAN: A secure service directory for service-centric wireless sensor networks. *Computer Communications*, 28(13):1517–1522, Aug. 2005. ISSN 0140-3664.

[5] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[6] Jeremy Elson and Deborah Estrin. Random, Ephemeral Transaction Identifiers in Dynamic Sensor Networks. In *Proceedings of the Twenty First International Conference on Distributed Computing Systems (ICDCS-21)*, pages 459 – 468, Phoenix, Arizona, April 2001.

[7] Matthias Gauger, Pedro José Marrón, Marcus Handte, and Kurt Rothermel. Routing in Sensor Networks based on Symbolic Coordinates. In P. Marrón, editor, *5. GI/ITG KuVS Fachgespräch "Drahtlose Sensornetze"*, Technischer Bericht 2006/07, pages 81–86. Stuttgart: Universität Stuttgart, Juli 2006.

[8] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, Nov.-Dec. 2001.

[9] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Schenker. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 161–172, New York, NY, USA, 2001. ACM Press.

Coordinated group adaptation in sensor networks

Daniel Minder Pedro José Marrón Andreas Lachenmann Kurt Rothermel
Universität Stuttgart, IPVS, Germany
{minder|marron|lachenmann|rothermel}@ipvs.uni-stuttgart.de

ABSTRACT

In Wireless Sensor Networks, several algorithms are used to perform different functionality, e.g. routing or clock synchronization. Each algorithm is intended for specific network characteristics and user requirements. But the acutal characteristics and requirements may change during system runtime. TinyCubus and particularly its Tiny Data Managment Framework use adaptation to solve this problem.

In this paper, we first explain the centralized adaptation process. Then, we examine how this can be done localized in the network. Since coordination between local adaptation decisions is found to be necessary, metrics for this coordination and their dependencies are shown.

1. INTRODUCTION

Sensor Networks are used today in several domains to monitor real-world phenomena, e.g. in logistics, health care, biological studies or smart offices. Due to their small dimensions and their autonomous operation, sensor nodes can be installed in places where traditional monitoring is complicated or even impossible. Since the research is still young we expect further application areas in the future.

Despite the variety in domains, most applications share a common set of basic algorithms, e.g. routing, clustering or time synchronisation. However, each particular algorithm is suitable for a set of environments, e.g. mobile or static, where it exhibits best performance. When developing an application, it is a cumbersome task for the programmer to select the algorithms for a specific environment.

Moreover, the conditions of many systems change over time. Consider a logistics application where goods may be stored for several days but are moved to different places using various means of transportation. A single algorithm is unlikely to work in all settings. Therefore, the performance of the sensor network changes significantly over time. The user of the sensor network might change her requirements during runtime as well. For example, he could decide that a high delivery ratio is now more important than low latency. This has to change the behavior of the algorithms as well.

To cope with the changing environment and changing user requirements, we presented the TinyCubus system [1]. Its Tiny Data Management Framework (TDMF) is able to select appropriate algorithms or parameterizations for an algorithm based on network conditions, user requirements and algorithm characteristics.

In larger networks, a centralized adaptation has a high overhead since the network conditions have to be collected at a single node. Moreover, larger networks are likely to be diverse so that different parameterizations for different groups of nodes result in a better behavior than a global setting. Since adjacent groups influence each other, the separate adaptation results of the groups have to be coordinated.

In this paper, we start with a centralized adaptation approach and show how the adaptation can be localized in the network. Then, we examine how the local adaptation decisions can be coordinated to achieve better network performance.

The rest of the paper is organised as follows: In Section 2, the adaptation process of TDMF is shown. The locality of the adaptation is examined in Section 3. Section 4 then deals with the coordination of group adaptation results. The paper concludes with Section 5.

2. ADAPTATION PROCESS

2.1 Simulation

The basic assumption of TDMF is that the accuracy of simulation is sufficient to derive properties for real-world sensor networks. Each algorithm can thus be evaluated for different settings with respect to several performance parameters P. Some parameters, like power consumption, are general for all algorithms, others, like the delivery ratio for routing algorithms, are specific for one particular class. These performance parameters are the output of the simulation and are, therefore, measured. The simulation process can be controled by several input parameters that are divided into groups as well: Some parameters N influence the whole network, e.g. node density or mobility; some parameters A only the algorithms, e.g. the maximal number of retransmissions of a routing algorithm.

The user of TDMF has to decide which input parameter space is relevant for him. For example, in a network that remains static it is of no use to simulate mobile nodes. Depending on the available computing power for simulation, the granularity of the input parameters N and A is chosen. Especially for continuous parameters like network density, reasonable increments have to be set. In a second step, the parts of the first simulation step where the output parameters exhibit a high rate of change can be simulated in more detail.

Input parameters and measured output parameters describe the behavior of an algorithm and are, therefore, called its

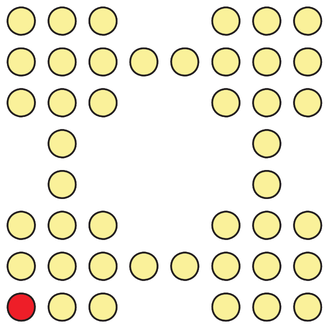


Figure 1: Scenario 1

meta-data. In TDMF, each algorithm module is annotated with this meta-data.

2.2 User preferences

The user of a sensor network has requirements that the application has to fulfil. They can be expressed as an inequation using the elements of P. For example, the user could specify that the estimated lifetime of the sensor network should be at least 1 year and the delivery ratio should be at least 80%.

Two interesting cases can happen: In the first case, more than one algorithm or more than one parameterization of an algorithm fits these requirements. In the second case, none is found. TDMF provides a way for resolving both. If more than one fits, the use can specify an optimization parameter and a direction. For example, the lifetime should be optimized in such a way that the network lives the longest. On the other hand, if no algorithm is found, the user lists the requirements which can be relaxed, e.g. the delivery ratio from 80% to 50%.

2.3 Adaptation process

Input and output parameters change in adaptation compared to the simulation. The network parameters N cannot be influenced by the user or the system but are given by the environment and must be measured. The performance parameters P are given by the user as described above. Thus, the adaptation systems selects the algorithms whose metadata match N and the conditions over P and performs necessary optimization or relaxation to chose exactly one algorithm. Then, this algorithm is installed and parameterized with parameters A if needed.

2.4 Example

To test and evaluate the adaptation system, we used the Avrora simulator [3]. 44 nodes are arranged in a grid as shown in Figure 1 with 10m distance in horizontal and vertical direction. The base station is located in the lower left corner.

Each node sends a data packet every 10 seconds to this base station. A routing algorithm implementing the GEM metric [2] is used. Two parameters A can be adapted in this algorithm: the transmission power and the maximal number of retransmissions per packet. In simulation, 9 different transmission power levels and 5 different retransmission limits, thus 45 different combinations in total, are evaluated.

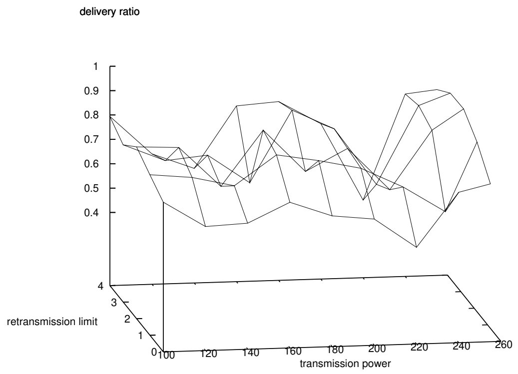


Figure 2: Parameter space for scenario 1

The performance parameters P the delivery ratio and the energy needed are measured. Figure 2 shows the delivery ratio for all 45 simulations.

A very simple adaptation goal could be to maximize the delivery ratio. The adaptation process would select the parameterization “transmission power = 255” and “retransmission limit = 3” which leads to a delivery ratio of 88% on average.

3. DECENTRALIZED ADAPTATION

3.1 Purely Local Adaptation

In the previous global and centralized adaptation approach, messages have to be sent through the network to collect the network parameters and to propagate the adaptation decision. Since the adaptation is often used to maximize the network lifetime, this adaptation message overhead has to be minimized. If the adaptation decision can be made purely local, there would be no message overhead.

We used the simulation results of the previous example and evaluated each of the 45 simulations per node, resulting in a single parameter curve for each node. Then, the adaptation process is executed as explained before on each node locally, resulting in different and independent parameter values for each node.

According to the simulation basis, the worst delivery ratio of a node should be 80%. But in fact, the average ratio for the whole adapted network was 60%, with 29% for the worst single node. The reason for this is that the simulation basis was created using the same parameterization for the complete network, but TDMF uses these values to adapt each node separately. Interferences between nodes were not taken into account.

To cover the interferences, all possible combinations would have to be simulated. This is not possible since it would lead to 45^{44} (or approximately $5.51 \cdot 10^{72}$) combinations in the example. Additionally, the adaptation process need to be done centralized again to assign a suitable parameter combination to all the nodes which contradicts the local approach.

3.2 Group Adaptation

A balance between global and local adaptation is the introduction of groups. Inside a group, the same parameterization is used. The behavior of a group is, therefore, expected to be more stable and to resemble the basic simulations that used the same parameters for the whole network. The disadvantage of this approach is that network parameters have to be collected and the adaptation decision has to be announced, but both happens in a smaller area than in the global case.

The example network is divided horizontally and vertically in the middle, thus forming 4 groups. The 45 basic simulations are evaluated per group, thus forming a new adaptation basis for the group adaptation. The adaptation process uses these per-group results to find optimal settings for each group.

An overall delivery ratio of 89% should be achieved according to the simulation basis. The resulting ratio of 71% is lower than expected, but the gap is lower than in the purely local adaptation.

3.3 Coordinated Groups

Most of the packet losses in the group adaptation example occur at group transitions. In the example, the transmission power of the group with the base station was set to 180, while all other groups use a transmission power of 255. The simulation shows that the packet loss is much higher at the two nodes connecting two groups if the transmission power differs between these groups.

Therefore, to achieve better delivery ratios the parameters of the groups have to be coordinated. Each adaptation process selects not only the best parameter setting for each group but the best k settings. We have chosen $k = 3$. A coordination metric calculates the distance of the transmission power settings for all possible combinations of the settings. Finally, the combination of parameters with the least distance is selected.

The coordination introduced in this approach revives a centralized step that was eliminated when moving from the global to the local view. The difference is that only a few parameters have to be joined at a single place and not all the network conditions from all the nodes.

Simulation shows that using a combination with the highest distance value has a low delivery ratio of 69% while the lowest distance values lead to high delivery ratios of 87%, which is very close to the predicted 89%. This dependency is also expressed by a Pearson's correlation coefficient of -0.78. This coefficient measures the mutual dependency of two variables and ranges from -1 to 1. A value of -1 or 1 show that there is a linear relationship, a value of 0 indicates that a linear model is inappropriate.

4. GENERAL METRICS

Not all scenarios allow to find a simple coordination metric. Moreover, a coordination metric suitable for one scenario might be useless in another. We, therefore, introduce a second scenario with a larger network and change the grouping and parts of the algorithm.

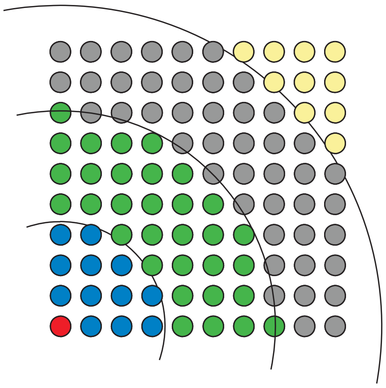


Figure 3: Scenario 2 with circular groups

4.1 Topology

In Scenario 2, 100 nodes are placed on a regular grid with 10m distance in either direction forming a square of 10x10 nodes. 4 groups of 25 nodes each are formed by dividing the network horizontally and vertically in the middle. The base station is still in the lower left corner. Again, the adaptation basis is determined by simulating all 45 combinations of transmission power and retransmission limit for the whole network and by evaluating these simulations per group. Then, the adaptation process selects the best 3 parameter settings for each group.

When applying the distance function of the first scenario as coordination metric to this scenario, "wrong" parameter combinations for the groups are selected. A correlation coefficient between distance metric and delivery ratio for all 81 combinations of 0.21 shows that the metric is less significant in this scenario, but indeed more distant transmission power levels are better than close ones. This contradicts the finding from the first scenario.

4.2 Different Grouping

The analysis of the previous experiment showed a problem in the center of the network where four different groups are adjacent. Such a constellation should be avoided by a clever grouping. Therefore, we examined another method with groups based on the distance from the base station. Figure 3 shows such a grouping with equidistant radii.

The 45 basic simulations of Scenario 2 were reevaluated using this new grouping and, again, the best 3 parameter settings for each group were determined. The simulation of all 81 combinations and the calculation of the correlation coefficient between delivery ratio and the distance metric show that, with a coefficient of -0.50, this grouping reacts similarly to differences in the transmission power as the first scenario.

4.3 Different Routing Metric

The groupwise adaptation of an algorithm is based on the assumption that the algorithm works inside a group independently from the surrounding groups. The previous examples show that this is not the case. Especially intelligent routing metrics react very sensitive to changing traffic due to changed neighboring groups. Thus, these metrics may counteract the adaptation goal.

Therefore, a scenario using static routing trees was set up. The trees were built before simulation by taking into account the radio model characteristics and by trying to balance the number of child nodes in each tree node. That way, a group shows similar behavior also with different adjacent groups.

After simulating the 81 combinations of the best 3 settings for each group based on 45 new basic simulations, the correlation coefficients were calculated. With quadratic groups, the correlation is -0.33, but 0.44 when using circular groups. Compared to the simulations that used the GEM routing metric, the static trees react diametrically opposed to differences in the transmission power.

4.4 Coordination Metrics

As the previous sections have shown, the simple distance metric developed for the example scenario of Section 2 does not work in all cases of the enlarged second scenario. Therefore, a different metric for this scenario has to be found. Since constructive approaches were not successful, possible metrics were tested systematically.

A metric is an addition of single components that are based on the algorithm parameters A. In the given scenario, the tranmission power or retransmission limit of a single group or the "distance" of the transmission power or the retransmission limit between two groups are used. A component is built using a parameter mapped to the interval $[0, 1]$. The $value$ resulting from the mapping is fed into the metric. Each component can also be used as $1 - value$ in the metric.

Having 4 groups, this leads to 40 different single components that could be part of a metric. Of course, some combinations make no sense to be used at the same time, e.g. $value$ and $1 - value$. Moreover, a metric consisting of too many single components is likely to be useful for a specific setting only. Therefore, we limited the search to 6 single components.

To evaluate each metric, 9 different settings were used: 5 using the GEM metric and 4 using static trees. The settings with GEM metric are further divided into 3 using quadratic groups and 2 using circular groups, the settings with static trees are divided into 2 settings of each grouping. The single settings differ again in further adaptation restrictions like energy.

For each metric and each setting, Spearman's rank correlation coefficient between the metric and the average delivery ratio of the network was calculated. The rank correlation coefficient does not assume a linear relationship between the variables and is, therefore, more suitable, for the abstract metric. The metrics were finally sorted by the average correlation coefficient, thus selecting metrics that pedict well the quality of a parameter combination.

As Figure 4 shows, a good metric covering all 9 settings could be found, exhibiting an average correlation coefficient of 0.30, but ranging up to 0.65 for a single setting. When calculating the best metrics for a subset of the settings that only used (1) static trees, (2) GEM routing metric, (3) quadratic groups or (4) circular groups, even better metrics could be found. As expected, the static trees behave better under adaptation than the algorithm using the GEM routing metric, even using the best coordination metric found. Also, circular groups can be better predicted than quadratic groups due to the "center problem" of the latter.

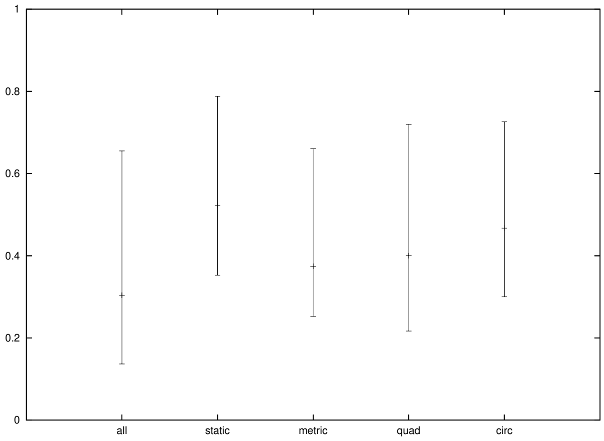


Figure 4: Correlation coefficients of best metric

5. CONCLUSION AND FUTURE WORK

Pure local adaptation of neighoring groups of nodes or even adaptation of single nodes in a sensor network can deteriorate network performance. Coordination of local adaptation decisions can improve the overall adaptation result although this implies additional overhead. The quality of the coordination is heavily dependent on the coordination metric. We have shown that such a metric can be found and that it can be improved further when restricting the domain.

As next steps, we will examine the metrics found in more detail to explain why each single component is part of a metric. Using this knowledge, it might be possible to develop a constructive method to build a metric.

When simulating the algorithms, no network or application characteristics have been changed. In the future, different traffic loads and node densities have to be included in the simulation space to cover the characteristics of the algorithm more completely.

6. REFERENCES

[1] P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. Rothermel. TinyCubus: A flexible and adaptive framework for sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 278–289, January 2005.

[2] O. Saukh, P. J. Marrón, A. Lachenmann, M. Gauger, D. Minder, and K. Rothermel. Generic routing metric and policies for WSNs. In *Proc. of the Third European Workshop on Wireless Sensor Networks*, pages 99–114, 2006.

[3] B. Titzer, D. Lee, and J. Palsberg. Avrora: Scalable sensor network simulation with precise timing. In *Proceedings of the Fourth International Conference on Information Processing in Sensor Networks*, pages 477–482, 2005.

Towards a Distributed Java VM in Sensor Networks using Scalable Source Routing

[Extended Abstract]

Bjoern Saballus
University of Karlsruhe
Am Fasanengarten 5
76131 Karlsruhe
saballus@ira.uka.de

Johannes Eickhold
University of Karlsruhe
Am Fasanengarten 5
76131 Karlsruhe
jeick@ira.uka.de

Thomas Fuhrmann
Technical University Munich
Boltzmannstrasse 3
85748 Garching
fuhrmann@net.in.tum.de

ABSTRACT

One of the major drawbacks of small embedded systems such as sensor nodes is the need to program in a low level programming language like C or assembler. The resulting code is often unportable, system specific and demands deep knowledge of the hardware details. This paper motivates the use of Java as an alternative programming language. We focus on the tiny AmbiComp Virtual Machine (ACVM) which we currently develop as the main part of a more general Java based development platform for interconnected sensor nodes. This VM is designed to run on different small embedded devices in a distributed network. It uses the novel scalable source routing (SSR) algorithm to distribute and share data and workload. SSR provides key based routing which enables distributed hash table (DHT) structures as a substrate for the VM to disseminate and access remote code and objects. This approach allows all VMs in the network to collaborate. The result looks like one large, distributed VM which supports a subset of the Java language. The ACVM substitutes functionality of an operating system which is missing on the target platform. As this development is work in progress, we outline the ideas behind this approach to provide first insights into the upcoming problems.

Keywords

distributed Java VM, sensor networks, embedded systems

1. INTRODUCTION

Today's life is hard to imagine without mobile phones, PDA's and other small devices. Additionally, everyday's life devices such as toasters and refrigerators are equipped with an increasing amount of computational power. As most of the usual tasks of these devices hardly exhaust this computational power, there is room to implement new, more sophisticated tasks. This idea becomes even more interesting if all these devices will be equipped with some kind of communication interface which allows to exchange data and trigger actions on remote nodes. The result is a distributed network of communicating small, embedded devices which belongs to the field of ubiquitous computing and ambient intelligence.

The BmBF research project *AmbiComp* aims at interconnecting all the above mentioned everyday life devices in an ad-hoc manner. Although the idea originated in the field of "digital" or "intelligent" homes, it extends to a much broader view of distributed computing: the devices will share data and distribute their workload among each other. The vision will be leveraged by the algorithms and protocols that we will develop in this project.

The paper is structured as follows: Section 2 gives an overview over other projects that have similar goals as the AmbiComp project. In section 3 we describe our novel AmbiComp VM, its design rationale and the tool chain from the Java file to the running program on the ACVM. Section 4 outlines how the ACVM addresses code and object migration. Finally, section 5 concludes with a summary of this paper.

2. RELATED WORK

Distributed Java VMs have been addressed in the literature before. Until now, most of these works originate from the fields of cluster computing and high performance computing. For example, Zhu et al. [9] propose JESSICA2, a distributed Java virtual machine (DJVM) which uses a global object space and transparent Java thread migration to provide a single system illusion. Haumacher et al. [5] describe how Java's remote method invocation (RMI) can be used to create and control transparent distributed threads in their JavaParty system.

In contrast to these heavy-weight approaches, we want to explore how they can be applied to embedded systems, for example, in sensor actor networks. To this end, our starting point is a small footprint Java VM. From its beginning, Java has been associated with the idea of cross-platform programming of embedded devices. Especially, the Java Micro Edition sets the focus on VMs that shall run on mobile devices like cell phones, PDA's etc.

Here, we give an overview of these VMs including their advantages and disadvantages concerning their use on even smaller embedded devices like sensor nodes. The criteria for this comparison are:

1. What are the specifics of the target platform hardware (processor, flash-rom, ram and clock speed) the VM is supposed to run on?
2. To which extent does the VM support the Java language specified by the Java Language Specification [3]?
3. Which set of class libraries is supported by the VM?

Beside the ACVM, the compared VMs are the *KVM* (Java ME: CLDC 1.1), *Squawk VM* (Sun SPOTs), *NanoVM* (Robots: Asuro) and the *ParticleVM* (Particles).

The KVM is a CLDC reference implementation implemented in ANSI C. It is the predecessor of the CLDC HotSpot Implementation [8] virtual machine and was specially developed for ARM processors. The KVM supports many different 16 and 32 bit micro processors which run at a minimum of 25MHz. Its memory demand is 160KB ROM and 32 KB RAM. The language complexity is specified by the CLDC 1.1 which also defines the supported API together with the MIDP.

The Squawk VM [7] was developed to run on the Sun SPOTs, developed by SUN Microsystems. It is nearly solely written in Java and does not need an operating system but comes with OS functions of its own. The target processor is the 32 Bit ARM-9 at a clock speed of 180MHz. Its memory demand is 149KB for the VM, 363KB for the VM suite, 158KB for the library suite in ROM. The Sun SPOTs have 512KB RAM from which about 20% are needed by the VM. The Java language is fully supported and the set of provided APIs are defined by the CLDC 1.1 extended by APIs for 802.15.4 radio communication and different low level hardware features.

The NanoVM [2][4] is a very limited VM which is distributed unter the GPL license and which runs e.g. on the Asuro roboters. The target platform is an ATmega8 or ATmega32 which both are 8 Bit microcontroller running at 8MHz clock frequency. The memory demand of this VM are 8KB ROM and less than 1KB RAM. It comes with a limited language support and only supports 15/31 Bit integer arithmetic, limited inheritance and a proprietary mechanism to support native code. There is no API present, only some native methods implemented in C.

The ParticleVM described in [6], is based on the NanoVM and is supported by the ParticleOS. Its memory demand is 45KB ROM and 0,5KB RAM. The target platform is a PIC18F6720 8 Bit microcontroller which runs at 20MHz. Its memory demand is 60KB ROM and 1,5KB RAM. The language support is the same as in case of the NanoVM, but the ParticleVM additionally supports interfaces. The API contains 20 classes with special functions for deployment and code migration via radio.

3. THE AMBICOMP VM

Like its small brother, the NanoVM, the new AmbiComp VM targets small microcontrollers such as the AVR 8-bit family, but provides more functionality than the NanoVM. The ACVM uses an external transcoding step, which transforms the original SUN Java opcodes and provides class loader functionality. This enables a very small footprint of the VM binary itself while supporting almost all SUN Java opcodes and a substantial part of the J2ME API. Moreover, the ACVM has been designed to support different target platforms with varying memory and processing capabilities. However, as a consequence, the entire bytecode that is needed by the ACVM must be passed through a transcoder instance before start up of a particular VM instance.

The tool chain for the ACVM starts with a regular *javac* compiler. It compiles a general albeit small Java program into one or more *.class*-Files. Then the transcoder takes this bytecode, transcodes it for the respective target platform, and links it, for example, statically to the plattform specific system libraries. Typically, only a small part of these libraries will be actually used. Thus the transcoder can eliminate a large portion of the library code.

The result of these transcoding process is a so-called *binary large object (BLOB)* file, which – in the case of static linking – contains the entire code that is needed to run the application on the target platform's ACVM. The BLOB is then transferred to the embedded device, for example, via Ethernet, USB, Bluetooth, or Zigbee depending on its respective networking capabilities. When the BLOB is fully loaded the ACVM starts to execute it.

Within the *AmbiComp* project, this tool chain will be fully integrated into the *Eclipse* software development environment, so that a programmer can directly deploy its code onto the target platform. In order for the transcoder to particularly transcode for respective target platform, the Eclipse user must select the desired target platform. The BLOB will then only be able to run on the according ACVM variant. Nevertheless, the Eclipse plugin will also have emulation capability so that the developers can check their program for compatibility with the respective target platform before deployment.

Unlike the NanoVM, the ACVM does not allow external users to provide native code directly. It comes however with several low level functions such as direct access to input and output pins of the microcontroller. Thereby, the ACVM – together with the system libraries – provides much of the functionality that is normally provided by the operating system: scheduling of threads in the VM, memory management and memory protection including garbage collection, and all kinds of IO functionality including the communication stacks. Hence, the ACVM does not need any further operating system.

4. DISTRIBUTED OPERATION OF ACVM

So far, we have described the ACVM as a stand-alone system. This impression is not quite true because the ACVM can access external objects. To this end, it makes use of the scalable source routing (SSR) protocol, which provides key based routing (KBR) for low-resource embedded nodes. A detailed description of SSR can be found in [1].

The ACVM uses SSR to route object access requests to remote objects. It can do so because KBR can address objects rather than nodes. Thus, it is easy to retrieve an object by its globally unique identifier. Unlike other distributed Java VMs this approach does not need any centralized server.

We will illustrate the use of this beneficial property with the following example: Consider a scenario where a temperature sensor shall send its measured values to a display device. Similar to a multi-threaded local system we use a sensor thread (cf. listing 1) and a display thread (cf. listing 2). Both are coupled via a singleton which stores the data that is published by the sensor (cf. listing 3). The display could regularly pull the published values from the data store, or it registers a listener with the data store in order to have the values pushed. To this end, the display implements the respective listener interface (cf. listing 4).

```
package push.sensors;

import hardware.TemperatureSource;
import push.TemperatureDataStore;

public class TemperatureSensor {

  private TemperatureDataStore dataStore;
  private TemperatureSource sensorHardware;
  private String name;

  public TemperatureSensor(String name) {
    this.name = name;
    this.dataStore = TemperatureDataStore.
        getInstance();
    this.sensorHardware = new TemperatureSource();
  }

  public void run() throws InterruptedException {
    while (true) {
      Thread.sleep(500);
      System.out.println(”s” + name + ” pushed”);
      this.dataStore.store(
          this.sensorHardware.getValue()
      );
    }
  }
}
```

Listing 1: The sensor

```
package push.actuators;

import hardware.Display;
import push.IDataListener;
import push.TemperatureDataStore;

public class TemperatureDisplay implements
    IDataListener {

  private Display actorHardware;
  private String name;

  public TemperatureDisplay(String name) {
    this.name = name;
    // create access to hardware
    this.actorHardware = new Display();
    // register ourself as listener to the data store
    TemperatureDataStore.getInstance().
        addDataListener(this);
  }

  public void notify(float value) {
    actorHardware.show(this.name + ”: ”+ value + ”C”);
  }
}
```

Listing 2: The display

```
package push;

import java.util.ArrayList;
import java.util.List;

public class TemperatureDataStore {

  // we are a singelton
  private static TemperatureDataStore instance = null;
  // the actual data store
  private List<Float> temperatureValues;
  // listeners to be notified about new temperature
  private List<IDataListener> dataListeners;

  private TemperatureDataStore() {
    this.temperatureValues = new ArrayList<Float>();
    this.dataListeners =
        new ArrayList<IDataListener>();
  }

  /**
   * This class implements the singelton pattern.
   * @return The instance of this singelton.
   */
  public static TemperatureDataStore getInstance() {
    if (instance == null) {
      instance = new TemperatureDataStore();
    }
    return instance;
  }

  /**
   * Publish a value to the store.
   * @param value The value to be stored.
   */
  synchronized public void store(float value) {
    this.temperatureValues.add(value);
    notify(value); // notify listeners
  }

  /**
   * Notify all registered IDataListeners
   * about a new value.
   * @param value The new value.
   */
  private void notify(float value) {
    for (IDataListener listener : dataListeners) {
      listener.notify(value);
    }
  }

  /**
   * Adds a new IDataListener to the list of listeners.
   * @param listener The listener to be added.
   */
  public void addDataListener(IDataListener listener) {
    this.dataListeners.add(listener);
  }
}
```

Listing 3: The data store

```
package push;

public interface IDataListener {
  public void notify(float  value);
}
```

Listing 4: The listener

Since the data store is a singleton, only one such instance will be available (in the local domain). Accordingly, the respective class member will be remote for the nodes except for the one that holds this member. SSR will find this node based on its globally unique ID. Such an ID is obtained by combining a BLOB-local ID with the hash of the BLOB that defines the member.

At the current stage of the project, this mechanism is only about to be available in practice. Moreover, not all design decissions have been made up to now. Let us briefly sketch some of the respective problems.

For example, the concept of administrative domains has only been phrased roughly. It addresses the fact that depending on the application, singleton objects should not be global but somehow local. In the described scenario, the temperature sensor and the display should belong to the same household or the same machine. Thus, the ACVM needs to be equipped with a means to let the application developers decided which singleton domain they want to have for a particular aspect of their application. We envisage to support this via an API, but it is still unclear which granularity of control this API should provide.

Another open issue that has not been decided yet is the question of how to treat synchronized remote objects. On a local machine synchronized objects are rather simple because deadlocks can be considered a bug in the application. In a distributed environment deadlocks can also be caused by failing nodes or lost communication links. At the time being, it is unclear to what extend the application programmer needs to be confronted with these issues.

Furthermore, we are still considering various aspects of object and code migration. For example, BLOBs cannot only be linked statically, but also dynamically so that other BLOBs are loaded on demand. This can again be done via SSR's KBR semantic, for example, with exploiting caching of BLOBs in nodes that have enough memory. This will enable each node to load and execute software on demand whenever it is needed. Moreover, this feature also allows to trade several remote object accesses (ROA) for potentially one remote method invokation (RMI). Albeit it is rather easy to implement RMI on top of SSR, it is yet unclear how a simple heuristic can help the ACVM decide between ROA and RMI.

5. SUMMARY

In this paper we have described ongoing work in the AmbiComp project: We have briefly presented the AmbiComp Virtual Machine (ACVM) that executes BLOBs that an external transcoder creates from Java *.class*-files; we have also described how the ACVM together with the scalable source routing (SSR) protocol manages remote object access; and we have illustrated the use of this idea with a simple temperatur sensor and display example.

Yet, since this is still very early work, we have not addressed all the open issues in this field. In essence, the ACVM can provide a powerful means to hide the distributedness of a sensor-actor-system. But in practice the application developers need a way to craft their distributed system. To this end they need to decide, for example, on the administrative domain in which an application may exchange data, or on the semantics of synchronization in case of failing nodes. We hope therefore that this paper stimulates the discussion about these questions.

6. REFERENCES

[1] T. Fuhrmann. Scalable routing for networked sensors and actuators. In *Proceedings of the Second Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Sept. 2005.

[2] T. Fuhrmann and T. Harbaum. A platform for lab exercises in sensor networks. Technical report, Z?rich, Switzerland, Mar. 23–24 2005.

[3] J. Gosling, B. Joy, G. Steele, and G. Bracha. *Java Language Specification.* Addison-Wesley Professional, third edition, July 2005.

[4] T. Harbaum. The NanoVM - Java for the AVR, 2005. `http://www.harbaum.org/till/nanovm`.

[5] B. Haumacher, T. Moschny, J. Reuter, and W. F. Tichy. Transparent distributed threads for java. page 147, Nov. 12 2003.

[6] T. Riedel, A. Arnold, and C. Decker. An OO Approch to Sensor Programming. In *EWSN 2007 - Adjunct poster proceedings*, number PDS-2007-001, pages 39–40, Delft, The Netherlands, Jan. 29-31 2007.

[7] N. Shaylor, D. N. Simon, and W. R. Bush. A java virtual machine architecture for very small devices. In *ACM SIGPLAN conference on Language, compiler, and tool for embedded systems (LCTES)*, pages 34–41, New York, NY, USA, 2003. ACM Press.

[8] Sun microsystems. *CLDC HotSpot Implementation Virtual Machine*, Feb. 2005. `http://java.sun.com/j2me/docs/pdf/CLDC-HI_whitepaper-February_2005.pdf`, accessed on 2007-05-21.

[9] W. Zhu, C.-L. Wang, and F. C. M. Lau. Jessica2: A distributed java virtual machine with transparent thread migration support. In *IEEE Fourth International Conference on Cluster Computing*, Chicago, USA, September 2002.

Misbehaviour Detection for Wireless Sensor Networks - Necessary or Not?

Sven Schaust
FG Simulation and Modellierung
Institute of Systems Engineering
Welfengarten 1, 30167 Hannover, Germany
svs@sim.uni-hannover.de

Helena Szczerbicka
FG Simulation und Modellierung
Institute of Systems Engineering
Welfengarten 1, 30167 Hannover, Germany
hsz@sim.uni-hannover.de

Keywords

misbehaviour detection, artificial immune system, sensor networks

ABSTRACT

As radio communication modules combined with adequate hardware based cryptography are becoming available at low cost, sensor network security has reached a new level. However it is still possible to gain access to networks, even if they require authorisation and authentication, as single nodes cannot be assumed to be tamper proof. Therefore the question remains what kind of misbehaviour is possible within a secured network and how it can be detected. A possible solution to detect misbehaviour is the usage of an artificial immune system (AIS). The advantage of an AIS in contrast to protocol based security improvements is the generality of the approach, as it tries to identify anomalies which are not known to exist within the regular range of network behaviour. In this paper we present some results obtained from our simulation experiments looking at the possibilities of artificial immune system based misbehaviour detection.

1. INTRODUCTION

Detection of misbehaviour in wireless sensor networks (WSN) is an important research area as the number of companies using such networks for logistics, production cycle maintenance, indoor security or area surveillance is rising. Companies cannot afford to experience losses due to malfunctioning systems. Sensor networks have to be able to detect any malfunction not only as fast as possible but also with high accuracy, thus having only a low level ratio of false alarms and a high level ratio of true alarms.

When talking about securing ad-hoc sensor networks, the crucial point is the hardware used to form such networks. Typically a single node is a battery powered device with limited computational abilities and therefore using strong software cryptography to secure communication channels is almost impossible. Thus a software based security system has to be frugal in terms of memory space usage and computation time consumed.

Due to the latest developments of wireless communication protocols intended to be used with sensor networks (for example the work of the ZigBee® Alliance [18]) and the development of radio modules with support of hardware based AES-128 cryptography, unauthorised access to sensor networks is becoming more and more difficult. However commonly available sensor nodes are not tamper proof, leaving possibilities to access secured networks. Even if the access to such a network is difficult, a sensor network still has to deal with malicious nodes as network protocols alone cannot guarantee to avoide security or node behaviour problems.

In this paper we present some results obtained from our simulation experiments looking at the possibilities of artificial immune system based misbehaviour detection. We examine the AIS's detection ability when exposed to Poisson and constant bit rate (CBR) traffic. The paper is organised as follows. Section 2 describes briefly the concept of an artificial immune system. Section 3 explains our experimental setup followed by Section 4 presenting the results. In section 5 related work and in 6 the conclusion are presented.

2. ARTIFICIAL IMMUNE SYSTEM

Artificial immune systems are derived from the human immune system and therefore several terms and descriptions have been adopted from the medical perspective on immune systems.

A *gene* is defined by a characteristic based on the volume of traffic, the assumed protocol behaviour or both (for example the number of complete MAC handshakes[1] during a specific time period).

An *antigen* is an observation within a time window for a set of genes which can either be interpreted as an *self-antigen* or a *non-self-antigen*.

A *detector* is based on several immune system methods which are related to the detection of alien cells. In AIS a detector is defined as a bit sequence which is produced by a negative-selection algorithm [7] which only matches against non-self antigens.

A simple artificial immune system can be divided into a learning phase and a detection phase [1]. During the learning phase detectors are produced (using the available self information) which will be used later in the detection phase to discover misbehaviour indicators. An AIS is running locally on every node, using the observed traffic to create the self set, the detectors and the antigens necessary to detect misbehaviour at its neighbours. There are several extensions which allow adaptive learning and maturation of detectors, thus avoiding the necessity of an intensive self-set only learning phase. See [2], [3] and [4] for more information on the different mechanisms and the immune system in general.

3. EXPERIMENTAL SETUP

The purpose of our experiments was to exam the detection abilities of our AIS when exposed to different packet injec-

---

[1] A MAC handshake is defined by the sequence of RTS, CTS, DATA and ACK packets.

tion models. We choose the Poisson distribution model (as example for event triggered packet injection) as sensor networks are expected to measure Poisson distributed events. The CBR model was chosen as comparison model assuming that CBR traffic is unlikely but possible in a sensor network. Similar to [1] a bit string representation was chosen for self, non-self and detectors. We captured self and non-self packet traffic and tested whether the AIS was able to detect misbehaviour. We captured for every transmitted packet the IP header type (UDP, 802.11 or DSR), the MAC frame type (RTS, CTS, DATA or ACK), the current simulation time, the node address, the next hop address, the global packet source, the global packet destination and the packet size. These values where used to compute the necessary genes. In each scenario we ensured that the average hop count distance between two nodes was about 8 hops. See section 3.1 and section 3.2 for simulation and AIS details.

We choose DSR [5] as routing protocol. Misbehaviour was implemented at 236 of our 1718 nodes using a simple packet dropping misbehaviour with a 10, 30 and 50% probability (Sink and source nodes were excluded). Although the number seems relatively high only one to three malicious nodes were actually part of a route as the nodes were distributed randomly.

Each scenario was simulated using Glomosim 2.03 (see [6]) 20 times with different seeds for the Glomosim random number generator.

3.1 Simulation details

- **Negative selection algorithm:** random generate and test. Implemented in C++, compiled with GNU g++ v4.0 with -O3 option.
- **Input parameters:** 1. r-contiguous matching rule with $r = 10$. 2. Encoding: 5 genes each 10 bits long = 50 bits. 3. Number of detectors $\{500, 1000, 2000\}$. 4. Misbehaviour level $\{10\%, 30\%, 50\%\}$ 5. Window size 500 seconds; 28 complete windows over 4-hours simulation time.
- **CBR Injection rate:** 1 packet/second. 14400 packets per connection were injected. Packet size was 512 bytes.
- **Poisson Injection rate:** $\lambda = 1.0$, meanArrivalExpectation = 1 packet/second. Packet size was 512 bytes.
- **Performance measures:** detection rate, data traffic rate at nodes; Values were produced per simulation run and compared as arithmetic average over all simulations for each misbehaviour probability.
- **MAC protocol**: IEEE 802.11b DCF.
- **Routing protocol:** DSR.
- **Other parameters:** (i) Propagation path-loss model: two ray (ii) Channel frequency: 2.4 GHz (iii) Topography: Line-of-sight (iv) Radio type: Accnoise (v) Network protocol: IPv4 (vi) Connection type: UDP.

3.2 AIS details

When defining a misbehaviour detection system several observable facts have to be specified, otherwise no detection is possible. For artificial immune systems these facts are defined by *genes*. For our experiments we decided to observe two layers of the OSI Stack namely the MAC and Routing layer using the following set of genes:

MAC layer:

#1 Ratio of complete MAC layer handshakes between nodes s_i and s_{i+1} and RTS packets sent by s_i to s_{i+1}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is averaged over a time period. A complete handshake is defined as a completed sequence of RTS, CTS, DATA, ACK packets between s_i and s_{i+1}.

#2 Ratio of data packets sent from s_i to s_{i+1} and then subsequently forwarded to s_{i+2}. If there is no traffic between two nodes this ratio is set to ∞ (a large number). This ratio is computed by s_i in promiscuous mode. This ratio is also averaged over a time period. This gene was adapted from the watchdog idea in [14].

#3 Time delay that a data packet spends at s_{i+1} before being forwarded to s_{i+2}. The time delay is observed by s_i in promiscuous mode. If there is no traffic between two nodes the time delay is set to zero. This measure is averaged over a time period. This gene is a quantitative extension of the previous gene.

Routing Layer:

#4 The same ratio as in #2 but computed separately for RERR routing packets.

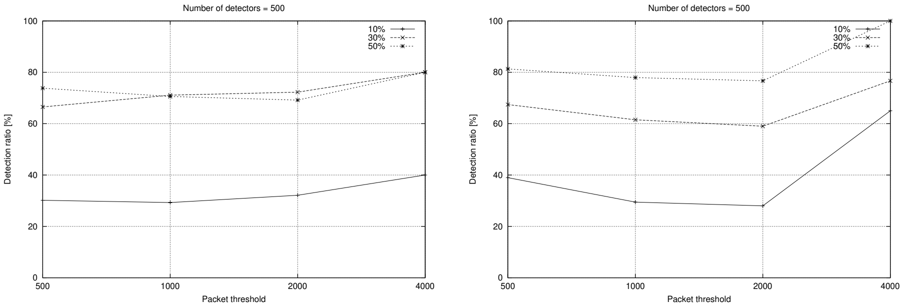
#5 The same delay as in #3 but computed separately for RERR routing packets.

Each gene was encoded using an interval representation of size 10 which was adopted from [8]. The correspondending interval was marked by a single 1 within the 10 bit sequence. Antigens were produced by the concatenation of all five genes and always checked against the complete detector set.

4. RESULTS

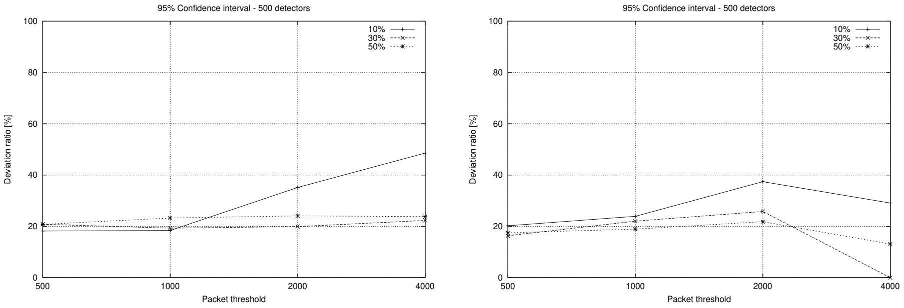
The task of detecting misbehaviour requires comparison of the computed detectors with observed non-self antigens. In our experiments a 500 second time window was used to sample overheard node traffic and to generate one antigen. Therefore the resulting number of time windows for 4 hours simulated times was 28 per node. In order to avoid outliers in our analysis we defined a detection threshold of 14 time windows to mark a node as misbehaving.

The evaluation of the detection rate requires that the number of packets that a node forwards is over a certain threshold. If a node lacks packets to forward in the learning phase, the AIS's ability to learn is limited. If it lacks packets to forward during the detection phase and at the same time wants to execute misbehaviour, the impact of misbehaviour is weakened. As a result we performed our evaluation using



(a) (b)

Figure 1: The figure shows the detection rate for different misbehaviour rates vs. forwarded packet threshold for 500 detectors (a) CBR packet injection model. (b) Poisson packet injection model.



(a) (b)

Figure 2: The figure shows the deviation ratio for different misbehaviour rates vs. forwarded packet threshold for 500 detectors. (a) Deviation for CBR packet injection model. (b) Deviation for Poisson packet injection model.

different threshold values for packet forwarding (minValue = 500, 1000, 2000 and 4000) and considered only those nodes which were above the given thresholds.

Definition: The *detection rate* is defined as $d_r = 100 * \frac{n_d}{n_m + n_d}$, were n_m = the number of misbehaving nodes to detect, and n_d = the number of detected nodes.

Definition: The *deviation ratio* is defined as $dv_r = 100 * \frac{d_v}{d_r}$, were d_v = the deviation value of the 95% confidence interval, and d_r = the detection rate for a specific misbehaviour.

The graphs in figure 1 show the average detection results[2] for CBR and Poisson packet injection using 500 detectors. While the detection rate is about 10% higher for the Poisson packet injection with a low packet threshold of 500 packets, the ratio for the CBR model seems to be better for higher thresholds. In order to verify the results for the CBR and Poisson model we calculated a 95% confidence interval for all three misbehaviour probabilities. The deviation ratios for each misbehaviour with 500 detectors are shown in figure 2. The CBR and the Poisson model show similar deviation ratio values, suggesting that both models have a similar influence on the detection rate.

5. RELATED WORK

Hofmeyr and Forrest presented in 1999 a paper about the usage of an immune system inspired misbehaviour detection system. Their artificial immune system was designed to work on a wired network observing TCP/IP connections. The pattern matching was based on a r-contiguous bits matching with $r = 12$ bits. The antigen and detector length was 49 bits. [1]

In [8] and [9] Sarafijanovic and Boudec presented an artificial immune system based misbehaviour detection system designed for wireless ad-hoc networks. The approach was tested using Glomosim to simulate a network of 40 mobile nodes (1 m/s) of which 5 to 20 nodes were misbehaving. They defined four genes to be used to capture local behaviour at the OSI network layer. The detection rate of the presented system was about 55%. They also used a reputation system with a *danger signal* which allows nodes to

---

[2] We used the arithmetic average of all simulation runs to calculate the detection rate.

inform neighbours on the routing path about misbehaviour. This interaction was adopted from the results by Aickelin et. al.[3].

Aickelin et. al. have been working on artificial immune systems since 2003 in an interdisciplinary project called *danger theory.* In [3] and [4] they introduced work showing links between intrusion detection systems and artificial immune systems. They also introduced a danger signal approach allowing nodes to judge the misbehaviour information and presented work on adaptive learning mechanisms.

In [12] Drozda et. al. showed that artificial immune systems can be applied to sensor networks having only low computational costs. The detection rate of the introduced AIS is higher than the one presented by Sarafijanovic and Boudec, as the authors use a static ad-hoc network of 1718 nodes with 10 CBR connections instead of a mobile network.

6. CONCLUSION

Finding misbehaviour in deployed sensor networks is an important factor when routing and data reliabiltiy are required. In the presented experiments we studied the impact of two different traffic patterns. The results obtained in our simulation experiments show a similar detection rate for both traffic models. Our AIS accomplished a detection rate of 70% to 80% independent from the traffic model. We conclude that artificial immune systems are a preferable mechanism for misbehaviour detection in sensor networks as traffic models or attack patterns are not known in advance. AIS offer a universal approach which is only limited by the quality of the chosen genes and the number of observed OSI layers. We are currently working on a simulation experiment using 50 fixed but randomly chosen connections with a different variety of misbehaving nodes. We want to consider different packet injection parameters in order to test the AIS with other typical sensor network traffic patterns (for example Poisson distributions with different packet arrival expectations or a massive packet injection by a small group of nodes followed by a long quiet period for the whole network) and different, more complex attack patterns. The intention of these experiments is to verify which *genes* should be part of an AIS in general.

Additionally we are working on a TinyOS [19] implementation of our AIS and the proposed AIS by Sarafijanovic and Boudec [8] using Crossbows [17] mica2 sensor nodes. We want to verify our simulation results and show that AIS are indeed a preferable method for misbehaviour detection using commonly available sensor network hardware. Our network will consist of 40 mica2 nodes using realistic data collection scenarios.

7. REFERENCES

[1] S. Hofmeyr, S. Forrest. Immunity by Design: An Artificial Immune System. *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, 1999.

[2] Charles A. Janeway Jr. How the immune system works to protect the host from infection: a personal view. *Proc. Natl. Acad. Sci. U S A.*, 2001 Jun 19;98(13):7461-8.

[3] U. Aickelin, P. Bentley, S. Cayzer, J. Kim, J. McLeod. Danger theory: The link between ais and ids. *Proc. International Conference on Artificial Immune Systems (ICARIS)*, 2003.

[4] U. Aickelin, J. Greensmith, J. Twycross. Immune System Approaches to Intrusion Detection - A Review. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS)*, 2004.

[5] D. Johnson, D. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. *Mobile Computing*, Tomasz Imielinski and Hank Korth, Eds. Chapter 5, pp. 153-181, Kluwer Academic Publishers, 1996.

[6] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, M. Gerla. GloMoSim: A Scalable Network Simulation Environment. UCLA Computer Science Department Technical Report 990027, May 1999.

[7] J. Kim, P.J. Bentley. Evaluating Negative Selection in an Artificial Immune System for Network Intrusion Detection, *Proc. Genetic and Evolutionary Computation Conference (GECCO)*, 2001.

[8] S. Sarafijanović, J.-Y. Le Boudec. An Artificial Immune System for Misbehavior Detection in Mobile Ad-Hoc Networks with Virtual Thymus, Clustering, Danger Signal and Memory Detectors. *Proc. the 3rd International Conference on Artificial Immune Systems (ICARIS)*, 2004.

[9] J.-Y. Le Boudec, S. Sarafijanović. An Artificial Immune System Approach to Misbehavior Detection in Mobile Ad-Hoc Networks. *Proc. Bio-ADIT'04*, 2004.

[10] J.P.G. Sterbenz, R. Krishnan, R. Rosales Hain, A.W. Jackson, D. Levin, R. Ramanathan, J. Zao. Survivable mobile wireless networks: issues, challenges, and research directions. *Proc. ACM workshop on Wireless security*, 2002.

[11] Y. Zhang, W. Lee, Y.A. Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *Wireless Networks*, vol. 9, no. 5, pp. 545–556, 2003.

[12] M. Drozda, S. Schaust, H. Szczerbicka. Is AIS Based Misbehavior Detection Suitable for Wireless Sensor Networks? *Proc. IEEE Wireless Communications and Networking Conference (WCNC)*, 2007.

[13] H. Karl, A. Willig. *Protocols and Architectures for Wireless Sensor Networks.* John Wiley & Sons, 2005.

[14] S. Marti, T. J. Giuli, K. Lai, M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Proc. the 6th annual international conference on Mobile Computing and Networking (MobiCom)*, 2000.

[15] J. Balthrop, S. Forrest, M. Glickman. Revisiting lisys: Parameters and normal behavior. *Proc. Congress on Evolutionary Computing*, 2002.

[16] D. Dasgupta, F. Gonzalez. An immunity-based technique to characterize intrusions in computer networks. *IEEE Trans. Evolutionary Computation*, vol. 6, no. 3, pp. 281–291, 2002.

[17] Crossbow Technology Inc. `www.xbow.com`

[18] ZigBee Alliance®`www.zigbee.org`

[19] TinyOS `www.tinyos.net`

Performance of Additive Homomorphic EC-ElGamal Encryption for TinyPEDS

Osman Ugus
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
ugus@netlab.nec.de

Alban Hessler
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
hessler@netlab.nec.de

Dirk Westhoff
NEC Europe Ltd.
Kurfuersten-Anlage 36
69115 Heidelberg, Germany
westhoff@netlab.nec.de

ABSTRACT

Data aggregation is a popular approach employed in wireless sensor networks (WSNs) to minimize data transmission and storage. With aggregation techniques, the monitored data is expressed in a condensed form: Therefore, instead of storing all data sensed by several nodes, the network stores a condensed value only such as the sum of these values. However, data aggregation becomes problematic when the data to be aggregated is encrypted. As a solution, we apply an additive homomorphic encryption scheme, namely the elliptic curve ElGamal (EC-ElGamal) cryptosystem, and present the performance results of our implementation for the prominent sensor platform MicaZ mote.

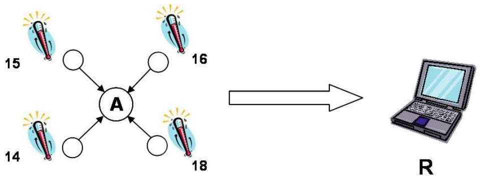
1. INTRODUCTION

Regarding the frequency of transmission wireless sensor networks (WSNs) may be divided into two subgroups, namely asynchronous and synchronous [7]. In synchronous WSNs the monitored data transmitted to a reader device is real-time responsive. In asynchronous WSNs, however, the monitored data is transmitted to a reader device only seldomly. Therefore, asynchronous WSNs need to store the data in the network in a distributed manner. However, the implementation of distributed data storage for WSNs is very challenging.

Firstly, due to the limited storage capacity of the nodes, the storage capacity of the whole WSN is restricted. Thus, it is necessary to reduce the amount of the data being processed, e.g. stored or transmitted, in the network without loosing relevant information. Secondly, the nodes have limited power capacity. Distributed data storage requires transmission between sensor nodes. Since the transmission affects the power consumption, techniques for minimizing it are mandatory. Finally, the nodes are in general equipped with non-tamper-resistant hardware. Since WSNs are usually employed in a public environment, data must be protected and concealed.

One popular method employed for minimizing the data transmission and storage is *in-network data aggregation*. This means that the monitored data is expressed in a condensed form such that instead of storing all the data sensed by several nodes, the network stores only condensed values such as the sum or the average of these values. In order to secure the data stored in the network, data encryption techniques are employed. However, in-network data aggregation becomes challenging when the data is encrypted. In such case the encryption scheme needs to allow homomorphic addition of ciphertexts. As solution for these problems, *tiny persistent encrypted data storage* (TinyPEDS) was proposed, see [7].

In this work we introduce the *additive homomorphic elliptic curve ElGamal* (EC-ElGamal) encryption scheme, which is employed for securing distributed storage and transmission of aggregated data in TinyPEDS. Furthermore, the performance results of the implementation on MicaZ mote are presented.



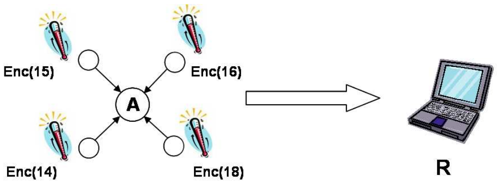
A calculates *Sum = (15+16+18+14)*
R calculates *Sum / 4*

Figure 1: Data aggregation in WSNs

2. CONCEALED DATA AGGREGATION

As resource consumption of the nodes is a critical factor for the overall lifetime of a wireless sensor network, it is necessary to employ techniques to reduce it. The in-network data aggregation works toward this goal by reducing the amount of the data being stored and transmitted, while still providing an appropriate degree of information to the reader device. Figure 1 depicts a simple example for the in-network data aggregation process where the sensed values are not encrypted.

However, as WSNs are usually employed in a public environment, the data has to be encrypted in order to limit damage caused by possible attacks. In this case, the aggregator node A should calculate the $Sum = [Enc(15) + Enc(16) + Enc(18) + Enc(14)]$ of the encrypted values. This scenario is depicted in Figure 2.



A calculates *Sum = [Enc(15)+Enc(16)+Enc(18)+Enc(14)]*
R calculates *Dec(Sum) / 4*

Figure 2: Concealed data aggregation in WSNs

In general, the decrypted sum of ciphertexts $Dec[Enc(15) + Enc(16) + Enc(18) + Enc(14)]$ will not be equal to the sum of the plaintexts $(15+16+18+14)$. Thus, the encryption scheme employed needs to support the property such that the following equation holds:

$$Enc(a_1 + a_2 + ... + a_n) = Enc(a_1) \diamond Enc(a_2) \diamond ... \diamond Enc(a_n),$$

where $Enc(a)$ denotes the encryption of a message a and $\diamond$ represents an addition performed on ciphertexts from a public encryption scheme. An encryption scheme with this property is called *additive homomorphic*. TinyPEDS proposes to employ both a symmetric and a public key encryption scheme. However, in this work we introduce only the asymmetric additive homomorphic encryption EC-ElGamal. An example for a symmetric additive homomorphic encryption scheme is [4].

3. ELLIPTIC CURVE ELGAMAL ENCRYPTION SCHEME

The original ElGamal encryption scheme, see [6], is *multiplicative homomorphic*. In order to get the desired additive homomorphic property, it can be implemented in elliptic curves. The methods for EC-ElGamal encryption and decryption is shown in Algorithm 1 ([13]).

The function $map()$ is a deterministic mapping function used to map values $m_i \in GF(p)$ into plaintext curve points $M_i \in E$ such that

$$map(m_1 + ... + m_n) = \underbrace{map(m_1)}_{M_1} + ... + \underbrace{map(m_n)}_{M_n}$$

holds. The function $map()$ is necessary, because the addition over an Elliptic Curve is only possible with points on that curve, thus, integers have to be mapped to corresponding points. For this purpose, each integer m is mapped to a curve point M, which is the m-multiple of the generator point G, i.e. $M = mG$. The reverse mapping function $rmap()$ then extracts m from a given point mG. The mapping function

$$map : m \rightarrow mG \text{ with } m \in GF(p)$$

fulfills the required homomorphic property, because

$$\begin{aligned} M_1 + M_2 + ... + M_n &= map(m_1 + m_2 + ... + m_n) \\ &= (m_1 + m_2 + ... + m_n)G \\ &= m_1G + m_2G + ... + m_nG \end{aligned}$$

holds, where $m_1, m_2, ..., m_n \in GF(p)$.

The mapping function is not security relevant, i.e. it neither increases nor decreases the security of the EC-ElGamal encryption scheme. Note that the reverse mapping function is the same as solving the *discrete logarithm problem* over an elliptic curve and, therefore, a weakness of this scheme. However, since the reverse mapping function is only performed on the powerful reader device and the maximum length of the final aggregation is assumed to be small for realistic WSNs, see [13], the m can be obtained by performing a brute force attack on $M = mG$ on the reader device.

Algorithm 1 Elliptic curve ElGamal encryption scheme

```
Private key: $x \in GF(p)$
Public key: $E, p, G, Y$, whereby $Y = xG$ and elliptic curve $E$ over GF(p) with $G, Y \in E$
Encryption: For a given plaintext $m \in [0, p-1]$ and random $k \in [1, n-1]$, where $n$ is the order of $E$.
    $M = map(m)$
    ciphertext $C = enc(m) = (R, S) = (kG, M + kY)$
Decryption:
    $M = dec(C) = dec(R, S) = -xR + S$
    $m = rmap(M)$
```

4. IMPLEMENTATION

The software architecture of the EC-ElGamal cryptosystem depicted in Figure 3 consists of three abstraction levels. The lowest level contains finite field arithmetic operations such as *modular addition*, *modular subtraction*, and *modular multiplication*. Since the operations on higher levels are based on the operations on this level, the finite field level is the most critical level for the performance. Elliptic curve arithmetic operations such as *point addition*, *point doubling*, and *scalar point multiplication* are grouped on the second level. Finally, on the application level, by using the operations from lower levels any elliptic curve cryptosystem such as EC-ElGamal may be implemented.

The performance of a cryptosystem is influenced by the employed algorithms. Therefore, it is important to choose the best combination carefully. In the following the algorithms employed in each abstraction level are introduced.

4.1 Design decisions at finite field level

4.1.1 *Selecting the underlying field*

In [3] Branovic *et al.* studied the performance and memory requirements of several elliptic curve algorithms over the prime *GF(p)* and binary field *GF(2^n)*. They found that the number of memory accesses and the code sizes of elliptic curve algorithms for binary fields is higher than those for prime fields. Moreover, the binary field arithmetic, particularly multiplication, is not sufficiently supported by usual microprocessors. Thus, the use of binary field would lead to lower performance [9].

On the other hand, the main disadvantage of the prime field arithmetic is that the inversion over *GF(p)* is computationally much more expensive than the inversion over a binary field. However, by using different coordinate systems, the number of inversions required by finite field and elliptic curve arithmetic can be reduced to only one, which is needed for converting the final results back to affine coordinates. Since the implemented cryptosystem in this work allows those final conversions on the external reader device, the performance drawback of prime fields stemming from expensive inversions can be ignored.

In contrast to the inversion, *modular reduction* can be performed over prime fields faster than over binary fields when special *Pseudo-Mersenne primes* [18] are used. Therefore, the arithmetic operations in this work were implemented over *GF(p)*.

4.1.2 *Multi-precision multiplication*

The efficiency of the multi-precision multiplication dominates the overall performance. In [9] Gura *et al.* show that on resource constrained devices such as MicaZ mote, 85% of the execution time is spent on the multi-precision multiplication for a typical point multiplication. That means that the performance of the multi-precision multiplication is critical for the entire cryptosystem and therefore, it is especially necessary to concentrate on its optimization possibilities.

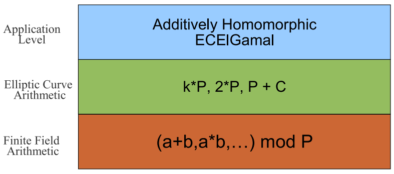


Figure 3: Elliptic curve ElGamal (EC-ElGamal) design architecture

Additionally to the well-known *Schoolbook* and *Karatsuba* multiplication, see [11], [17] analyzed the *Comba* multiplication, see [8], and the *Hybrid* multiplication from [9]. Our analysis showed that the Hybrid method is the most promising one, because it combines the advantages of the Schoolbook and Comba method. Therefore, it requires a small amount of registers and memory accesses.

4.1.3 Modular reduction

After each finite field operation the modular reduction is necessary whenever the result is bigger than the modulus. Therefore, the performance of the modular reduction is as important as multiprecision multiplication for the overall performance of the cryptosystem.

In [17] we analyzed three methods to perform modular reduction: *Montgomery reduction*, *Barrett reduction*, and the *pseudo-Mersenne prime reduction* [11]. Our analysis showed that the pseudo-Mersenne prime reduction is superior to the other methods.

4.2 Design decisions at elliptic curve level

4.2.1 Selecting the coordinate system

Elliptic curve points may be represented in several coordinate systems. For each of such systems, the performance and the memory requirements of the elliptic curve operations are different. Therefore, a good choice for the coordinate systems is an important factor for the successful implementation of the elliptic curve cryptosystems on resource limited devices. In [17] we compared six popular coordinate systems in terms of performance and memory requirements, namely *Affine*, *Projective*, *Jacobian*, *Chudnovsky-Jacobian*, *modified Jacobian*, and *mixed* coordinate systems.

We showed that AJJ coordinates are superior to the other mixed coordinate systems. Thereby, A, J and M denotes the affine, Jacobian and modified Jacobian coordinates, respectively. Note that AJJ denotes the point addition, whereby the result is in Jacobian coordinates, while the input points are in affine and Jacobian coordinates. Similarly, MJ denotes the point doubling with the input point is in modified Jacobian and the result is in Jacobian coordinates. The performance of the point doublings in MJ coordinates is the highest, whereas the point doublings in those coordinates require more temporary storage than the point doublings in JJ coordinates. However, since the results of the point additions in the application level are used as input for the point doublings during the scalar point multiplication, the use of MJ coordinates needs one more additional conversion from Jacobian to modified Jacobian coordinates. This means the computational efficiency of point doublings in MJ coordinates is equal to the point doublings in JJ coordinates. Thus, AJJ and JJ coordinates were chosen in the implementation of the point addition and point doubling, respectively.

4.2.2 Scalar point multiplication

The main operation in elliptic curve cryptosystems is the scalar point multiplication. Since the scalar point multiplication is a time critical operation, many efficient implementations have been proposed. We evaluated the *Left-to-Right* and *Right-to-Left* binary methods, see [11], as a core algorithm for scalar multiplication.

The examination in [17] showed that he Left-to-Right method is superior due to lower storage requirements and the fact that the Right-to-Left method does not allow AJJ point additions.

Table 1: Comparison of the scalar point multiplication implementations

160-bit (sec160r1)	#Precom. points	Ex. time [s]
This work (L2R)	0	1.23
This work (2MOF)	0	1.03
[9]	0	0.81
[18]	0	1.35
TinyECC-0.2	0	1.78
This work	1	0.69
This work	2	0.57
[18]	15	1.24

Several techniques may be employed to further speed up the scalar point multiplication on elliptic curves. One of them is the *Interleave method* [12], which aims at reducing the required number of point doublings. The Interleave method was originally proposed for the multi-scalar multiplications of the form $(k_1 \cdot P_1 + k_2 \cdot P_2 + ... + k_n \cdot P_n)$. However, this idea can be used to perform single scalar multiplications. This is due to the fact that the scalar multiplication kP with a n-bit scalar k and a point P may also be represented as a sum of t partial multiplications with t scalars k_i and t points P_i as follows

$$kP = (k_t \cdot 2^{(t-1)n/t} + ... + k_2 \cdot 2^{n/t} + k_1) \cdot P \quad (1)$$

$$= \sum_{i=1}^{t} k_i \cdot P_i \quad (2)$$

Thereby, each scalar k_i is a n/t-bit long part of k and $P_i = 2^{(i-1)n/t}P$. Note that n is padded with 0's from left until it is a multiple of t. In order to increase the performance the points $P_t, P_{t-1}, ..., P_2$ are ideally precomputed and stored off-line.

Another way for accelerating scalar multiplication is to reduce the required number of point additions. The point additions are preformed, if the corresponding bit of the scalar is 1. This means that the number of point additions may be further reduced, if the scalar is represented with a low *Hamming weight*, which is the number of the non-zero bits. In general the Hamming weight of a scalar is low, if it is represented in a signed representation. [17] analyzed both the *non-adjacent form* (NAF) and the *mutual opposite form* (MOF), see [15] and [14], respectively. Although both lead to the same Hamming weight, MOF is superior to NAF, because it exhibits lower memory requirements.

5. IMPLEMENTATION RESULTS

The implementation was done on the MicaZ mote [2], which is a typical device for WSNs and equipped with 8-bit processor. The operating system employed in the implementation was TinyOS-2.0 [1], an open-source operating system designed for wireless embedded sensor networks. The timing results are the average over 500 executions with random numbers and were generated using the curve parameters *secp160r1* from [16]. The operations on the lowest abstraction level were realized using Assembler, while those on higher levels were written in *NesC*.

Table 1 compares the performance of the point multiplication with existing solutions, namely TinyECC-0.2 [10] and the solution from [9] and [18]. Note that in Table 1 the results in the first row imply that the point multiplication employed in the ECElGamal was realized by using only the Left-to-Right binary method. The values in the second row of the table present the point multiplication accelerated by using 2MOF. The results presented in the remainder of the table were obtained by employing the Interleave method. Therefore, the first five rows imply *general point multiplication*, while the rest of the table represents *fixed point multiplication*.

The performance of the point multiplication from TinyECC-0.2 is not presented in [10]. Thus, the TinyECC-0.2 was integrated in the test suite employed for evaluating this work. Note that the finite

Table 2: Implementation results for EC-ElGamal

#Precom. points	Ex. time [s]
0 (L2R)	2.48
0 (2MOF)	2.16
2	1.42
4	1.19

field inversion is required for converting the elliptic curve points from Jacobian to affine coordinates during the decryption process. Since that needs to be performed only on the reader device, the finite field inversion was not implemented. However, since one inversion and four multiplications are needed for the conversion, some assumptions are necessary. According to [5], the ratio between finite field multiplications and inversions $Inv/Mult$ is 30. Hence, together with the additional four modular multiplications needed for the conversion, this results in $34 \cdot 0.532ms = 18ms$, where $0.532ms$ is the execution time of one modular multiplication, see [17]. For a fair comparison, this value is already added to the results in Table 1. Finally, Table 2 shows the performance of the different realizations of the EC-ElGamal, which contains two point multiplications with n-bit scalar k and one short point multiplication with the sensed data m, see Algorithm 1. Note that for testing purposes m was chosen to be 8-bit.

6. CONCLUSION

We implemented elliptic curve and finite field arithmetic operations on MicaZ mote which is a typical device employed in wireless sensor networks. Our tests showed that scalar point multiplication with a random base takes 1.03s, while it takes only 0.57s in the case of fixed point multiplication, when 2 precomputed points are employed. According to our knowledge, our implementation is the second fastest for general point multiplication. Moreover, compared to the best previous result, our implementation is at least 44% faster for fixed point multiplication. The implementation of the elliptic curve EC-ElGamal encryption showed that the encryption operation only takes 1.19s, when in total 4 precomputed points are utilized.

7. ACKNOWLEDGMENTS

The work presented in this paper is supported in part by the European Commission within the STREP UbiSec&Sens. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the UbiSec&Sens project or the European Commission.

8. REFERENCES

[1] `http://www.tinyos.net`.

[2] *Crossbow Technology Inc.* `http://www.xbow.com`.

[3] I. Branovic, R. Giorgi, and E. Martinelli. Memory Performance of Public-Key cryptography Methods in Mobile Environments. In *ACM SIGARCH Workshop on MEmory performance: DEaling with Applications, systems and architecture (MEDEA-03)*, pages 24–31, New Orleans, LA, USA, September 2003.

[4] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient Aggregation of Encrypted Data in Wireless Sensor Networks. *3rd Intl. Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Sensor Networks, Italy*, April 2005.

[5] H. Cohen, A. Miyaji, and T. Ono. Efficient Elliptic Curve Exponentiation Using Mixed Coordinates. In *Advances in Cryptology - ASIACRYPT '98*, volume 1514 of *Lecture Notes in Computer Science*, pages 51–65, London, UK, 1998. Springer-Verlag.

[6] T. E. Gamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. In *Advances in cryptology – Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer-Verlag New York, Inc., 1985.

[7] J. Girao, D. Westhoff, E. Mykletun, and T. Araki. TinyPEDS: Tiny Persistent Encrypted Data Storage in Asynchronous Wireless Sensor Networks. *Elsevier Ad Hoc Journal*, 5(7):1073–1089, September 2007.

[8] J. Großschädl, R. M. Avanzi, E. Savas, and S. Tillich. Energy-efficient software implementation of long integer modular arithmetic. In *Cryptographic Hardware and Embedded Systems*, volume 3659 of *Lecture Notes in Computer Science*, pages 75–90, 2005.

[9] N. Gura, A. Patel, A. Wander, H. Eberle, and S. C. Shantz. Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. In *Cryptographic Hardware and Embedded Systems - CHES*, volume 3156 of *Lecture Notes in Computer Science*, pages 119–132, Cambridge, MA, USA, August 2004.

[10] A. Liu and P. Ning. *TinyECC: Elliptic Curve Cryptography for Sensor Networks*, September 2006.

[11] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1997.

[12] B. Möller. Algorithms for Multi-exponentiation. In *Selected Areas in Cryptography – SAC 2001*, volume 2259 of *Lecture Notes In Computer Science*, pages 165–180, London, UK, 2001. Springer-Verlag.

[13] E. Mykletun, J. Girao, and D. Westhoff. Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks. In *IEEE Int. Conference on Communications – ICC*, Istanbul, Turkey, June 2006.

[14] K. Okeya, K. Schmidt-Samoa, C. Spahn, and T. Takagi. Signed Binary Representations Revisited. In *Advances in Cryptology - CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 123–139, 2004.

[15] G. W. Reitwiesner. Binary arithmetic. *Advances in Computers*, 1:231–308, 1960.

[16] Standards for Efficient Cryptography Group (SECG). *Specification of Standards for Efficient Cryptography — SEC 2: Recommended Elliptic Curve Domain Parameters*, September 2000.

[17] O. Ugus. Asymmetric Homomorphic Encryption Transformation for Securing Distributed Data Storage in Wireless Sensor Networks. Master's thesis, Technische Universität Darmstadt - in Cooperation with NEC Europe Ltd., Heidelberg, Darmstadt, 2007.

[18] H. Wang and Q. Li. Efficient Implementation of Public Key Cryptosystems on Mote Sensors. In *Eighth International Conference on Information and Communications Security (ICICS 2006)*, volume 4307 of *Lecture Notes in Computer Science*, pages 519–528, December 2006.

SNIF: A Comprehensive Tool for Passive Inspection of Sensor Networks

Matthias Ringwald, Kay Römer
Institute for Pervasive Computing
ETH Zurich, Switzerland
{mringwald,roemer}@inf.ethz.ch

ABSTRACT

Deployment of sensor networks in real-world settings is a labor-intensive and cumbersome task: environmental influences often trigger problems that are difficult to track down due to limited visibility of the network state. In this extended abstract, we summarize our ongoing efforts to develop a tool for passive inspection of sensor networks, where the network state can be inferred without instrumentation of sensor nodes. We also discuss next steps to make this tool applicable to a larger class of applications.

1. INTRODUCTION

Deployment of sensor networks in real-world settings is typically a labor-intensive and cumbersome task (e.g. [5, 6, 12, 13, 15]). While simulation and lab testbeds are helpful tools to test an application prior to deployment, they fail to provide realistic environmental models (e.g., regarding radio signal propagation, sensor stimuli, chemical/mechanical strain on sensor nodes). Hence, environmental effects often trigger bugs or degrade performance in a way that could not be observed during pre-deployment testing. To track down such problems, a developer needs to inspect the state of network and nodes. While this is easily possible during simulation and experiments on lab testbeds (wired backchannel from every node), access to network and node states is very constrained after deployment.

Current practice to inspect a deployed sensor network requires *active* instrumentation of sensor nodes with monitoring software. Monitoring traffic is sent in-band with the sensor network traffic to the sink (e.g., [6, 11, 14]). Unfortunately, this approach has several limitations. Firstly, problems in the sensor network (e.g., partitions, message loss) also affect the monitoring mechanism, thus reducing the desired benefit. Secondly, scarce sensor network resources (energy, cpu cycles, memory, network bandwidth) are used for inspection. Thirdly, the monitoring infrastructure is tightly interwoven with the application. Hence, adding/removing instrumentation may change the application behavior in subtle ways, causing probe effects. Also, it is non-trivial to adopt the instrumentation mechanism to different applications.

In contrast to the above, we propose a *passive* approach for sensor network inspection by overhearing and analyzing sensor network traffic to infer the existence and location of typical problems encountered during deployment. To overhear network traffic, a so-called *deployment support network* (DSN) [1] is used: a wireless network that is temporarily installed alongside the actual sensor network during the deployment process (see Fig. 1). Each DSN node provides two different radio front-ends. The first radio is used to overhear the traffic of the sensor network, while the second radio is used to form a robust and high-bandwidth network among the DSN nodes to reliably collect overheard packets. A data stream framework performs online analysis of the resulting packet stream to infer and report problems soon after their occurrence.

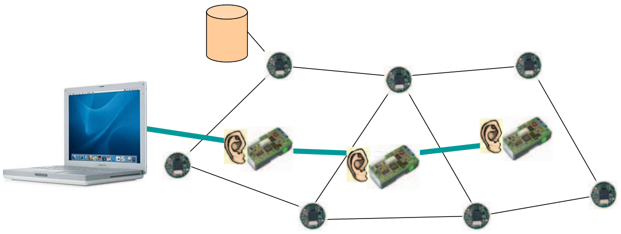


Figure 1: A deployment-support network (rectangular nodes) is a physical overlay network that overhears sensor network (round nodes) traffic and delivers it to a sink using a second radio.

This approach removes the above limitations of active inspection: no instrumentation of sensor nodes is required, sensor network resources are not used. The inspection mechanism is completely separated from the application, can thus be more easily adopted to different applications, and can be added and removed without altering sensor network behavior.

So far, we analyzed and classified problems typically found during deployment [7], implemented a basic version of the Sensor Network Inspection Framework (SNIF) [10], and conducted a case study on data gathering applications [9] to demonstrate the feasibility and benefits of our approach. Although possible in principle, it is currently difficult to adopt SNIF to application types other than data gathering application. Further work is needed to add this missing flexibility. In the remainder of this abstract, we give an overview of SNIF's architecture and discuss next steps.

2. PASSIVE INSPECTION OF DATA GATHERING APPLICATIONS

SNIF supports the detection of specific *problems* (e.g., node reboot) that occur during deployment of a sensor network. For this, we first need to identify the problems that can occur and that should be detected by SNIF. Secondly, we need to provide *passive indicators* for each of the problems, which allow to infer the existence of a problem from observed packet traces. For this, one needs to analyze the message protocols used in the sensor network. Finally, SNIF needs to be configured to implement these passive indicators.

In our work, we focus on so-called *data gathering applications* (e.g., [12, 15]), where nodes send raw sensor readings at regular intervals along a spanning tree across multiple hops to a sink. The reason for our choice is that almost all existing non-trivial deployments are data gathering applications. Below, we will first characterize data gathering applications in more detail, before presenting typical problems with these applications and matching passive indicators.

2.1 Application Model

Systems for data gathering such as the Extensible Sensing System (ESS) [4] need to maintain a spanning tree of the network along which sensor values are routed to the sink. To support neighbor discovery, all nodes broadcast *beacon messages* at regular intervals. Each beacon message contains a sequence number. To discover neighbors, nodes overhear these messages and estimate the quality of incoming links from neighbors based on message loss. Nodes then broadcast *link advertisement messages* at regular intervals, containing a list of neighbors and link quality estimates. Overhearing these messages, nodes compute the bidirectional link quality to decide on a good set of neighbors. To construct a spanning tree of the network with the sink at the root, nodes broadcast *path advertisement messages*, containing the quality of their current path to the sink. Nodes overhearing these messages can then select the neighbor with the best path as their parent and broadcast an according path advertisement message. Finally, *data messages* are sent from nodes to the sink along the edges of the spanning tree across multiple hops.

2.2 Problems and Indicators

A indicator is an observable behavior of a sensor network that hints (in the sense of a heuristic) the existence of a specific problem. We are interested in *passive* indicators that can be observed purely by overhearing the traffic of the sensor network as this does not require any instrumentation of the sensor nodes.

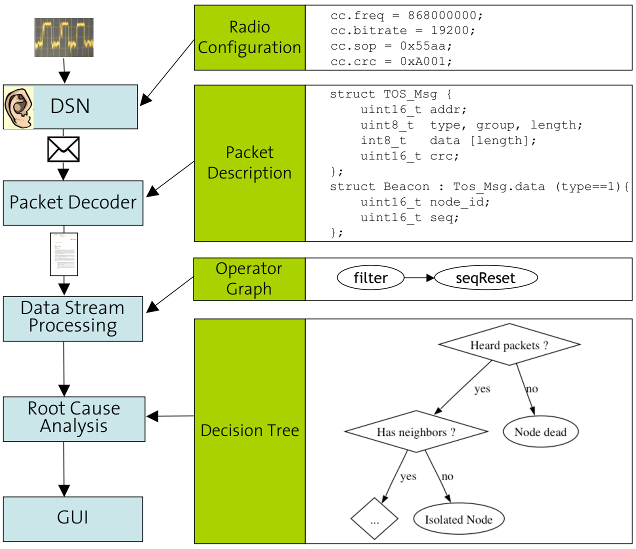
In [7] we studied existing deployments to identify common problems and derived passive indicators for them. We classify problems according to the number of nodes involved into four classes based on existing deployments: *node problems* that involve only a single node, *link problems* that involve two neighboring nodes and the wireless link between them, *path problems* that involve three or more nodes and a multi-hop path formed by them, and *global problems* that are properties of the network as a whole. Below, we provide for each category an exemplary problem and passive indicators to detect it.

Node reboot, as an example for a node problem, causes the sequence number counter of the affected node to be reset to an initial value (typically zero). Hence, the sequence number contained in beacon messages sent by the node will jump to a smaller value after a reboot with high probability even in case of lost messages, which can serve as an indicator for reboot.

An *isolated node*, as an example for a link problem, has no neighbors in the network topology. An indicator for this problem is that the node is not listed in any link advertisement messages send by other nodes.

An *orphaned node*, as an example for a path problem, has no parent in the routing tree. Such nodes will either send no path announcement messages at all or path announcements contain an infinite distance to the sink (depending on the protocol details), which can be used as a passive indicator.

A *partitioned node*, as an example of a global problem, is disconneted from the sink, for example due to death of a node on the path. A node is considered as partitioned if all paths from the node to the sink involve dead nodes. This predicate requires an approximate view on the network topology which is reconstructed on the base of observed data packets. Periodic checks on the reconstructed topology serve as a passive indicator here.



```
cc.freq = 868000000;
cc.bitrate = 19200;
cc.sop = 0x55aa;
cc.crc = 0xA001;
```

```
struct TOS_Msg {
    uint16_t addr;
    uint8_t  type, group, length;
    int8_t   data [length];
    uint16_t crc;
};
struct Beacon : Tos_Msg.data (type==1){
    uint16_t node_id;
    uint16_t seq;
};
```

Figure 2: Architecture of SNIF.

3. SNIF

In this section we outline how passive indicators discussed in the previous section can be implemented in SNIF. For this, consider the architecture of SNIF as depicted in Fig. 2, which consists of a deployment support network to overhear sensor network traffic, a packet decoder to access the contents of overheard packets, a data stream processor to analyze packet streams for problems, a decision tree to infer the state of each sensor node, and a user interface to display these states. The key design goal for SNIF is generality that is, it should support passive inspection of a wide variety of sensor network protocols and applications. Below we give an overview of these components. More details can be found in a [10].

3.1 Deployment Support Network (DSN)

To overhear the traffic of multi-hop networks, multiple radios are needed, forming a distributed network sniffer. We use a so-called deployment support network for this purpose, a wireless network of DSN nodes, each of which provides two radios.

Our current implementation of a DSN is based on the BTnode Rev. 3 [2], which provides two radio front-ends: a Zeevo ZV 4002 Bluetooth 1.2 radio which is used as the DSN radio, and a Chipcon CC 1000 (e.g., also used on MICA2) which is used as the WSN radio. Using a scatternet formation algorithm, the DSN nodes form a robust Bluetooth scatternet (see [1] for details). A laptop computer with Bluetooth acts as the SNIF sink that connects to a nearby DSN node. This DSN node thereupon acts as the DSN sink and forms the root of an overlay tree spanning the whole DSN. The SNIF sink can send data to DSN nodes down the tree while DSN nodes send overheard packets up the tree to the sink.

Time synchronization exploits the fact that Bluetooth uses a TDMA MAC protocol and thus performs clock synchronization internally, providing an interface to read the Bluetooth clock and its offset to the clocks of network neighbors. We use this interface to compute the clock offset of each DSN node to the DSN sink. A detailed description of our time synchronization protocol can be found in [8].

3.2 Physical Layer and Medium Access

DSN nodes need a receive-only implementation of the physical (PHY) and MAC layers in order to overhear sensor network traffic. Due to the lack of a standard protocol stack, many variants of PHY and MAC are in use in sensor networks. Our generic PHY implementation supports configurable carrier frequency, baud rate, and checksumming details as illustrated in Fig. 2. Regarding MAC, we exploit the fact that – regardless of the specific MAC protocol used – a radio packet always has to be preceded by a preamble and a start-of-packet (SOP) delimiter to synchronize sender and receiver. In our generic MAC implementation, every DSN node has its WSN radio turned to receive mode all the time, looking for a preamble followed by the SOP delimiter in the received stream of bits. Once an SOP has been found, payload data and a CRC follow. This way, DSN nodes can receive packets independent of the actual MAC layer used.

3.3 Packet Decoder

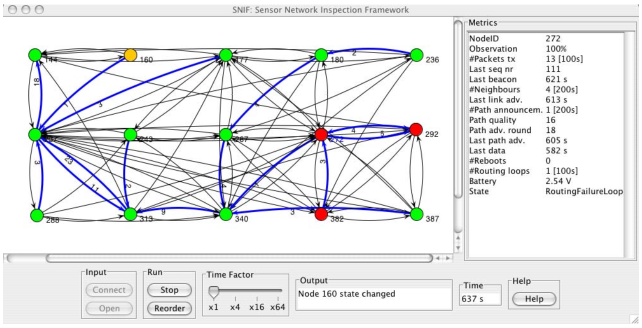
Again, since no standard protocols exist for sensor networks, we need a flexible mechanism to decode overheard packets. Since most programming environments for sensor nodes are based on the C programming language or a dialect of it (e.g., nesC for TinyOS), it is common to specify message contents as (nested) C structs in the source code of the sensor network application. Our packet decoder uses an annotated version of such C structs as a description of the packet contents. This way, the user can copy and paste packet descriptions from the source code.

The configuration of the packet decoder consists of some global parameters (such as byte order and alignment), type definitions, and one or more C structs. One of these structs is indicated as the default packet layout. Note that such a struct can contain nested other structs, effectively implementing a discriminated union.

Fig. 2 shows an example of a TinyOS message (`TOS_Msg`) holding a beacon data unit if the message type of the TinyOS message equals 1. The result of packet decoding is a record consisting of a list of name-value pairs, where each pair holds the name and value of a data field in the packet.

3.4 Data Stream Processor

The resulting stream of packets is then fed to a data stream processor to detect any problems with the sensor network. The data stream processor executes operators that take a stream of records as input and produce a different stream of records as output. The output of an operator can be connected to the input of other operators, resulting in a directed operator graph. SNIF provides a set of standard operators, e.g., for filtering, aggregation over time windows, or merging of multiple streams into one. In addition, application-specific operators to detect specific problems in the sensor network may be required. Fig. 2 shows an simple operator graph that is used to detect node reboots as described in Sect. 2.2. The first operator (`filter`) reads the packet stream generated by the DSN and removes all packets that are not beacon packets. The second operator (`seqReset`) remembers the last sequence number received from each node and checks if a newly received sequence number is smaller than the previous one for this node, in which case the node has rebooted unless there was a sequence number wrap-around (i.e., maximum sequence number has been reached and sequence counter wraps to zero).



```
NodeID              272
Observation         100%
#Packets tx         13 [100s]
Last seq nr         111
Last beacon         621 s
#Neighbours         4 [200s]
Last link adv.      613 s
#Path announcem.    1 [200s]
Path quality        16
Path adv. round     18
Last path adv.      605 s
Last data           582 s
#Reboots            0
#Routing loops      1 [100s]
Battery             2.54 V
State               RoutingFailureLoop
```

Output: Node 160 state changed

Time: 637 s

Figure 3: An instance of SNIF's user interface.

3.5 Root Cause Analysis

The next step is to derive the state of each sensor node, which can be either "node ok" or "node has problem X". Note that the operator graphs mentioned above may concurrently report multiple problems for a single node. In many cases, one of the problems is a consequence of another problem. For example, a node that is dead also has a routing problem. In such cases, we want to report only the primary problem and not secondary problems. For this, we use a decision tree, where each internal node is a decision that refers to the output of an operator graph, and each leaf is a node state. In the example tree depicted in Fig. 2, we first check (using the output of an operator graph that counts packets received during a time window) if any messages have been received from a node. If not, then the state of this node is set to "node dead". Otherwise, if we received packets from this node, we next check if this node has any neighbors (using an operator graph that counts the number of neighbors contained in link advertisement packets received from this node). If there are no neighbors, then the node state is set to "node isolated". Here, the check for node death is above the check for isolation in the decision tree, because a dead node (primary problem) is also isolated (secondary problem).

3.6 User Interface

Finally, node states and additional information are displayed in the graphical user interface. The core abstraction implemented by the user interface is a network graph, where nodes and links can be annotated with arbitrary information. The user interface also supports recording and playback of executions. A snapshot of an instance of the user interface is shown in Fig. 3. Here, node color indicates state (green: ok, gray: not covered by DSN, yellow: warning, red: severe problem), detailed node state can displayed by selecting nodes. Thin arcs indicate what a node believes are its neighbors, thick arcs indicate the paths of multi-hop data messages.

4. RELATED WORK

Most closely related to SNIF is work on active debugging of sensor networks, notably Sympathy [6] and Memento [11]. However, both systems require instrumentation of sensor nodes and introduce monitoring protocols in-band with the actual sensor network traffic. Also, both tools only support a fixed set of problems, while SNIF provides an extensible framework.

Tools for sensor network management such as NUCLEUS [14] provide read/write access to various parameters of a sensor node that may be helpful to detect problems. However, this approach also requires active instrumentation of the sensor network.

5. NEXT STEPS

As mentioned in Sect. 2.1, our current work is focused on data gathering applications. As other types of applications such as tracking and event detection are deployed, we will analyze experiences gained from deployments and add support for inspection of these applications to SNIF. For this, novel indicators may have to be implemented in SNIF. While SNIF supports this flexibility in principle through composition and parametrization of data stream operators, currently Java code needs to be written and the developer has to be familiar with SNIF internals. One of the next steps is therefore the development of appropriate high-level specification techniques to support more convenient configuration of SNIF for different types of applications. In particular, we envision a graphical notation, allowing a user to devise these specifications using a graphical user interface.

Ultimately, we want to achieve (semi-)automatic generation of these specifications from application programs. For this, we will work on analyzing high-level declarative program specifications such as SNlog [3]. These capture the application semantics in a more direct way than procedural programs, such that it may be possible to derive SNIF configurations without explicit annotations.

6. ACKNOWLEDGMENTS

The work presented in this paper was partially 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.

7. REFERENCES

[1] J. Beutel, M. Dyer, L. Meier, and L. Thiele. Scalable Topology Control for Deployment-Sensor Networks. In *IPSN 2005*.

[2] BTnodes. A Distributed Environment for Prototyping Ad Hoc Networks. www.btnode.ethz.ch.

[3] D. C. Chu, L. Popa, A. Tavakoli, J. M. Hellerstein, P. Levis, S. Shenker, and I. Stoica. The design and implementation of a declarative sensor network system. Technical Report UCB/EECS-2006-132, EECS Department, UC Berkeley, October 2006.

[4] R. Guy, B. Greenstein, J. Hicks, R. Kapur, N. Ramanathan, T. Schoellhammer, T. Stathopoulos, K. Weeks, K. Chang, L. Girod, and D. Estrin. Experiences with the Extensible Sensing System ESS. Technical Report 61, CENS, 2006.

[5] P. Padhy, K. Martinez, A. Riddoch, H. L. R. Ong, and J. K. Hart. Glacial Environment Monitoring using Sensor Networks. In *REALWSN 2005*.

[6] N. Ramanathan, K. Chang, R. Kapur, L. Girod, E. Kohler, and D. Estrin. Sympathy for the Sensor Network Debugger. In *SenSys 2005*.

[7] M. Ringwald and K. Römer. Deployment of Sensor Networks: Problems and Passive Inspection. In *WISES 2007*.

[8] M. Ringwald and K. Römer. Practical Time Synchronization for Bluetooth Scatternets. In *BROADNETS 2007*.

[9] M. Ringwald, K. Römer, and A. Vialetti. Passive Inspection of Sensor Networks. In *DCOSS 2007*.

[10] M. Ringwald, K. Römer, and A. Vialetti. SNIF: Sensor Network Inspection Framework. Technical Report 535, Departement of Computer Science, ETH Zurich, 2006.

[11] S. Rost and H. Balakrishnan. Memento: A Health Monitoring System for Wireless Sensor Networks. In *SECON 2006*.

[12] R. Szewcyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *SenSys 2004*.

[13] J. Tateson, C. Roadknight, A. Gonzalez, S. Fitz, N. Boyd, C. Vincent, and I. Marshall. Real World Issues in Deploying a Wireless Sensor Network for Oceanography. In *REALWSN 2005*.

[14] G. Tolle and D. Culler. Design of an Application-Cooperative Management System for Wireless Sensor Networks. In *EWSN 2005*.

[15] G. Tolle, J. Polastre, R. Szewczyk, D. Culler, N. Turner, K. Tu, S. Burgess, T. Dawson, P. Buonadonna, D. Gay, and W. Hong. A Macroscope in the Redwoods. In *SenSys 2005*.

Management of Heterogeneous Wireless Sensor Networks[*]

Markus Anwander, Gerald Wagenknecht, Thomas Staub, and Torsten Braun
Institute of Computer Science and Applied Mathematics
Neubrückstrasse 10
3012 Bern, Switzerland
{anwander|wagen|staub|braun}@iam.unibe.ch

ABSTRACT

Wireless sensor networks (WSNs) are taking a big step forward to productive deployments. Heterogeneous WSNs are gaining importance. Complex problem settings consisting of different environmental conditions require specific sensor nodes for the individual tasks resulting in heterogeneous networks. As the different types of sensor nodes may be incompatible, a more general management architecture for these heterogeneous environments is a necessity. Individual nodes have to be reconfigured and updated during their lifetime. Our WSN management framework supports common management tasks such as monitoring the WSN, configuration of the WSN, code updates, and managing sensor data. Our management architecture consists of the following infra-structural elements: a management station, a number of management nodes and a high number of heterogeneous sensor nodes. All management tasks are controlled by the management station. Management nodes are implemented as wireless mesh nodes.

1. INTRODUCTION

A WSN consists of a number of sensor nodes. They have a large area of applications, e.g., event detection, localization, tracking, monitoring and many more, which require specific nodes to perform the individual tasks. This results in heterogeneous networks.

Currently available sensor nodes are mainly prototypes for research purposes. A number of sensor nodes have been evaluated. We selected four of them to build a heterogeneous sensor network: ESB nodes [1], tmote SKY [2], BTnodes [3] and micaZ [4]. For the management backbone a Wireless Mesh Network (WMN) consisting of Wireless Router Application Platform boards (WRAP) [5] nodes have been selected.

Contiki [6] is being used as operating system running on the sensor nodes. It is a dynamic operating system with special focus on portability. It is written in C and supports 14 platforms and 5 CPU types. A small TCP/IP [7] stack (μIP) is implemented. Contiki, moreover supports preemptive multithreading, inter-process communication and dynamic runtime linking of standard Executable Linkable Format (ELF) files. Program modules can be updated and loaded at runtime. Contiki and the network simulator COOJA are open source projects and run under BSD license.

To improve current research our concept adds mechanisms to support heterogeneity for management in WSNs. MANNA [8] presents an information architecture and a functional management architecture for WSNs. The management architecture provides functions to establish configurations for sensor network entities. Each of the hierarchical deployed manager nodes is responsible for a cluster of sensor nodes. The information architecture defines the information units and the information exchange among the entities. Currently no implementation of MANNA exists. Guidelines are proposed, but the communication model and other issues are not yet defined. TinyCubus [9] presents a management and configuration framework for WSNs. It also bases on a clustered architecture assigning certain roles to sensor nodes. Another focus is code distribution minimizing the code fragments to be distributed in a WSN. Reliability shall be supported by implicit acknowledgments and retransmissions. The code distribution mechanism has been evaluated in rather friendly environments without high error rates. The Deployment Support Network (DSN) to observe, control, and reprogram a deployed WSN over the air is presented in [10]. Major drawback of this approach is the need of an additional wireless backbone network. The Global Sensor Networks (GSN) [11] provides a middleware for fast and flexible integration and deployment of heterogeneous sensor networks. The key concept in GSN is the usage of virtual sensors, which abstract from implementation details of access to sensor data and correspond either to data streams received directly from sensors. [12] surveys software update techniques in WSNs. Its design space consists of the execution environment at the sensor nodes, the software distribution protocol in the network and optimization of transmitted updates.

The following sections describe the WSN management framework including the management scenario and the management tasks (Section 2), the management architecture

---

[*]This work has been supported by the Hasler Foundation under grant number ManCom 2060 and the Swiss National Science Foundation under grant number 200020-113677/1

(Section 3), as well as the management protocols (Section 4). Future work is presented in Section 5.

2. MANAGEMENT SCENARIO AND TASKS

A heterogeneous wireless sensor network consists of different types of sensor nodes, which might measure different data and perform different tasks. To operate such a (sub)network the following devices are required: one management station, several mesh nodes and a comparatively high number of heterogeneous sensor nodes. A possible scenario is shown in Figure 1.

The sensor nodes might have different sensors for monitoring the environment. All sensor nodes of one type are able to communicate with each other and build a sensor subnet. Many existing sensor platforms have different radio modules and are thus not able to communicate with each other. The different subnets are interconnected by wireless mesh nodes. They provide interfaces for different sensor subnets and act as gateways. Besides the inter-subnet communication, the mesh nodes perform management tasks. Each mesh node is responsible for one or more subnets. Control of the sensor nodes is done via the management station. The management station is connected to the mesh nodes via Ethernet or via IEEE 802.11.

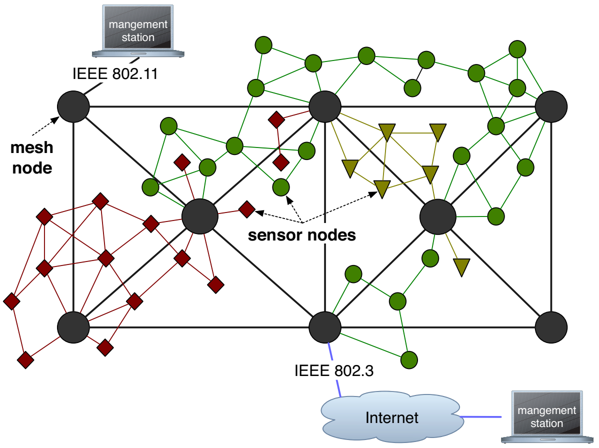


Figure 1: A possible management scenario

From the management point of view there are several tasks required to manage a WSN and its sensor nodes. In general we can divide these into four areas: (1) monitoring the WSN and the sensor nodes, (2) (re)configuring the WSN and the sensor nodes, (3) updating the sensor nodes and (4) managing the sensor data.

The **monitoring task** requires that all sensor nodes in the several subnets are displayed at the management station with all necessary information. This includes sensor node hardware details (e.g. chip, transceiver), sensor node software details (e.g. operating system versions), and dynamic properties (e.g. battery). The node ID and other static information is sent when a sensor node joins the network. Additionally the management station may query sensor nodes. The **(re)configuration task** includes sensor node configuration and network configuration. **Code distribution** mechanisms perform the operating system or the application updates. Mechanisms to handle incomplete, inconsistent and failed updates have to be provided.

3. MANAGEMENT ARCHITECTURE

The management architecture contains the following structural elements: a management station, some mesh nodes as management nodes, sensor node gateways plugged into a mesh node, and the different sensor nodes.

3.1 Management Station

The management station is divided into two parts. It consists of a laptop or remote workstation to access a web interface to control the WSN and a mesh node running the management system for Wireless Mesh Networks (WMNs) [13] including a web server (shown in Figure 2).

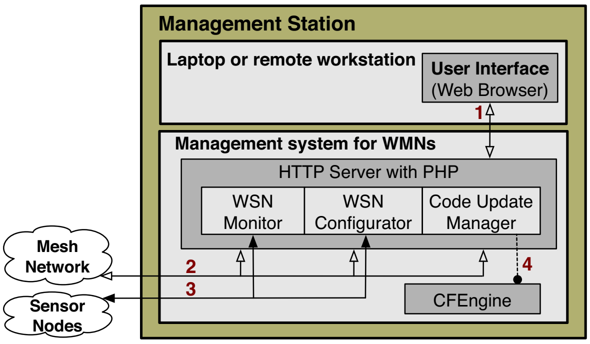


Figure 2: Management station architecture

The communication between the **user interface** and the management system for WMNs is done via HTTPS. The **management system for WMNs** contains a small Linux distribution including all required applications, especially a HTTP server supporting PHP. The HTTP server maintains several modules to handle the requests and transmit them to mesh nodes, sensor nodes or CFEngine [14]. Communication with a mesh node is done via TCP/IP with HTTPS servers running on the mesh nodes (depicted as **2** in Figure 2). The communication between the management station and the sensor nodes is done via TCP/IP (depicted as **3** in Figure 2). For data transmission within the mesh network, CFEngine is used. The **WSN monitor** is responsible for monitoring the whole network. It shows the mesh nodes with their subordinate sensor nodes including all available information of the sensor nodes. The user may request information from a single sensor node. The **WSN configurator** is responsible for the configuration of the WSN. The **code update manager** distributes the uploaded image via CFEngine. It shows the available program versions and performs the updating process.

3.2 WSN Manager

The WSN manager running on the mesh nodes consists of the following components: program version database, WSN information database, sensor database, WSN monitor module, WSN configurator module, and code update manager module (shown in Figure 3). The **sensor database** stores all measured data as tuples (node ID, sensor ID, value,

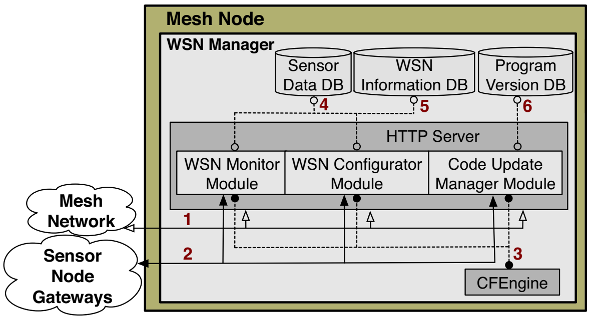


Figure 3: Mesh node architecture

timestamp). The **WSN information database** consists of all infra-structural data. Each entry contains ID, property ID, value, and timestamp. The properties are e.g. chip, transceiver, and battery status. The **program version database** stores the versions of all available programs containing program ID, version, target platform, timestamp, and link to the image. The **CFEngine** is responsible for distributing the databases within the mesh network. The **WSN monitor module** connects to the WSN information DB and to the sensor data DB for responding the requests from the management station. It writes sensor and node data into the databases. The **WSN configurator module** connects to the WSN information database to read and write data. It further queries the sensor node properties and sends commands. The **code update manager module** stores newly uploaded images in the program version DB. It also updates the sensor nodes. It includes methods to reduce the distributed code (differential patch, compression).

3.3 Sensor Node Manager

As shown in Figure 4, the management tasks are handled by a **sensor node manager**. It consists of sensor node monitor, sensor node configurator, sensor data sender, and code updater. The **sensor node monitor** sends the requested values to the mesh node. The **sensor node configurator** executes the configuration requests and notifies the mesh node. The **code updater** receives the image of the application or operating system (differential patch, compressed, or uncompressed) and performs the update. It confirms the success of the update to the mesh node.

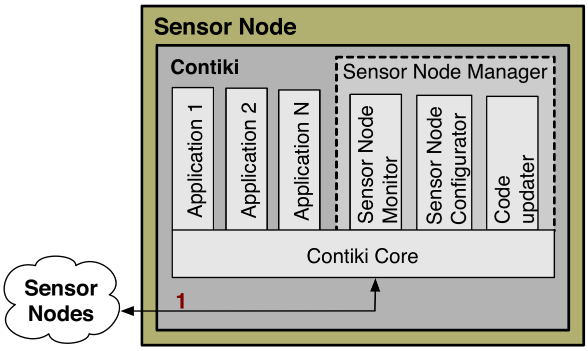


Figure 4: Sensor node architecture

4. WSN MANAGEMENT PROTOCOLS

4.1 WSN Monitoring Protocol

The monitoring protocol enables a management node to get all information about the network topology, all sensor node properties and all measured sensor data. It can be divided into 2 cases: The first case describes how the management station explores the mesh network and the subordinate sensor node networks. The second case describes the situation when the user queries a selected sensor directly.

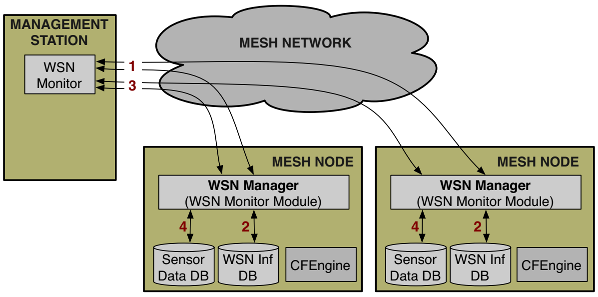


Figure 5: WSN monitor queries the mesh nodes

When the management station joins the mesh network, it connects to the next available mesh node. Because the information is distributed using CFEngine within the mesh network, the information about the WSN topology of distant mesh nodes is several minutes old. In order to receive the actual topology, the management station queries all other mesh nodes (Figure 5). The protocol works as follows: (1) The management station queries all mesh nodes about their subordinated sensor nodes. (2) The WSN monitor modules queries the WSN information DB. (3) The management station requests the current sensor data of every subordinated sensor node. (4) The WSN monitor module queries the sensor data DB.

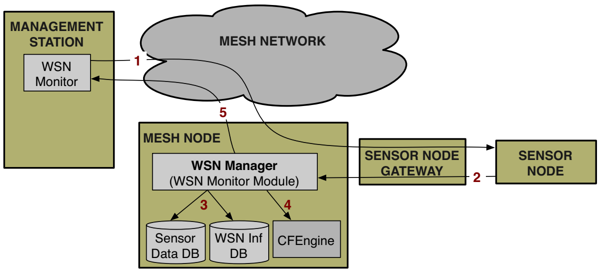


Figure 6: User requests sensor node information directly

The second case is shown in Figure 6 and works as follows: (1) The user requests either sensor node information or sensor data from a single sensor node or from a group of sensor nodes. The request is transmitted via a unicast, multicast or broadcast transport protocol to the queried sensor nodes. (2) The sensor sends the requested information back to the mesh node. (3) The WSN manager module writes the new information into the according database. (4) It copies the information to CFEngine which distributes it within the WMN. (5) The WSN sends a confirmation to the WSN monitor.

4.2 WSN Configuration Protocol

With the WSN configuration protocol the properties of the sensor nodes as well as the network can be configured. Examples are changing sensing intervals or routing tables. It works similar as the WSN monitoring described in 4.1. The request message contains a configuration command.

When a new sensor node joins the following tasks are performed (see Figure 7): (1) First, it broadcasts a 'Hello' message to the WSN configurator module. (2) An initial network configuration is negotiated. Then all available information of the sensor node is requested. (3) The sensor node is registered in the WSN information DB. (4) All available information is propagated to CFEngine for distribution.

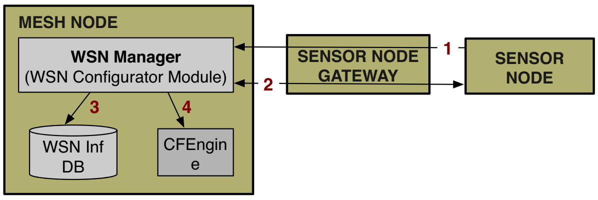


Figure 7: A new sensor node joins the sensor network

4.3 Code Update Protocol

The code update protocol contains mechanisms to upload and distribute the images within the mesh network, notifying the management station about the available programs, and performing the update. The new image of an application or the operating system is uploaded and stored in the program version database and distributed within the WMN. The management station is notified which programs are available in the program version database.

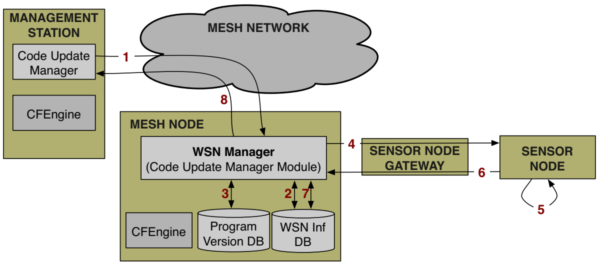


Figure 8: The user initiates the code update for the sensor node

Figure 8 shows the update process of the sensor nodes: (1) The program version and the sensor nodes are selected. The code update manager sends this request to the concerned mesh nodes. (2) The installed program version is checked by querying the WSN information DB. (3) A patch is generated from the old and new image. (4) This is sent to the selected sensor nodes. (5) On the sensor node the update is installed. (6) The update is acknowledged. (7) The WSN information DB is updated. (8) The management station is notified about the success.

5. FUTURE WORK

After defining the management architecture and selecting the appropriate sensor node platforms and operating system, the next tasks concern the implementation of the sensor node management architecture. Other important tasks are the development of reliable communication mechanisms for WSNs. This includes reliable transport protocols for unicast, multicast and broadcast communication. The next step is to develop a reliable point-to-point transport protocol. In detail, the Contiki TCP stack for Distributed TCP Caching (DTC) [15] mechanism and TCP Support for Sensor networks (TSS) [7] will be extended.

6. REFERENCES

[1] J. Schiller, A. Liers, H. Ritter, R. Winter, Th. Voigt. ScatterWeb Low Power Sensor Nodes and Energy Aware Routing. *HICSS-38, Hawaii, USA, Jan 2005.*

[2] Tmote SKY. http://www.moteiv.com. Last visit 04.07.

[3] J. Beutel, M. Dyer, M. Hinz, L. Meier, M. Ringwald. Next-generation prototyping of sensor networks. *SenSys'04, Baltimore, USA, Nov 2004.*

[4] J. Hill, D. Culler. MICA: A Wireless Platform for Deeply Embedded Networks. *IEEE Micro, November 2002.*

[5] WRAP. http://www.pcengines.ch. Last visit 04.07.

[6] A. Dunkels, B. Grönvall, Th. Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. *1st IEEE Workshop on Embedded Networked Sensors, Tampa, USA, Nov 2004.*

[7] T. Braun, T. Voigt, A. Dunkels. TCP Support for Sensor Networks. *IEEE/IFIP (WONS 2007), Obergurgl, Austria, Jan 2007.*

[8] L. Ruiz, J. Nogueira, A. Loureiro. Manna: A management architecture for wireless sensor networks *IEEE Communications Magazine, Vol. 41, No. 2, Feb 2003, pp. 116-125.*

[9] P. Marron, A. Lachenmann, D. Minder, M. Gauger, O. Saukh, K. Rothermel. Management and configuration issues for sensor networks *Int. Journal of Network Management, Vol. 15, 2005, pp. 235-253.*

[10] M. Dyer, J. Beutel, L. Thiele, T. Kalt, P. Oehen, K. Martin, P. Blum. Deployment Support Network - A Toolkit for the Development of WSNs *EWSN'07, Delft, Jan 2007.*

[11] K. Aberer, G. Alonso, D. Kossmann. Data Management for a Smart Earth. *SIGMOD Record, Vol. 35, No. 4, Dec 2006.*

[12] C. Han, R. Kumar, R. Shea, M. Srivastava. Sensor network software update management: a survey. *Int. Journal of Network Management, Vol. 15, 2005, pp. 283-294.*

[13] T. Staub, D. Balsiger, M. Lustenberger, T. Braun. Secure Remote Management and Software Distribution for Wireless Mesh Networks *ASWN'07, Santander, Spain, May 2007.*

[14] Cfengine. http://www.cfengine.org. Last visit 04.07.

[15] A. Dunkels, T. Voigt, H. Ritter, and J. Alonso. Distributed TCP Caching for Wireless Sensor Networks. *Annual Mediterranean Ad Hoc Networking Workshop, Bodrum, Turkey, Jun 2004.*

Verteiltes Sniffen von IEEE 802.15.4 Netzen unter Zuhilfenahme eines WLAN ad-hoc Netzwerks

Lasse Thiem und Klaus Scholl
Fraunhofer Institut für Offene Kommunikationssysteme (FOKUS)
Kaiserin Augusta Allee 31
10589 Berlin
Germany
+ 49 (0)30 3463 7297

{Lasse.Thiem, Klaus.Scholl}@fokus.fraunhofer.de

ZUSAMMENFASSUNG

In diesem Papier werden aktuelle Arbeiten des Fraunhofer Institutes für offene Kommunikationssysteme FOKUS zum verteilten Sniffen von IEEE 802.15.4 Netzen vorgestellt. Dabei wird sowohl auf einen verteilten Ansatz unter Nutzung von WLAN als ad-hoc Steuerungsnetz als auch auf die Visualisierung der gemessenen Daten eingegangen. Der beschriebene Ansatz sieht dabei vor, verschiedene Sensornetzwerkprotokolle auf Basis von IEEE 802.15.4 zu unterstützen.

Schlüsselwörter

WLAN, TinyOS, IEEE 802.15.4, Sniffing, Wireless Sensor Networks, Visualisierung, Interferenzen, Ad- Hoc

1. EINLEITUNG

Im Bereich der drahtlosen Sensornetzwerke wird seit einigen Jahren innerhalb einer großen Gemeinde weltweit geforscht. Schwerpunkte sind dabei ad-hoc Protokolle, Energieeinsparung und sich selbst konfigurierende Netze. Auf Seiten der erforderlichen Laufzeitumgebungen und Software-Architekturen für Sensornetze gibt es ebenfalls bereits einige brauchbare Lösungen. Das Angebot an geeigneten Management-Tools für die Installation, Konfiguration und Fehlerüberwachung von Sensornetzen ist dabei noch eher dürftig. Ein Großteil der zukünftigen Endkunden wollen Sensornetzwerke in erster Linie schnell, zuverlässig und einfach nutzen, ohne sich näher mit den internen Details beschäftigen zu müssen [1]. Aus diesem Grund müssen geeignete Tools zur Installation und Wartung entwickelt werden, die es Service Technikern ermöglichen, vor Ort schnellstmöglich Wartungsaufgaben zu übernehmen und eventuell anfallende Probleme zu lösen. Der vorliegende Artikel stellt hierfür ein intelligentes System zum verteilten Sniffing und Visualisieren von Sensornetzwerken unter Zuhilfenahme von WLAN vor. Problematisch ist hier vor allem der parallele Betrieb der beiden Funksysteme. Aus diesem Grund wird zunächst auf die grundsätzliche Interferenz-Problematik zwischen WLAN und IEEE 802.15.4 eingegangen und eine Methode zur Vermeidung dieser Probleme erläutert, um dann das System-Konzept vorzustellen.

2. WLAN/ IEEE 802.15.4 INTERFERENZEN

Grundsätzlich stehen IEEE 802.15.4 [3] und IEEE 802.11b (WLAN) [4] nicht in direkter Konkurrenz zueinander. WLAN wurde zur Vernetzung von Computern mit hohen Datendurchsätzen entwickelt. IEEE 802.15.4 dagegen ist für batteriebetriebene Kleinstsysteme zum Steuern und Monitoren ausgelegt. Des Weiteren legen Knoten in IEEE 802.15.4 Schlafperioden ein, um so lange Batterielaufzeiten zu ermöglichen. Dauerhaftes Streaming von Daten wie bei WLAN ist hier nicht vorgesehen.

Da beide Übertragungstechniken im freien ISM-Band (Industrial, Scientific and Medical [5]) bei 2,4 GHz arbeiten, ergeben sich zwangläufig Interferenzen beider Systeme. Dabei werden IEEE 802.15.4 Systeme eindeutig stärker von WLAN beeinflusst als umgekehrt. Die Hauptursache hierfür stellt die deutlich höhere Sendeleistung von WLAN von bis zu 100 mW im Gegensatz zu IEEE 802.15.4 mit 1 mW dar. In ersten Messungen wurde untersucht, ob trotz der beschriebenen Problematik ein paralleler Betrieb beider Systeme möglich ist.

Eine grundsätzliche Möglichkeit Störungen dieser Art zu minimieren, liegt darin, den Kanal bei zu schlechten Übertragungsraten ähnlich dem Verfahren des Frequenzsprungverfahren (Frequency Hopping) zu wechseln. Grundsätzlich ist dies in den beiden Standards IEEE 802.15.4 und IEEE 802.11b nicht vorgesehen. Andere Übertragungsverfahren wie z. B. Bluetooth [6] zeichnen sich dadurch aus, dass sie diese Technik nutzen, um ihren Kanal zu spreizen und somit unanfälliger gegen externe Störer zu sein. Der mit diesem Verfahren entstehende höhere Kommunikationsaufwand für die Koordination der Kanalwechsel wirkt sich negativ auf die Energiebilanz aus. Dies ist zumindest in den IEEE 802.15.4 Netzwerken nicht erstrebenswert. Sollte dennoch ein Frequenzsprungverfahren erwünscht sein, kann dies in den überliegenden Schichten durch eine geeignete dynamische Wahl der Kanäle realisiert werden.

2.1 Übersicht über 2,4 GHz Kanäle

Wie in Abbildung 1 zu erkennen ist, sind alle IEEE 802.15.4 Kanäle in Europa vollständig von den WLAN Kanälen überdeckt.

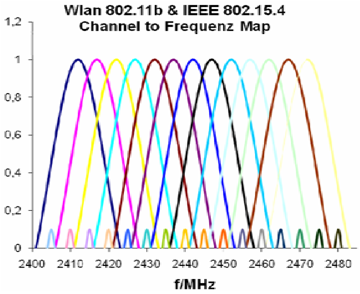


Abbildung 1: Kanalübersicht von IEEE 802.15.4 und WLAN

Für einen möglichst reibungslosen Betrieb empfiehlt der IEEE 802.11b Standard den Betrieb der drei „überlappungsfreien“

Kanäle 1, 7 und 13 in Europa und Japan und die Kanäle 1, 6 und 11 in den USA. Daraus ergibt sich die Möglichkeit, einige IEEE 802.15.4 Kanäle ohne den Einfluss von WLAN zu nutzen.

2.2 Interferenzmessungen

Zur Untersuchung der am Anfang von Kapitel 2 beschriebenen Problematik wurden zunächst Messungen in einem kontrollierten Umfeld durchgeführt. Dabei wurden sowohl eine WLAN- als auch eine IEEE 802.15.4 Funkstrecke aufgebaut.

2.2.1 Messaufbau

In Abbildung 2 ist der genaue Messaufbau mit den jeweiligen Entfernungen der einzelnen Kommunikationspartner dargestellt. Dabei wurde durch die Übertragung eines Videos eine möglichst hohe Auslastung auf der WLAN-Strecke erzeugt. Die Datenpakete der IEEE 802.15.4 Schnittstelle hatten eine Länge von 50 Bytes und tauschten im Roundtrip-Verfahren diese in einem zeitlichen Abstand von einer Sekunde fortlaufend aus.

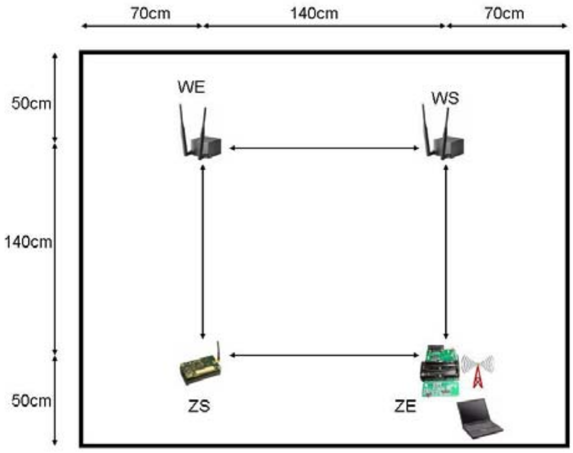


Abbildung 2: Messaufbau zu Interferenzmessungen [12]

2.2.2 Messergebnisse

Folgend werden einige der Messergebnisse vorgestellt. Zunächst wurden Messungen von völlig überlappenden IEEE 802.11b und IEEE 802.15.4 Kanälen durchgeführt. Die IEEE 802.15.4 Funkstrecke wurde dabei sehr stark von der WLAN Strecke beeinflusst. In der Tabelle 1 sind die Messergebnisse dargestellt, diese beziehen sich jeweils auf die Übertragung von 500 Paketen.

Tabelle 1: Messergebnisse WLAN und IEEE 802.15.4

WLAN Kanal 7 und IEEE802.15.4 Kanal 18	
Empfangen	2%
Verloren	98%

Tabelle 2: Messergebnisse WLAN und IEEE 802.15.4

WLAN Kanal 7 und IEEE 802.15.4 Kanal 16	
Empfangen	99%
Verloren	1%

Bei den folgenden Messungen wurden die IEEE 802.15.4 und WLAN Kanäle in ihrer Sendefrequenz immer weiter voneinander entfernt. Dabei wurde festgestellt, dass auch bei einer noch geringen Überlappung der Kanäle eine relativ sichere Kommunikation in einem IEEE 802.15.4 Netzwerk möglich ist (siehe auch Tabelle 2 und Abbildung 3)

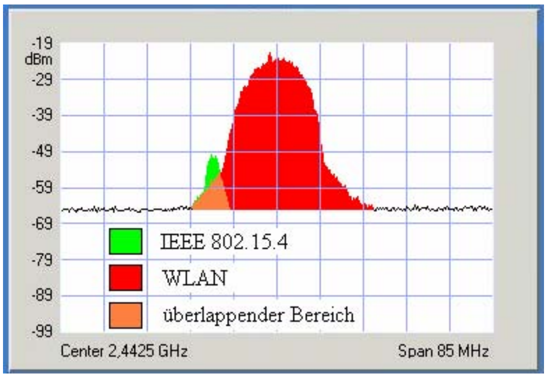


Abbildung 3: Aufnahmen des Spektrum-Analyzers für WLAN und IEEE802.15.4 [7]

3. AUFBAU

In diesem Kapitel soll kurz auf den Aufbau des Systems zum Sniffing von IEEE 802.15.4 Netzen eingegangen werden. Mögliche Lösungsansätze zur Vermeidung der oben beschriebenen Problematik der Interferenzen und zum Aufbau eines ad-hoc Steuerungs- und Kontrollnetzes auf Basis von WLAN sind ebenfalls enthalten.

3.1 Lösung der Interferenzproblematik

Um das Problem der Frequenzüberlagerung zu vermeiden, sind dynamische Wechsel der WLAN-Kanäle entsprechend des aktuell gewählten IEEE 802.15.4 Kanals erforderlich. Hierfür wird mit der Auswahl eines IEEE 802.15.4 Kanals ein entgegengesetzt liegender WLAN-Kanal ausgewählt. Dadurch liegen die Frequenzen beider Übertragungstechniken weit genug auseinander, um Interferenzen zu vermeiden. Der parallele Betrieb des WLAN Steuerungsnetzes und des vom Sensornetzwerk genutzen IEEE 802.15.4 Netzes ist somit gewährleistet.

3.2 Programm

Das entwickelte System besteht aus zwei Programmteilen bzw. Komponenten. Ein Teil (Mesh Cube[2], s. 4.3) ist für den Funkverkehr zuständig. Dieser ist für das Empfangen der Pakete des Sensornetzes durch ein angeschlossenes IEEE 802.15.4 Modem verantwortlich (siehe hierzu 4.2) und ermöglicht das Weiterleiten der ermittelten Daten über WLAN. Der zweite Teil stellt das Endgerät dar, das für die Analyse und Visualisierung der Daten durch eine Benutzerschnittstelle zuständig ist.

Alle vom Mesh Cube empfangenen Pakete werden in der Benutzerschnittstelle analysiert, in einem Packet Store gespeichert und je nach Einstellung in verschiedenen View-Areas visualisiert. Gleichzeitig besteht die Möglichkeit, über einen Control-Bereich Pakete über den Mesh Cube in das Sensornetz einzubringen, um regulativ in das Netzwerk einzugreifen.

Reicht die räumliche Verteilung des Sensornetzes über den Bereich eines Mesh Cubes hinaus, besteht die Möglichkeit, mehrere Mesh Cubes in die Umgebung des Sensornetzwerks einzubringen und so die gesamte räumliche Ausdehnung des

Netzwerks abzudecken. In einem solchen Fall bilden die Systeme untereinander ein Netzwerk, bei dem an einer zentralen Stelle auf die vollständige Umgebung aller eingesetzten Mesh Cubes zugegriffen werden kann.

4. AUSARBEITUNGEN

In diesem Kapitel wird die grafische Benutzerschnittstelle des Systems kurz dargestellt.

4.1 Benutzerschnittstelle

Innerhalb der Benutzerschnittstelle werden alle im Sensornetzwerk relevanten Daten visualisiert. Diese sind in 5 Bereiche (siehe auch Abbildung 4) aufgeteilt:

1. PAN-View:

   Die PAN-View bildet den zentralen Teil der Benutzerschnittstelle. Hier werden alle Sensorknoten, unabhängig vom verwendeten Protokoll, durch typspezifische Icons und Verbindungen zwischen einzelnen Knoten durch Kanten dargestellt. Informationen zu einzelnen Knoten können hier zusätzlich abgerufen werden.

2. PAN-List

   In der PAN-List werden alle im Umfeld erkannten PANs aufgelistet. Gleichzeitig wird für jedes PAN der Protokolltyp der enthaltenen Sensorknoten gespeichert und für spätere Analysen zur Verfügung gestellt.

   Des Weiteren dient die Liste als Filter für den PAN-View. Die zu einem PAN gehörigen Nodes können hier durch Markierung aus der View ausgeblendet werden.

3. Packet-List

   Alle im Betrieb gesnifften Pakete werden in einer Packet-List aufgeführt. An dieser Stelle führt das Programm eine Voranalyse zum verwendeten Protokoll durch und listet abhängig vom erkannten Protokoll Angaben zum Ursprung, Ziel, Typ und Inhalt des Pakets auf. Durch Auswahl eines gewünschten Pakets, wird dieses in den Packet-Details-Bereich übernommen.

4. Packet-Details

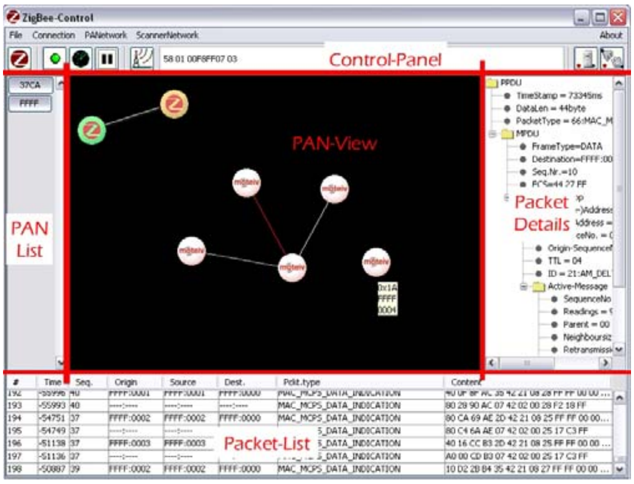
   Gewählte Pakete werden in diesem Bereich gemäß dem verwendeten Protokoll detailliert aufgeschlüsselt. Der komplette Inhalt wird den einzelnen OSI-Layern entsprechend aufgeteilt, und die in den Bytes enthaltenen Codes als Klartext ausgegeben.

5. Control-Panel

   Das Control-Panel bildet den zweiten Hauptteil der Benutzerschnittstelle. Dem Benutzer bietet sich hier die Möglichkeit, das Sniffing ein- oder auszuschalten, durch Versenden eigener Pakete in das Sensornetz einzugreifen oder das System in ein Netzwerk aus mehreren Mesh Cubes zum verteilten Sniffen zu integrieren.

Die Benutzerschnittstelle bietet dem User einen zentralen Anlaufpunkt, um das umgebende Sensornetz zu analysieren, zu visualisieren und zu managen. Die Struktur eines Netzes kann von hier aus beeinflusst und umkonfiguriert werden. Eventuell vorhandene Fehler können somit durch aktives Eingreifen des Benutzers oder automatisiert durch einen Algorithmus behoben werden.

Weiterhin bietet die Applikation die Möglichkeit, als zentraler Server eines verteilten Sniffing-Netzwerks zu fungieren und/oder sich als Client bei einem anderen Server anzumelden. In letzterem Fall werden alle analysierten Strukturen und empfangenen Pakete an den Server weitergeleitet, der dann mit den Informationen mehrerer Clients ein Netzwerk in seinem vollen Umfang darstellen kann.



#	Time	Seq.	Origin	Source	Dest.	Pckt.type	Content
193	-55993	40	----:----	----:----	----:----	MAC_MCPS_DATA_INDICATION	80 28 90 AC 07 42 02 00 28 F2 18 FF
194	-54751	37	FFFF:0002	FFFF:0002	FFFF:0000	MAC_MCPS_DATA_INDICATION	80 CA 69 AE 2D 42 21 08 25 FF FF 00 00...
195	-54749	37	----:----	----:----		S_DATA_INDICATION	80 C4 6A AE 07 42 02 00 25 17 C3 FF
196	-51138	37	FFFF:0003	FFFF:0003		S_DATA_INDICATION	40 16 CC B3 2D 42 21 08 25 FF FF 00 00 ...
197	-51136	37	----:----	----:----		S_DATA_INDICATION	A0 00 CD B3 07 42 02 00 25 17 C3 FF
198	-50887	39	FFFF:0002	FFFF:0002	FFFF:0000	MAC_MCPS_DATA_INDICATION	10 D2 28 B4 35 42 21 08 27 FF FF 00 00 ...

Abbildung 4: Benutzerschnittstelle

4.2 Sensorknoten

Als Kommunikationsschnittstelle zwischen dem Mesh Cube und dem Sensornetz kommt ein IEEE 802.15.4 USB-Dongle (CompXS, [9]) zum Einsatz. Dieser kann an den Mesh Cube über eine USB-Schnittstelle angeschlossen werden und diesen somit in die Lage versetzten, auf einem frei wählbaren IEEE 802.15.4 Kanal die dort gesendeten Pakete zu empfangen und an das verarbeitende System weiterzuleiten. Weiterhin können vom System generierte Pakete entgegengenommen werden und über den gewählten Kanal versendet werden.

Das Protokoll der verwendeten Sensorknoten spielt hier keine entscheidende Rolle, da das System so programmiert wurde, dass anhand der empfangenen Pakete ein Protokolltyp für das entsprechende PAN eingestellt wird und anhand diesem empfangende Pakete analysiert und zu sendende Pakete generiert werden können.

Unterstützt werden bisher die Protokolle von ZigBee [10], das TinyOS Message-Format [11] und Formate von 6lowpan [13]. Andere Protokolle sind über ein modular aufgebautes System problemlos integrierbar. Hierzu muss lediglich eine Adapterklasse für das neue Protokoll ergänzt werden.

4.3 Mesh Cube

Der Mesh Cube besteht aus zwei Komponenten. Zum einen enthält er eine USB-Schnittstelle, an die der in Abschnitt 4.2 beschriebene IEEE 802.15.4-Dongle, der zum Empfangen und Senden von Paketen dient, angeschlossen werden kann. Zum anderen beinhaltet er eine WLAN-Schnittstelle, an der sich ein mobiler Client anmelden kann. Dieser erhält über den Mesh Cube eine Zugriffsmöglichkeit über Port 80 auf eine Benutzerschnittstelle. Diese Schnittstelle bietet die Möglichkeit, auf das komplette Sensornetz zuzugreifen und dieses zu analysieren und zu manipulieren.

Als Betriebssystem kommt ein frei verfügbares Linux zum Einsatz, welches an eigene Besonderheiten angepasst werden kann. Ein Batteriebetrieb ist mit einem vertretbaren Aufwand über ein Zeitraum von bis zu einer Stunde möglich. Selbstverständlich kann der Mesh Cube auch über das Stromnetz versorgt werden.

Abbildung 5: Mesh Cube der Firma 4G Systems

Eine Besonderheit der Mesh Cubes ist die Möglichkeit eines sehr flexiblen Aufbaus eines WLAN-Netzes. Die hieraus entstehenden Vorteile können zur Vermeidung von Interferenzen mit IEEE 802.15.4 Netzen voll genutzt werden. So ist das System darauf ausgelegt, den Frequenzunterschied zwischen dem verwendeten WLAN und dem IEEE 802.15.4 Kanal so groß wie möglich zu wählen und zu halten.

Des Weiteren kann bei entsprechend guter WLAN-Verbindung die Sendeleistung auf ein Minimum reduziert werden, wodurch eine noch sicherere Kommunikation ermöglicht wird.

5. SCHLUSSFOLGERUNG

In diesem Artikel wurde eine Möglichkeit des verteilten Sniffens von IEEE 802.15.4 Netzen unter Zuhilfenahme eines WLAN Ad-hoc Netzwerks vorgestellt, das die Visualisierung von Netzwerken verschiedenster Protokolle ermöglicht und somit vor allem für den Servicebereich von Sensornetzwerken in Zukunft von Interesse sein könnte.

Des Weiteren kann durch die entwickelte Architektur auch eine Verlängerung der Reichweiten durch WLAN positiv für die Sensornetzwerke genutzt werden. Wie gezeigt werden konnte, ist der parallele Betrieb von WLAN und IEEE 802.15.4 Netzwerken bei einem intelligenten Management der Kanäle ohne Interferenzen möglich.

Einen zusätzlichen Vorteil stellt die Bandbreite des WLAN dar. So könnte z. B. ein Sniffing Device als Datensenke für mehrere Teilnetze dienen und die Daten an andere Komponenten über die von WLAN zur Verfügung gestellte Bandbreite weiterleiten.

6. REFERENZEN

[1] Prof. Gober, P. und Thiem, L. Making your smart home with ZigBee. IPQC Workshop Wireless Sensornetworks, Nov 2005

[2] MeshCube Homepage, 4G Systeme GmbH, http://www.meshcube.org, Mai 2007

[3] IEEE 802.15.4.4 – Homepage, The Institute of Electrical and Electronics Engineers, Inc., Februar 2007, http://www.ieee802.org/15/pub/TG4.html IEEE 802.11b Standard

[4] IEEE 802.11TM WIRELESS LOCAL AREA NETWORKS Homepage, Institute of Electrical and Electronics Engineers, Inc., http://grouper.ieee.org/groups/802/11/, 2007

[5] Bundesnetzagentur, Allgemeinzuteilung von Frequenzen in den Frequenzteilbereichen gemäß Frequenzbereichszuweisungsplanverordnung (FreqBZPV), Vfg 76 / 2003

[6] The Official Membership Site – Homepage, Bluetooth® SIG Inc., Februar 2007, http://www.bluetooth.org

[7] Eisenreich, A., Thiem, L., Koexistenz nicht gewährleistet, funkschau 25/2006, WEKA Fachzeitschriften-Verlag GmbH

[8] Shin, S. Y., Park, H. S., Choi, S. und Kwon, H., Packet Error Rate Analysis of IEEE 802.15.4 under IEEE 802.11b Interference, Workshop. on Wired/Wireless Internet Communications, April, 2005, Phoenix, Arizona, USA

[9] CompXS Homepage, Integration Associates, Inc., http://www.integration.com , 2007

[10] ZigBee Homepage, Zigbee Alliance, http://www.zigbee.org, 2007

[11] TinyOS Homepage, UC Berkeley, http://www.tinyos.net, 2007

[12] Eisenreich, A., Interoperabilität zwischen ZigBee und WLAN im 2,4 GHz ISM-Band, Diplomarbeit TFH Berlin, August 2006

[13] 6lowpan Homepage, Ajou University, http://6lowpan.net/, 2007

Smart Composition of Sensor Network Applications

Stefan Schmitz, Olaf Landsiedel, Klaus Wehrle
RWTH Aachen University, Distributed Systems Group
Aachen, Germany
stefan.schmitz2@rwth-aachen.de, [olaf.landsiedel, klaus.wehrle]@cs.rwth-aachen.de

ABSTRACT

Sensor Networks have received much attention in Computer Networks research in the recent years. They are small, long living, need no infrastructure and can communicate with each other. Therefore actual projects using sensor networks cover a wide range of applications. In architecture they are used to monitor buldings or bridges or they help observing natural environments in biology. Most of the intended operators of sensor network deployments are without computer science background. Thus, it is quite hard for them to use the currently provided tools to develop sensor node applications. The goal of this research project is to provide a intuitive tool which supports the user to create sensor network applications. Based on the application needs the user can compose the components provided by the sensor node operating system. Afterwards the tool can create a source code framework automatically which the user can fill with the detailed application behaviour.

1. INTRODUCTION

TinyOS 2.x [6] is an operating system for sensor nodes. Programs for sensor nodes must be very limited in their use of memory and computation time (*i.e. power consumption*). They must be stable running for a long period (about years) of time. Therefore, TinyOS is not programmed in a common computer language like *ANSI C* but in a extension to it called *nesC*[2]. With the extensions a programmer can – amongst others – implement modular and structured applications with strictly defined interfaces between each module. That is a good entry point to meet sensor nodes hardware and programming restrictions. The TinyOS operation system itself gives a lot of predefined and hierarchical structured [5] modules to an application designer. The modules (called *components* in TinyOS) *provide* or *use* specific functionality of other components. A component can expose such a specific functionality via an abstract *interface*. Interfaces define *events* that are a kind of callback functions which a using component must implement and *commands* are functions that initiating arbitrary calculation like common programming languages functions or method calls will do. A component in TinyOS is either a *configuration* or *module*. A configuration specifies a certain (part of an) application. It groups other components and defines the so called *wiring* between components. A wiring defines which interfaces connect two components. Finally the modules hold the "functionality" of a program, such as the working code like calculations and calls to commands of other interfaces.

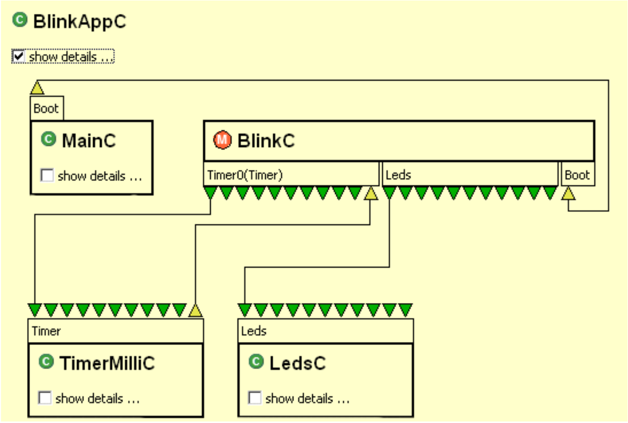


Figure 1: Diagram of the *Blink* application

Figure 1 shows a graphical represetation of a typical sample application *Blink* which consists of four components in detail: one module (BlinkC) and three configurations (MainC, LedC, TimerMilliC). The interfaces are depicted as rectangles with small triangles on the outside edge. The triangles are symbols for an event (triangle point outside) or a command (points inside) of an interface. The interfaces itself are at top or bottom of a component depending if they are used or provided. The arrows between the interfaces are showing the actual wiring for the BlinkAppC configuration. This graphical representation is based on the typical representation of TinyOS components [3].

2. MOTIVATION

As discussed *nesC* gives advantage to the experienced developer to implement stable and small applications but it is quite hard to get used to *nesC*. Especially people with no computer science background like biologists or others that want to implement sensor nodes can get into some trouble. It is hard to learn the concepts and much *code browsing* to find all required information even for the simplest application is needed. Thus, the idea was to give such newbies but also experienced developers a guidiance in developing sensor node applications. To demonstrate how the information is spread over many files listing 1 gives a short overview of involved files and their source code.

A graphical user interface is very helpful for development because one basic design task is to group different components and to plug them together. On the one side such tool can display information which is spread over many files in a compact and well arranged way. On the other side the user can compose applications, do manipulations and define configurations according to his needs. A smart interface can guide his actions and avoid errors. For example, it can restrict certain actions such as connecting (wiring) interfaces that match in type. Current available graphical tools and application for development help the user while browsing the application and system code and do syntax highlighting of source codes, but do not offer generation of applications in a graphical way [7].

After such a restricted and guided design step the tool generates source code. In this phase a smart implementation can provide additional benefits. For example, if a module uses the interface `Timer` it must implement a `Timer.fired` event (that is the callback method for a started timer). All this can cause compilation errors and interfere with the development. However it could be avoided by knowing the structural context of an application.

```
/* BlinkAppC.nc */
configuration BlinkAppC { }
implementation {
  components MainC, BlinkC, LedsC;
  components new TimerMilliC() as Timer0;
  BlinkC.Boot -> MainC;
  BlinkC.Timer0 -> Timer0;
  BlinkC.Leds -> LedsC;}
/* BlinkC.nc */
module BlinkC {
  uses interface Timer<TMilli> as Timer0;
  uses interface Leds;
  uses interface Boot;
} implementation {
  event void Boot.booted() {
    call Timer0.startPeriodic( 250 ); }
  event void Timer0.fired() {
    call Leds.led0Toggle(); } }
/* Timer.nc */
interface Timer {
  command void startPeriodic(uint32_t dt);
  event void fired(); /* ... */ }
```

Listing 1: Code for sample application Blink ***only main parts***

3. EDITOR DESIGN

The goal of this research project is to implement a tool which gives advanced support and guidance to a sensor node application designer. It must integrate into the existing toolchain of the TinyOS environment to provide up-to-dateness with the constant development in that research topic. Additionally, it must produce an error-free source framework with all information a user can graphically express. Figure 2 shows an actual screenshot of the current version of the editor developed in this project.

3.1 Tool Architecture

The architecture of the editor is quite straight-forward. At first it parses the TinyOS environment to generate abstract syntax trees (AST) of the existing components and interfaces in the TinyOS system folder. This is done only once (or if the user demands after an update). The created library of components is provided to the user as basic building blocks of his new design. After that the graphical editing process begins. The user can create an own configuration as top level part for his new application. From a toolbar he can add some of the components or new modules into his configuration and then wire them up. When he finished his work he can generate the source code files for his design. The files contain all necessary parts and the user just has to fill the missing module implemetation parts in the newly created modules. After this has been done the compilation can be initiated and the new sensor node application can be loaded to the nodes.

The graphical user interface must be stable running and user-friendly. Therefore, we decided to implement it in the Eclipse Environment [1]. Eclipse provides a generic Integrated Development Environment (IDE) with a quite common feature set like menues, toolbars and views and some new features like workspaces, plugins and perspectives which enhance the design and implementation process. Eclipse is highly customizable and gives developers many entry point for new plugins and features.

To implement the graphical part of the editor the Eclipse Project *Graphical Editor Framework* (GEF) is used [4]. It provides a framework designed for graphical editors based on generic models. Much features like user feedback, generic layouting, zooming and toolbars give powerful techniques to an editor developer. The GEF is strictly designed using the Model View Controller paradigm. All these help an experienced developer to create new fetures very fast and stable. The user interfaces created with GEF are additionaly very intuitive to handle for users.

3.2 Workflow Example

To demonstrate the ease of use of such an editor we describe a typical design process in this section (see figure 3). The sensor node application designer starts the eclipse environment and selects to create a new TinyOSDiagram via a creation wizard. Doing so changes automatically to the TinyOS Perspective which changes the layout of the Eclipse IDE to the needs of the user. The Editor changes to an empty graphical workspace with a toolbar for creation tools and libray components. An outline view shows the structure of the current design. At bottom a view shows actual source code. The user can create a new configuration at the empty workspace by just selecting the “New Configuration” tool at the toolbar (figure 3(a)). He can rename it e.g. `BlinkAppC` (figure 3(b)) and then add a `MainC`, a `TimerMilliC` and a `LedsC` component from the toolbar (figure 3(d)). To add functionality he then creates a new module inside the `BlinkAppC` called `BlinkC`. At this point he has only to do some wiring to make the components work together. He can easily drag e.g. the `Timer` interface from the `TimerMilliC` component to the `BlinkC` module (figure 3(e)). He repeats this step and wires all interfaces with the `BlinkC` module. Now he can generate the source code and gets the complete source code file `BlinkAppC.nc` like in listing 1. This source code is complete and must not be changed anymore by the user. All information was collected within the design process. Additionaly a skeleton similar to `BlinkC.nc` (the implementation of the functions are empty) wil be created. There the user has just to fill in what to do after `booted` and when the timer is `fired`.

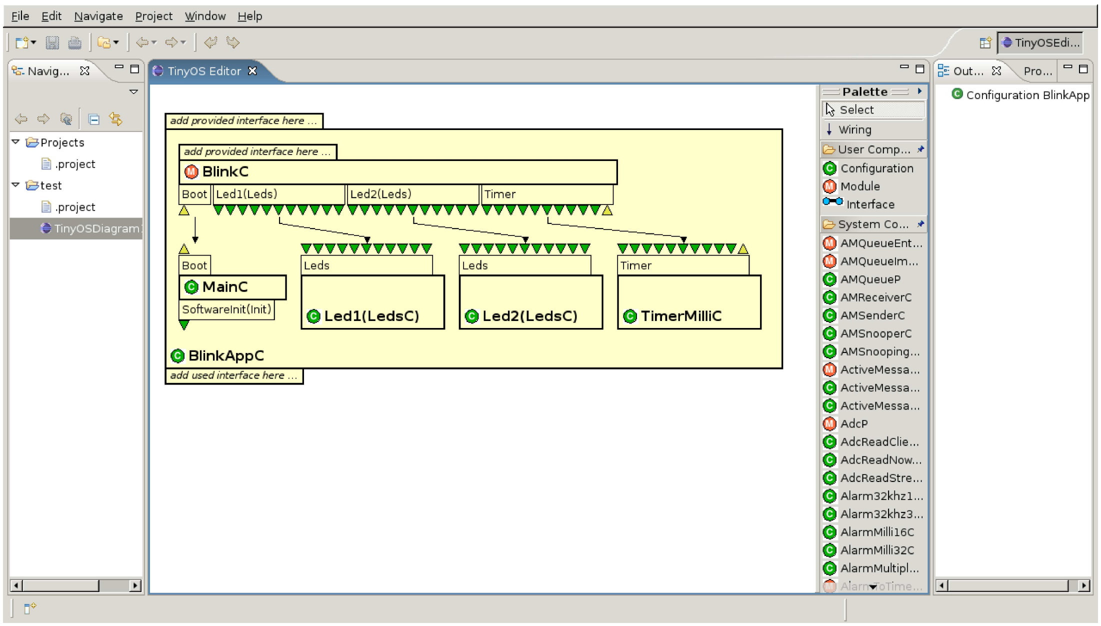


Figure 2: Screenshot of application

4. CONCLUSIONS

This paper presented a tool with which users can intuitivly design sensor network applications. It is integrated as plugin into the Eclipse Integrated Development Environment. Additionally, a framework for graphical editing is used. First the tool analyses the TinyOS environment to collect all components and interfaces the system provides. The plugin shows the available components to the developer. The user can plug these components togehter and wire them up in an intuitive way. After this design step he can generate an apropriate code framework as starting point for the implementation of the detailed application behavior.

A smart user interface can strongly improve the design process of sensor node applications. The user can implement simple programs in the manner of *wake up*, *sense*, *anaylze data*, *sleep* very fast and without writing much code. This gives especially new and inexperienced designers an easier entry point for sensor node applications and can enlarge the audience for this actual research topic. Experienced developers can also get benefit by avoiding errors at design time like forgetting to implement events or type mismatch of wired interfaces. Additionally, they can keep an overview of bigger projects by making use of the browsing capabilities.

The current implementation allows to create new and browse existing components and see its included components (in case of a configuration), provided and used interfaces and wirings in its context. Thus, there is a parser for *nesC* source code. The user can add interfaces by dragging of an existing interface to the target component with automatic wiring or add new interfaces to an component and wire them up manually. Next steps are code generation and optimizations of the user interface.

5. REFERENCES

[1] Eclipse. www.eclipse.org.

[2] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesc language: A holistic approach to networked embedded systems. *Proceedings of Programming Language Design and Implementation (PLDI)*, June 2003.

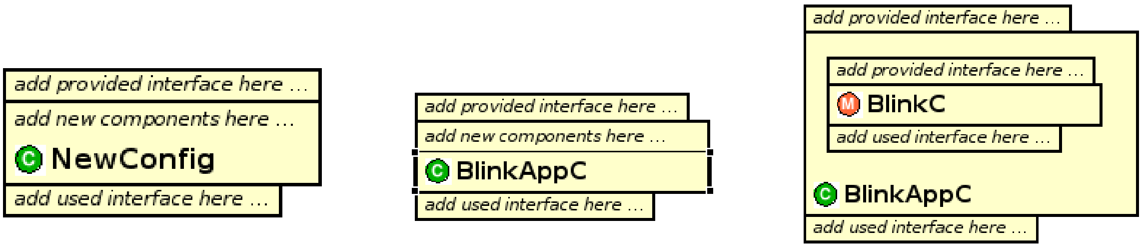
[3] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *Proceedings of Programming Language Design and Implementation (PLDI'03)*, June 2003.

[4] Graphical editor framework. www.eclipse.org/gef.

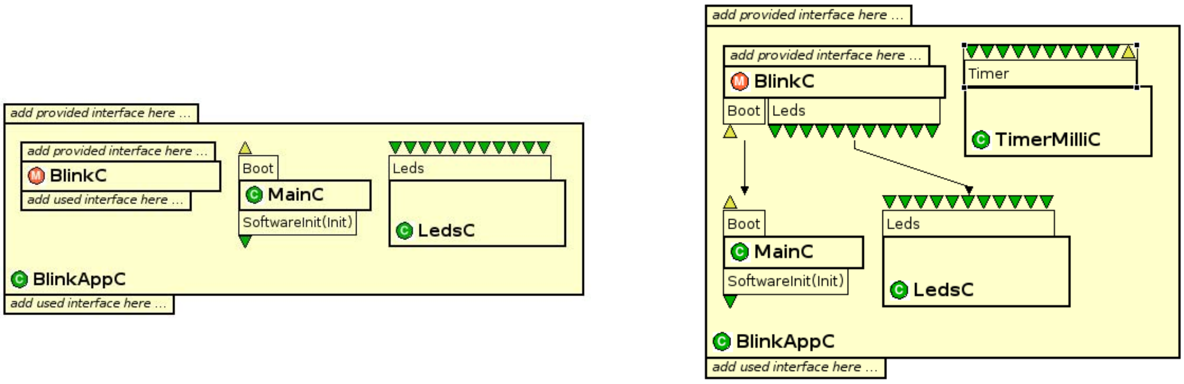
[5] V. Handziski, J.Polastre, J.H.Hauer, C.Sharp, A.Wolisz, and D.Culler. Flexible hardware abstraction for wireless sensor networks. *Proceedings of the 2nd European Workshop on Wireless Sensor Networks (EWSN 2005)*, January 2005.

[6] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. Brewer, and D. Culler. The emergence of networking abstractions and techniques in tinyos. *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI 2004)*, March 2004.

[7] Tinyos plugin for eclipse. dcg.ethz.ch/projects/tos_ide.



(a) Begin of design (b) Renaming configuration (c) Adding main user module



(d) Adding additional components (e) Wiring up components

Figure 3: Example sensor node application design steps.

Geographic Routing in 3D

Matthias Witt and Volker Turau
Hamburg University of Technology
Institute of Telematics
Schwarzenbergstr. 95, 21073 Hamburg, Germany
{matthias.witt,turau}@tuhh.de

ABSTRACT

Existing geographic routing algorithms assume a two-dimensional topology. Dedicated wireless sensor network scenarios demand for algorithms that operate in three-dimensional environments. This paper discusses issues which arise when making the step from 2D to 3D. Simulation studies show that 3D routing is less efficient than its 2D counterpart when comparing topologies with the same average node degree.

1. INTRODUCTION

Geographic routing algorithms forward messages using location information rather than node addresses. Each node is aware of its position via GPS or some localization algorithm. The location of the destination is included in the message, and the nodes route the message to the destination using their location information. This is especially relevant for wireless sensor networks, where the network topology is unknown a priori and subject to change, radio communication is unreliable, and position is more important than node IDs.

Existing geographic routing algorithms assume Euclidean two-dimensional topologies. Location information is held as x and y coordinates. This is sufficient when the network is deployed in a plane, e. g., for environmental monitoring in a large area. In some cases, however, the network can become three-dimensional. Application scenarios include networks within buildings, underwater networks, or even networks in space. Especially underwater sensor networks gained research interest recently [2, 8]. To enable 3D routing, the existing algorithms have to be extended. However, it is not sufficient to just add the z coordinate. This paper describes the difficulties that arise when making the step from 2D to 3D with a special focus on the Blind Geographic Routing (BGR) algorithm, which proved to come with very small communication overhead and be robust against failures, location errors, and radio irregularity.

Current research on three-dimensional networks is focused on connectivity and coverage [1, 3, 9]. Little work has been published that addresses 3D routing. In [8], two routing algorithms for underwater networks are proposed based on link metrics. The first algorithm, which is delay-insensitive, only selects nodes that are closer to the destination, and hence is subject to fail for sparse networks. The second algorithm is delay-sensitive, but uses a centralized approach. A heuristic variant of face routing, which does not guarantee delivery, but has been shown to perform well in simulations, is proposed in [4]. Some challenges in designing 3D algorithms are presented in [7].

This paper is organized as follows: Section 2 provides a short overview of the BGR algorithm. The problems when making the step toward 3D are discussed in Section 3. Simulation results for 2D and 3D topologies are presented in Section 4. Section 5 concludes the paper.

2. OVERVIEW OF BGR

This section provides a short overview of the Blind Geographic Routing (BGR) algorithm. For a detailed description and comparison to similar routing algorithms, see [10].

BGR is a two-dimensional beacon-less geographic routing algorithm using a broadcast-based contention scheme. The nodes do not carry any neighborhood or topology information. Packets are forwarded via broadcast. Nodes which receive this broadcast determine if they are located within a special area called *forwarding area*. A description of the forwarding area is included in the packet. The forwarding area is oriented toward the destination location, and its dimension ensures that all nodes within it can mutually communicate with each other (provided the unit disk graph model; however, BGR also performs well with more realistic, irregular radio propagation). Examples for forwarding areas are shown in Figure 1.

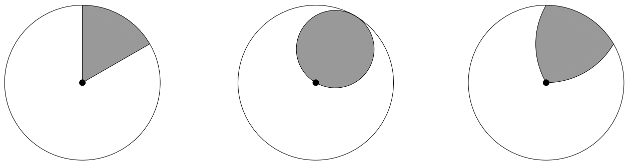


Figure 1: Forwarding areas: 60° sector, circle, and Reuleaux triangle

Nodes which receive a broadcast and are located within the forwarding area start a contention timer depending on their distance to the destination. The timer of the node which is closest to the destination expires first; this node declares itself as next hop and forwards the packet again. The other nodes which have still a timer running also receive this packet and cancel their timers.

To detect empty forwarding areas, the forwarder starts a recovery timer which is scheduled to expire after the last possible contention timer of any node within the forwarding area has expired. When the recovery timer expires, the forwarder turns the forwarding area by 60° in an arbitrary direction and broadcasts the packet again. If this forwarding area is also empty, it is turned in the other direction. When this third attempt also fails, the message is considered undeliverable and dropped. Simulation results show that this is unlikely to happen when the network density is not too low.

3. THE STEP FROM 2D TO 3D

When making the step from two to three dimensions, it is not sufficient to simply add z coordinates to location descriptions. In the following, the major issues of the transition to 3D are discussed.

Most beacon-based geographic routing algorithms perform some variant of face routing [5, 6]. The faces to be traversed are determined by the line from source to destination. However, in 3D graphs, this line does not determine the faces [4]. Thus, 2D face routing algorithms are not directly applicable to 3D. It is not known if a distributed 3D face routing algorithm exists.

Beacon-less algorithms, on the other hand, can easily be extended to operate in 3D space. The forwarding areas have to be converted into forwarding volumes by constructing the solid of revolution around the forwarder-destination axis. Hence, the 2D sector becomes a spherical sector, the circle becomes a sphere, and the Reuleaux triangle becomes the solid of revolution of a Reuleaux triangle. Note that this is different from the Reuleaux tetrahedron, whose diameter is slightly larger than the radius of the intersecting spheres from which it is constructed.

A problem with three-dimensional topologies is that more nodes are needed for network coverage than in two-dimensional topologies. For a quantitative comparison, suppose that the same average number of neighbors is to be achieved in a 2D topology of area a^2 and a 3D topology of volume a^3. The transmission range r is fixed, i. e., the unit disk graph (or, in 3D, unit sphere graph) model is assumed. If the total number of nodes in the 2D topology is n, then the average number of neighbors is $\frac{n\pi r^2}{a^2}$ (ignoring border effects). For this value to be the average number of neighbors in the 3D topology, the total number of nodes has to be

$$\frac{n\pi r^2}{a^2} \cdot \frac{a^3}{\frac{4}{3}\pi r^3} = \frac{3an}{4r}.$$

This means that in a 3D topology, the total number of nodes has to be by a factor of $\frac{3a}{4r}$ larger than in a 2D topology.

Even when the number of nodes is increased according to these calculations, a remaining problem is the additional dimension of the destination location, which leads to more possible routing directions, which result in lower delivery rates. This can easily be seen when considering the fraction of the transmission area/volume that is covered by the forwarding areas/volumes. Table 1 indicates that in 3D, the forwarding volumes cover only half as much of the transmission volume as the corresponding forwarding areas in 2D. As a consequence, the 3D version of BGR performs recovery up to four times per hop in contrast to two times in the 2D version. The first forwarding volume is obtained by turning the forwarder-destination axis by 60° in an arbitrary direction first, then in the opposite direction (mirrored about this axis), then turned by 90° about this axis, and in the last try mirrored again. A problem is that there are gaps between the turned forwarding volumes, which is not the case for the 2D forwarding areas (with the exception of the circle, which leaves two small gaps). Also, the overlapping regions are larger than in the 2D case.

Table 1: Sizes of forwarding areas/volumes as fraction of transmission area/volume

	Sector (Sph. sector)	Circle (Sphere)	Reuleaux triangle
2D	$\frac{1}{6} \approx 0.167$	$\frac{1}{4} = 0.25$	$\frac{1}{2} - \frac{\sqrt{3}}{2\pi} \approx 0.224$
3D	$\frac{1}{2} - \frac{\sqrt{3}}{4} \approx 0.067$	$\frac{1}{8} = 0.125$	$\frac{1}{2} - \frac{\pi}{8} \approx 0.107$

4. SIMULATION RESULTS

Simulation experiments have been conducted using the network simulator ns-2 to compare the performance of BGR in 2D and 3D topologies. The experiments were run with 150 nodes and a sink in the center of the topology; each node generates a data packet and sends it to the sink. The transmission range is 40 m. The average node degree was varied between 10 and 40 by adjusting the size of the (square or cubic) topology. Each value represents the average of 20 simulation runs. The Reuleaux triangle was used as forwarding area/volume.

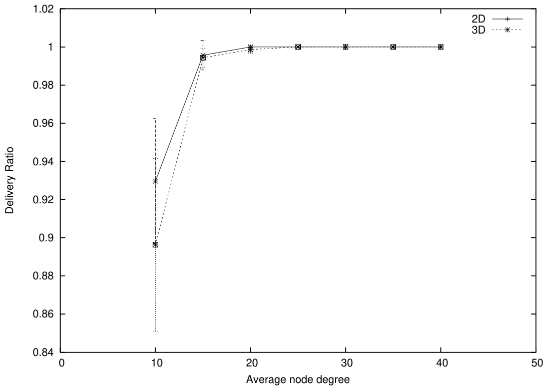


Figure 2: Delivery ratio in 2D and 3D topologies

Figure 2 shows the average delivery ratio. The error bars depict confidence intervals of 95 %. As expected, 2D routing performs better than 3D routing; the difference is most evident at low node degrees. At medium and high node degrees, however, both 2D and 3D routing perform very well and achieve constant delivery ratios of 100 %.

The total number of sent packets is depicted in Figure 3. At medium and low node degrees, 3D routing needs significantly more packets than 2D routing, which is an indicator that much more recovery is performed in 3D. The difference vanishes at high node degrees.

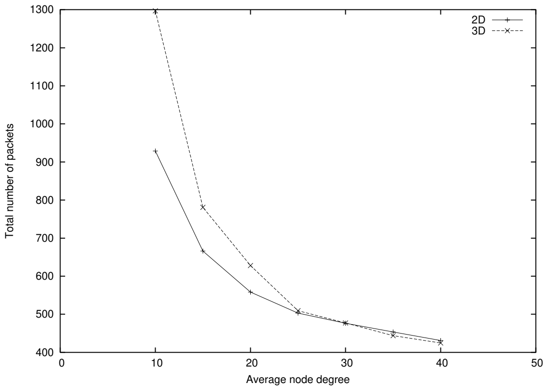


Figure 3: Number of packets in 2D and 3D topologies

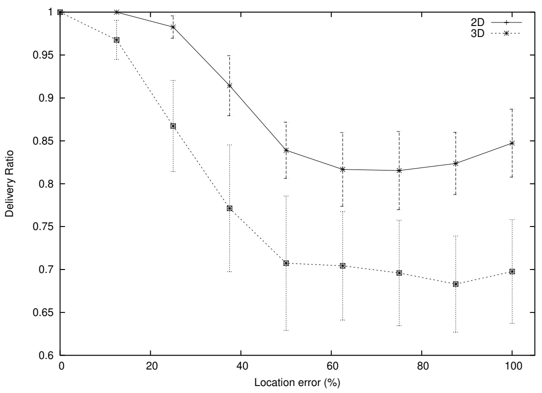


Figure 4: Delivery ratio in the presence of location errors (standard deviation as fraction of transmission range), topologies with average degree 30

As a result, the simulation studies show that the performance blockers identified in Section 3 are only relevant for low node degrees, where also 2D topologies have delivery ratios under 100 %. When the node degree is above 25, both 2D and 3D routing perform similarly well.

Location errors have a negative impact on routing performance. The impact of location errors on BGR has been studied in [11]. The error has been modeled using a two-dimensional Gaussian distribution with mean zero and standard deviation between zero and the transmission range. For 3D routing, the location error can be modeled using a three-dimensional Gaussian distribution. Figure 4 shows the corresponding simulation results, which indicate that 3D topologies are significantly more vulnerable to location errors than 2D topologies. Even small errors lead to a noticeable decrease of the delivery ratio. The major reason is that recovery is not performed when the forwarding volume is empty and the destination is assumed to be within transmission range, but due to location errors, it is not. In 2D topologies, this situation occurs less often because of the larger coverage of the the forwarding areas (cf. Table 1). Another issue is that in 3D topologies the average distance between the real and the estimated location is greater than in 2D topologies with the same standard deviation σ. In 2D, the distance follows a Rayleigh distribution, in 3D a Maxwell-Boltzmann distribution. The expected value is $\sqrt{\frac{\pi}{2}}\sigma \approx 1.253\sigma$ in 2D and $\sqrt{\frac{8}{\pi}}\sigma \approx 1.596\sigma$ in 3D.

5. CONCLUSION

Three-dimensional geographic routing is an important technique for future wireless sensor network scenarios. However, existing algorithms support merely the 2D case. The step toward 3D routing imposes some fundamental problems. Beacon-based algorithms which operate on a variant of face routing cannot simply be adopted for 3D topologies because the techniques are not directly applicable to 3D graphs. Beacon-less algorithms like BGR, on the other hand, can easily be enhanced to a 3D version by defining forwarding volumes instead of forwarding areas.

An inherent problem of 3D topologies is that more nodes are required to achieve a similar node coverage for topologies with the same average number of neighbors. BGR's forwarding volumes suffer from an analogical problem, since the covered fraction of the transmission volumes is only half as large as in the 2D case. Hence, recovery mode is triggered more often and the number of sent packets increases. The delivery ratio is also slightly lower. Additionally, simulation revealed that in case of location errors, 3D routing has significantly more problems than its 2D counterpart.

6. REFERENCES

[1] S. M. N. Alam and Z. Haas. Coverage and Connectivity in Three-Dimensional Networks. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, pages 346–357, Los Angeles, CA, USA, Sept. 2006.

[2] J. Heidemann, W. Ye, J. Wills, A. Syed, and Y. Li. Research Challenges and Applications for Underwater Sensor Networking. In *Proc. IEEE WCNC 2006*, pages 228–235, Las Vegas, NV, USA, Apr. 2006.

[3] C.-F. Huang, Y.-C. Tseng, and L.-C. Lo. The Coverage Problem in Three-Dimensional Wireless Sensor Networks. In *Proc. Global Telecommunications Conference (GLOBECOM)*, pages 3182–3186, Dallas, TX, USA, Dec. 2004.

[4] G. Kao, T. Fevens, and J. Opatrny. Position-Based Routing on 3-D Geometric Graphs in Mobile Ad Hoc Networks. In *Proc. 17th Canadian Conference on Computational Geometry (CCCG'05)*, pages 88–91, Windsor, Ontario, Canada, Aug. 2005.

[5] B. Karp and H.-T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254, Boston, MA, USA, Aug. 2000.

[6] F. Kuhn, R. Wattenhofer, Y. Zhang, and A. Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. In *Proc. 22nd ACM Symposium on Principles of Distributed Computing (PODC)*, pages 63–72, Boston, MA, USA, July 2003.

[7] S. Poduri, S. Pattem, B. Krishnamachari, and G. Sukhatme. Sensor Network Configuration and the Curse of Dimensionality. In *Proc. Third Workshop on Embedded Networked Sensors (EmNets 2006)*, Cambridge, MA, USA, May 2006.

[8] D. Pompili and T. Melodia. Three-Dimensional Routing in Underwater Acoustic Sensor Networks. In *Proc. ACM Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks*, pages 214–221, Montreal, Quebec, Canada, Oct. 2005.

[9] V. Ravelomanana. Extremal Properties of Three-Dimensional Sensor Networks with Applications. *IEEE Transactions on Mobile Computing*, 3(3):246–257, July 2004.

[10] M. Witt and V. Turau. BGR: Blind Geographic Routing for Sensor Networks. In *Proceedings of the Third International Workshop on Intelligent Solutions in Embedded Systems*, Hamburg, Germany, May 2005.

[11] M. Witt and V. Turau. The Impact of Location Errors on Geographic Routing in Sensor Networks. In *Proc. Second International Conference on Wireless and Mobile Communications (ICWMC'06)*, Bucharest, Romania, July 2006.

A Prototype Study on Hybrid Sensor-Vehicular Networks

Extended Abstract

Elias Weingärtner
RTWH Aachen
elias.weingaertner@cs.rwth-aachen.de

Frank Kargl
Ulm University
frank.kargl@uni-ulm.de

1. INTRODUCTION

In this paper we present a concept where we combine two forms of networks that both attracted a lot of research efforts recently. Both Vehicular Ad-hoc Networks (VANETs) and Wireless Sensor Networks (WSNs) are subject of ongoing research activities. However, the characteristics of VANETs and WSNs are very different.

Nodes in sensor networks are highly miniaturized, mostly static, very resource and energy constrained, and usually have good sensing capabilities. In contrast, VANETs have very dynamic topologies and the vehicles do not suffer from significant energy constraints. The vehicles could be equipped with sensors themselves. However, the sensor coverage cannot be guaranteed, as vehicles are not present everywhere and at all times and some kinds of events cannot be reliabily detected by moving entities.

We introduce the new concept of Hybrid Sensor-Vehicular Networks so that both network types can benefit from the strengths of each other while compensating the weaknesses. We use a Wireless Sensor Network deployed in or near roads as a sensor grid with constant availability and dense coverage in contrast to the vehicle-to-vehicle network which might have only sparse coverage. The sensor network constantly communicates its sensor data to the vehicles driving on the road, delivering them with accurate and up-to-date sensor information. Vehicles communicate to disseminate this information to over comparatively long distances. There, the vehicles deliver this data back the sensor network where it is stored for future retrieval by other vehicles. This relieves the sensor network from the energy-consuming task to transfer the data hop-by-hop inside the WSN itself.

Such Hybrid Sensor-Vehicular Networks are suited for all applications where a stationary WSN collects sensor data that is disseminated only on a small scale within the WSN, then delivered to vehicles which transfer it to other regions by multi-hop routing or vehicle movement and either hand it off to interested vehicles or to remote WSN nodes that store the data and deliver it to approaching cars on certain conditions.

We use an example for further explaining this concept. Assume we have a road segment as shown in Figure 1. The application implements a dangerous road condition warning, where drivers are warned about potentially dangerous road conditions like e.g. icy road. First, a WSN node detects ice on the road and shares this information with neighboring motes (①) inside a small region.

When a vehicle A enters this region, the WSN triggers a vehicle-present event and the information is transmitted to the car (②). This way, the driver of vehicle A would receive a short-term warning and can react accordingly. Vehicle A forwards the information via the long-range VANET to vehicle B (③). A relevance function determines that vehicle B is in a good position to feed the information back to the WSN for further availability. As the road splits at position B, the icy road information would otherwise become unavailable to vehicles approaching from the lower road. So B transfers the information back to the WSN (④) and the nodes distribute it in a small neighbourship for redundancy purposes (⑤). A and B leave this road segment (⑥) and are out of communication range, when vehicle C approaches the intersection and still receives the warning information from the WSN (⑦). The vehicle displays a warning to the driver who has plenty of time to adapt his driving style to the approaching danger.

There are two important observations here:

1. Vehicle C is never in direct reach of the other cars. So a vehicular sensor network composed only of the vehicles would never deliver the warning to vehicle C.
2. There is a significantly lower number of mote transmissions compared to the case where the motes try to deliver the information hop-by-hop from the source to the position of vehicle C. A lot of mote energy is saved and the lifetime of the WSN nodes is prolonged.

2. INFORMATION DISTRIBUTION

Within Hybrid Sensor-Vehicular scenarios, five different ways of information flow can be distinguished:

1. Information flow within the Wireless Sensor Network
2. Data transition from WSN to VANET
3. Dissemination within the VANET
4. Information injection from VANET to WSN
5. "Physical" data transport by moving vehicles

Each information distribution method faces specific challenges, which we are going to point out in the following sections.

2.1 Distribution inside the WSN

When events are captured by the wireless sensor network these events are about to be reported to mobile nodes. A central idea in our perspective on such scenarios is that the

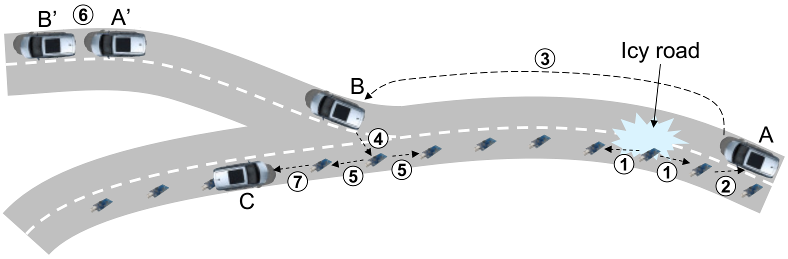


Figure 1: Example scenario for a Hybrid Sensor-Vehicular application

WSN and the VANET are coupled directly, and hence, certain wireless sensor nodes that transmit WSN data to vehicles have to be chosen. Those **gateways** are distinct from the **reporting nodes** which are mainly responsible for event detection and multi-hop routing.

As we rely on information transport using a VANET, the sensors do not form one huge wireless sensor network. Instead, we propose a dynamic decomposition of the sensor nodes in many small to mid-size sensor networks in which data is reported to gateway sensors. Further spatial coverage of events is reached using VANET message dissemination.

After an event has been detected or sensor data has been collected, this information needs to be reported to a gateway sensor which can be realized by standard WSN source-to-sink techniques like spanning trees. To equalize energy consumption, the gateway role will change periodically.

2.2 Data transition from WSN to VANET

As illustrated, gateway sensors interact directly with mobile nodes, and basically two questions are of particular interest according to this way of information flow. First of all, one might ask *when information is to be sent to the vehicle.* Instead of a periodic transmission, we propose a triggered scheme: Once a new vehicle is present, data should be transmitted - and therefore, vehicle presence can be regarded as a special type of an event detected by special sensors.

Once triggered, the actual transmission from sensors to mobile nodes is a time-crucial task: As sensor nodes have a limited transmission range, the data needs to be sent within a very short time. Vehicles may move with relative speeds up to 70 m/s, and as a delay between vehicle detection and data transmission exists, the time frame left may be lower than a second. Hence, all data being transmitted must be fitted compactly into one or very few frames in order to improve the probability of the transmission to succeed.

2.3 VANET Message Dissemination

The main task of the Vehicular Network in our approach is to disseminate messages picked up from a local gateway sensor. The primary goal is of course to inform approaching vehicles about a potential hazard that was measured by a local WSN. However, for the Hybrid Sensor-Vehicular Network, the purpose of the vehicular part is also transporting messages to remote locations for re-injection into a WSN. This way, information becomes time stable and does not rely on the presence of vehicles.

Geocast [1] is a well-known primitive in the VANET domain that is suitable for this kind of information dissemination.

2.4 Information Injection

Once information has been distributed over the VANET, it can be stored back from the vehicle to the wireless sensor network. While at first glance this might look like a unnecessary gimmick, we argue that this mechanism is a key feature of Hybrid Sensor-Vehicular information distribution:

It might happen that the VANET looses connectivity if vehicles move out of range. In this case, messages cannot be distributed to other vehicles using the VANET itself, although cars approaching the current node's position later in time might be interested in this information, for example if it is a warning notification. In this case, the warning message can be stored to the WSN where it is kept until other vehicles pass by and retrieve this information again. Similarly, this mechanism could be further utilized for other tasks like gateway notifications.

2.5 Physical data transport

Besides the vehicles' interconnection, their spatial movement can be utilized for data dissemination as well. If vehicle density is sparse, data sampled from the wireless sensor network will be cached by vehicles. Based on the relevance function's results, the information is injected back to wireless sensor network.

3. PROTOTYPE ARCHITECTURE

One approach to investigate the characteristics and issues that arise in Hybrid Sensor-Vehicular Networks would be a complex simulation that incorporates realistic models of WSN data propagation, traffic flow, and an appropriate simulation of VANET message dissemination. Combining those modules into one simulation is challenging. In addition, it is questionable, how meaningful such results would be, as integration of multiple radio systems like ZigBee and IEEE 802.11 within one single simulation is not well understood.

Therefore, we decided to create a prototype architecture as starting point for future work. The goal behind the prototype is to prove that all five ways of information flow can be realized. Furthermore, the prototype allows us to spot out more interesting issues and questions related to the new field of Hybrid Sensor-Vehicular Networks. Within this section, a brief overview over the prototype and its architecture is given.

Figure 2 depicts the basic architecture of the prototype, which consists of two subsystems, a *Sensor Network Subsystem* running on motes and a *Mobile Node System.*

3.1 Sensor Network Subsystem

As laid out before, the main task of the *Sensor Network*

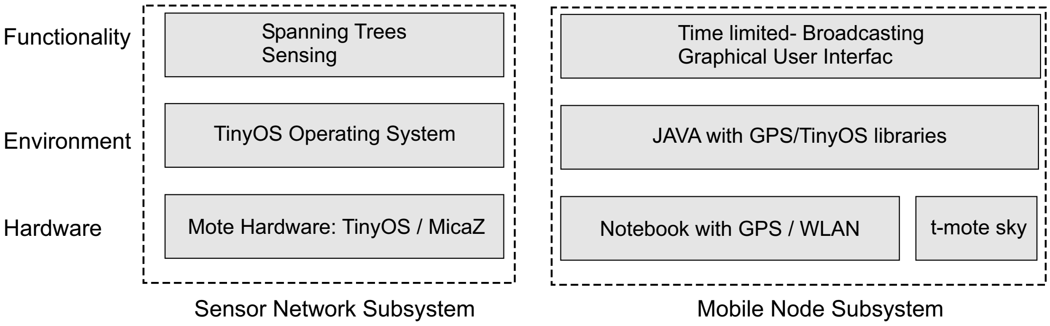


Figure 2: Prototype architecture

Subsystem is the detection of events and the delivery of adequate notifications to a near gateway sensor: Among the sensor nodes, we distinguish between ordinary sensor nodes and gateway sensors. While ordinary sensors report their readings to a near-by gateway sensor, those gateway sensors are responsible for reporting that information to a vehicle once it is in range. Within the wireless sensor network, we use simple spanning trees which are rooted at the gateway sensors for data collection. As we assume that all sensor nodes are provided with the same, limited energy supply, we argue that having fixed gateway sensors assignment is not feasible. Instead, all sensors act as gateway sensors at some point in time: If a node does not know any gateway sensor, it simply waits for a randomized time after which it decides to become a gateway sensor itself. In this case, a spanning tree is constructed within a limited range (measured by the hopcount). Similarly, gateway sensors cease functioning as such after a certain period which forces other sensors to take over.

Gateway sensors have to report the collected information to mobile nodes once they are in range, and hence, the presence of vehicles must be detected. Two basic approaches exist: Passive vehicle detection and active detection. Active detection of vehicles relies on periodic beacon messages that are sent by the mobile nodes, announcing their presence. While passive detection of cars is possible using adequate sensors, like magnetometers, we argue that a main drawback of this approach is the disability to distinguish between cars that can interact with gateway sensors directly and ones that can not: If passive detection is used, cars might be detected correctly although they are not equipped with the hardware required to receive data from gateway sensors. Passive detection will however lead to a transmission of data in such cases. As unnecessary transmissions are to be avoided in order to save energy, we decided to use active detection within the prototype, where the cars announce their presence using periodic beacon messages. Once such a beacon is received, the collected data is sent to the mobile node immediately. It is noteworthy that this way of information flow might be time crucial as mobile nodes are in range of a gateway sensor for a short time period only. Hence, the data is fitted into one single data packet.

All mentioned functionality has been implemented using TinyOS 2.0, which allows us to run the software on a variety of platforms.

3.2 Mobile Node Subsystem

The *Mobile Node System* is responsible for data-collection from the gateway sensors and the propagation of that data to other vehicles. In order to retrieve sensor data, periodic beacons are sent which also contain data to be stored into the wireless sensor network: This way, readings from a distant location can be stored at gateway sensors. Doing so enables us to notify vehicles that pass by later when no other cars are in reach. By using this piggy-back scheme, the information flows from the sensors to the cars and vice versa are interweaved and therefore the communication overhead is reduced.

In order to send data to other vehicles, so far a WLAN-based UDP flooding scheme is used in the implementation. The sensor information that is retrieved from gateway sensors is serialized and broadcasted to other vehicles in range. A additional timestamp allows to control the data's period of validity. This also leads to a limitation of the informations' geographical spread. While the current UDP dissemination scheme is a very basic approach, we argue that much research is currently carried out in the field of VANET message distribution. The implementation allows an easy integration of more advanced schemes, like Geocasting.

Considering the interaction of the mobile nodes with the gateway sensors, information is not simply retrieved from the sensor network, but also stored back into the WSN. By doing so, some kind of information persistence can be achieved: It is imaginable that the VANET breaks down, for example because of low traffic density. In this case, it is still possible to deliver information about distant events to a car if they have been sent to a gateway sensor beforehand. Within our architecture, the messages to be stored at the gateways are embedded in the beacon messages used for vehicle detection. Consequently, the gateway sensors in fact report both their own data and, if available, injected information upon the retrieval of a beacon message in an alternating manner.

4. FIELD STUDIES

In order to investigate the principal feasibility of Hybrid Sensor-Vehicular Networks, we conducted two field experiments during Spring 2007 in order to address two questions. First of all, we were interested if the direct communication between mobile nodes and gateway sensors is possible. Furthermore, a second field study with the prototype introduced in Section 3 was carried out to show that all five ways of in-

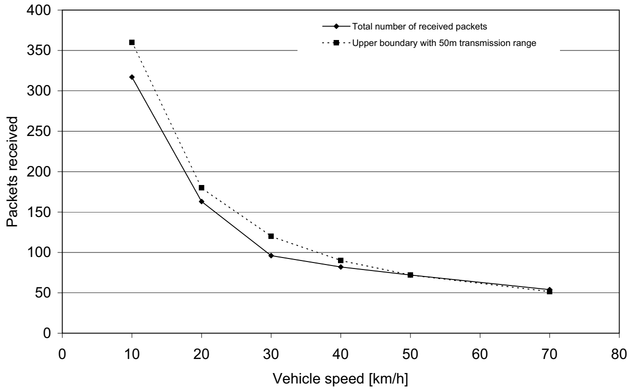


Figure 3: Beacon messages received at various speeds

formation flow can be integrated into one, cooperative system. Within this section, we present our results and experiences originating from those experiments.

4.1 Direct Communication

Within the first experiment, we deployed a Tmote SKY module alongside a country road about 30 cm over ground. The mote was flashed with a small application that broadcasted 40 packets per second. Passing by the mote with a car at various speeds, we investigated how the used 802.15.4 technology deals with mobility. Therefore, we simply counted the number of packets that could be received at various speeds. The results are summarized in Figure 3 where we compare the number of packets that were totally received with a theoretical, upper boundary. The upper boundary is derived from the vehicle speed, the packet rate and the maximum transmission range according to the specification of a Tmote SKY. As one can see, we were able to receive a significant number of packets at all speeds between 10 km/h and 70 km/h, suggesting that 802.15.4 might be a considerable option for environments where care move with low to medium speeds. Tests with higher speeds are planned for the future, but need special preparation for safety reasons.

4.2 Prototype Field Test

In a second experiment, we verified the functionality of the prototype introduced in Section 3. For this test, we deployed 16 motes on a parking lot, where the distance between two motes was about 30 m each. In order to allow an analysis of the topology, we configured five motes as static gateway nodes and used the TinyOS 2.0 Collection framework to gather the global topology within the network. A mobile node, which consisted of a notebook computer equipped with GPS and an attached Tmote SKY, was carried through the field. Figure 4 shows a screenshot of the prototype's GUI component that provides real time visualization of the sensor data gathered from gateway sensors.

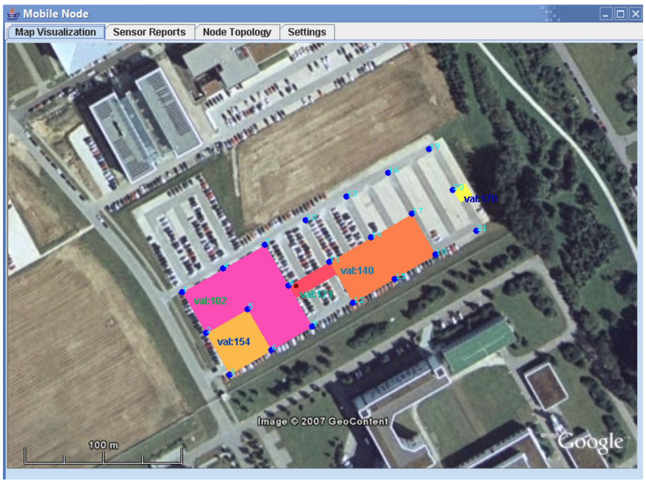


Figure 4: Sensor Data Visualization during field test

Furthermore, the prototype allows to control the injection behaviour of the mobile node and logs any packet that is received from a gateway sensor. An analysis of these log files revealed that both ways of information flows, from the gateway sensors to the mobile node and vice versa, could be realized. In addition, we checked if the communication between two mobile nodes was possible. Therefore, a second mobile node was turned on in range, and as this mobile node was equipped with WLAN but no mote for sensor network communication. As expected, all data was available at the second mobile node within a one or two seconds.

5. SUMMARY

Within this paper, we presented some key ideas and results from our work on on Hybrid Vehicular-Sensor Networks. Our work can be considered as initial analysis of such systems where yet many challenging questions exist. Of particular interest are questions regarding energy efficiency within the WSN and questions on reliability. We believe that Hybrid Sensor-Vehicular Networks could be a tool to effectively warn drivers in case of dangerous road situations such as ice and aquaplaning and that the direct combination of wireless sensor networks and VANETs is also more cost effective than solutions which address the same application domain but which rely on a more complex infrastructure. At the same time, availability and accuracy should be much higher compared to solutions that rely on vehicles alone.

6. REFERENCES

[1] J.C. Navas, T. Imielinski. GeoCast – Geographic Addressing and Routing. Proceedings of the 3rd annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom), 1997, 66-76

Handover in Sensor Networks using Statistic-Based Routing

Alexander Klein
Innovation Works
EADS Deutschland GmbH
Munich, Germany

klein@informatik.uni-wuerzburg.de

Phuoc Tran-Gia
University of Wuerzburg
Institute of Computer Science
Wuerzburg, Germany

trangia@informatik.uni-wuerzburg.de

ABSTRACT

The support of mobility represents one of the challenging tasks in Wireless Sensor Networks (WSN). The problem caused by frequent topology changes is to find a tradeoff between reliability and energy efficiency. The support of mobility often results in additional complexity of the routing protocol. However, due to limited memory and computational power of small wireless devices, the routing protocol has to be kept as simple as possible.

In this paper, we give a short introduction of our Statistic-Based Routing protocol. Furthermore, we analyze the capability of the protocol to deal with the problem of handovers. In addition, the reliability during the process of handover is simulated under various conditions.

Categories and Subject Descriptors

A.0 [**Introductory and Survey**]: Miscellaneous; I.6.4 [**Model Validation and Analysis**]: Miscellaneous;

General Terms

Documentation, Theory, Measurement Performance

Keywords

Statistic, Routing, Wireless, Sensor, Networks

1. INTRODUCTION

The improved capabilities of wireless sensors have raised the interest in WSN solutions. The higher demand results in lower hardware prices. Therefore, an increased number of applications become interesting under economical aspects. Sensor networks have high requirements on the routing protocol due to the large number of nodes. Scalability represents an important issue in WSNs since the nodes have to solve the problem of routing table explosion. Beside the problem of scalability, the routing protocol has to deal with frequent topology changes. These changes can be the consequence of link breaks caused by e.g. sleep times, interference, energy exhaustion or mobility.

Consider an ad hoc routing protocol like the Optimized Link State Routing Protocol (OLSR) [1] in a mobile WSN. The routing table with its expiry timers and the required neighbor, two hop neighbor and Multi Point Relaying (MPR) lists would consume most of the nodes memory in large WSNs. Furthermore, the frequent recalculation of the topology and the MPR set would consume most of the computational power of a node in large and dense networks.

For that reason, we are looking for a new solution that is able to deal with the typical WSN issues without the need of complex algorithms. We investigate Statistic-Based techniques because they represent a possibility to implement adaptive behavior.

The work is organized as follows. Section II gives a short overview of protocols that have similarities to our approach. A brief description of the functionalities of the presented protocol is given in Section III. The problem of handover and the capability of the approach to deal with it are presented in Section IV. The results of our simulations are discussed in Section V. An introduction of our work in the future is given in the final section.

2. RELATED WORK

In this section we describe two paradigms that have many similarities compared to our approach. The basic functionalities of these paradigms were used to create a new approach.

The first one is represented by direct diffusion which was introduced by C. Intanagonwiwat et. al [2]. The idea behind direct diffusion is to label data by attribute-value pairs. These pairs are used to identify the content and further allow data aggregation.

A request or interest for this data is propagated through the network via broadcast messages that are transmitted by the sink. The broadcast messages are forwarded by intermediate nodes to the source. Each intermediate node sets up a gradient towards the previous node from which it has received the broadcast message.

Data that is transmitted by the source is routed along the gradients. The strength of the gradients is set according to the capabilities of the forwarding node and the attribute-value pair. The variation of the gradient strength can be used to disseminate traffic equally across the network.

The Minimum Cost Forward Algorithm (MCFA) [3] presents a popular idea that is based on the paradigm of direct diffusion. In this protocol, the base station broadcasts messages. These messages contain a cost field which is set to zero. Before a node retransmits a message it increases the value stored in the field by the cost of the link from which it has received the message. However, a message is only forwarded if the new value is smaller than the one previously stored.

Another way to deal with routing issues is followed by protocols using swarm intelligence. The most known heuristic for solving the problem of routing is presented by ant-based algorithms. The principle of ant-based algorithms is that shorter paths are traveled more frequently than others. Thus, the pheromone that is left on a path strongly depends on how often the path is traveled. Data is forwarded according to a link probability function. The function uses the amount of pheromone as input to calculate the next hop. Therefore, the protocol uses the information gathered from the past to decide the next hop. After each time step a certain percentage of pheromone evaporates. As a result, existing paths may change or even disappear over time. However, Zhang et. al [4] have shown that standard ant-based routing algorithms have to be adapted to meet the requirements of WSNs.

3. STATISTIC-BASED ROUTING

In this section the functionalities of the Statistic-Based Routing are introduced. Furthermore, we describe the creation and forwarding of hello messages. In addition, the usage of the routing table is explained.

3.1 Hello Messages

Hello messages are periodically transmitted by each node. The message consists of a source, intermediate, sequence number, and a time-to-live field. The source field holds the address of the originator of the message. The address of the node which has forwarded the packet is stored in the intermediate field. The identifier of the message is represented by the value in the sequence number field. Each time a node transmits a new hello message it increments the sequence number by one. The time-to-live field indicates how often a hello message can be retransmitted.

A node creates an entry in its routing table if it receives a hello message from another node for the first time. If a node already knows the creator of the message it compares the sequence number in the hello message with the value stored in its routing table. If the number in the message is higher than the stored value, the entry in the table is updated and the packet is considered for forwarding. In the case that both values are equal the node compares the time-to-live fields. If the time-to-live value is smaller than that of the stored entry it is forwarded. Otherwise the message is discarded. Furthermore, messages are only considered for forwarding if they are received from the best ranking neighbor.

3.2 Routing Table

The routing table stores values that correspond to the link quality or another metric depending on the used routing entry increase algorithm. The value is increased each time a new hello message is received. A hello message is called new if it is stored. Figure 1 shows a connectivity graph of an example network.

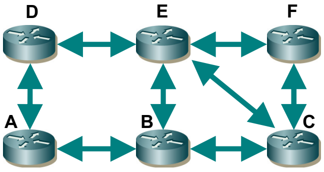


Figure 1: Connectivity Graph of the Example Net

If the entry is increased by one every time a new hello message is received then the stored value directly corresponds to the number of received hello messages. Table 1 represents an example routing table of node B resulting from the connectivity graph shown in Figure 1.

Table 1: Example Routing Table of Node B

Node B	**Number of Received Hello Messages**				
Originators	A	C	D	E	F
A	20	-	-	-	-
C	-	20	-	-	-
D	12	-	-	7	-
E	-	-	-	18	-
F	-	8	-	12	-

The columns represent the neighbors through which the new hello messages are received. An empty column is the result of the fact that the node is not a neighbor. Several different paths to the destination exist if more than one value is stored in a row.
The values in the routing table are decreased according to the used routing entry decrease algorithm. The function to decrease the values is called every Decrease Routing Value Interval.

3.3 Configuration

The Statistic-Based Routing offers the possibility to adapt its behavior by setting the following parameters: Hello Message Interval, Hello Message Time-To-Live, Hello Message Forwarding Delay, Decrease Routing Value Interval, Maximum Routing Value, Increase Routing Value Function, and Decrease Routing Value Function.

Each node transmits hello messages according to the Hello Message Interval. The broadcast of hello messages can be reduced by using a smaller Hello Message Time-To-Live value. In addition, the forwarding of hello messages can be delayed such that a node falls back in the routing tables of the other nodes. This mechanism could be used to spread traffic equally across the WSN. A Decrease Routing Value Interval is used to trigger the decrease of the routing entry values. Furthermore, the values in the table are limited by the Maximum Routing Value to limit the reaction time of the system to recognize topology changes. Functions can be used instead of constant values to change the entries of the table.

4. HANDOVER

The behavior of the protocol during a handover is discussed in this section. First, we explain the increase and decrease of the entries in the routing tables when a mobile node X passes two fix nodes A and B. To keep things as simple as possible we assume a constant increase and decrease value of one. The Decrease Routing Value Interval is twice as long as the Increase Routing Value Interval.

Figure 2 shows the trajectory of node X and the transmission range of nodes A and B. Furthermore, important points in time are marked by dashed lines. In addition, we assume that no hello messages are lost and node X moves with constant speed.

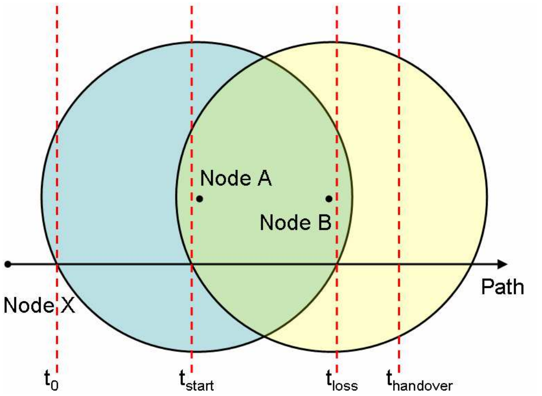


Figure 2: Example Handover Scenario

Node X enters the transmission range of node A at time t_o resulting in the increase of the routing value in node A. At time t_{start} node X can be reached by nodes A and B. Therefore, the routing entry in node B increases, too. Node X leaves the range of node A at time t_{loss}. As a consequence, hello messages that are transmitted by node X are only received by node B. Thus, the routing value stored in node A decreases whereas the value in node B further increases. The behavior of the entries is shown in Figure 3.

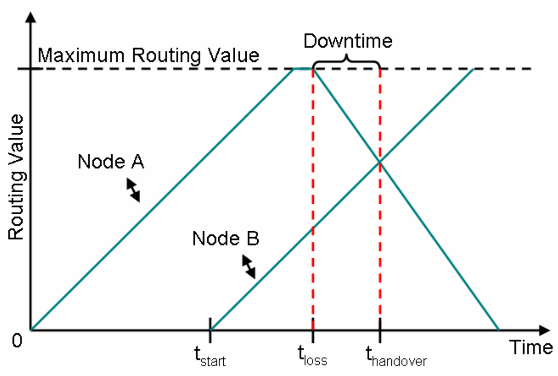


Figure 3: Routing Values during Handover

No packets can be routed to node X between t_{loss} and $t_{handover}$ because the routing value of the corresponding entry is still higher in node A. The protocol assumes that node A is the node in charge to reach node X until the value in node B becomes higher. The time needed by the protocol to select the correct node is referred to as Downtime. No packets can be forwarded to node X during the Downtime.

However, linear functions do not represent the best choice because of their limited capabilities to minimize the Downtime. It has to be kept in mind that using a higher gradient for the Increase Routing Value function results in a higher routing value which has to be decreased later on by the Decrease Routing Value function. The only possibility to shorten the Downtime with linear functions is to use a higher gradient for the Decrease Routing Value function. In addition, a Maximum Routing Value could be chosen to further minimize the Downtime. Nevertheless, the gradients and the Maximum Routing Value have to be selected in respect to the set Hello Message Interval and the Decrease Routing Value Interval.

Instead of using linear functions we chose functions which take the current value into account. The following characteristics are desired. The gradient of the Increase Routing Value function should be high for low values and low for high values. The gradient of the Decrease Routing should be high for high values and low for low values. The chosen Increase Routing Value function has to be asymptotic. Otherwise a Maximum Routing Value has to be set to limit the routing entry values.

Many different functions can be used to achieve quite good performance. Here we use the following equations to calculate the increase and decrease of the routing entry values.

$$f_{n+1} = 2f_n + \frac{4}{f_n^{\,2} + 1}$$

Equation 1: Sample Increase Routing Value Function

$$g_{n+1} = \frac{g_n}{2}$$

Equation 2: Sample Decrease Routing Value Function

To give a better impression of the characteristics of the functions we make the following assumptions. The Hello Message Interval and the Decrease Routing Interval are set to the same value. The increase is directly followed by the decrease. In addition, we simulate the loss of a hello message by using a uniform distribution. We set the uniform distribution such that three percent of the hello messages are discarded. Thus, the decrease function is called more often than the increase function. A typical development of the value within a routing entry is shown in Figure 4.

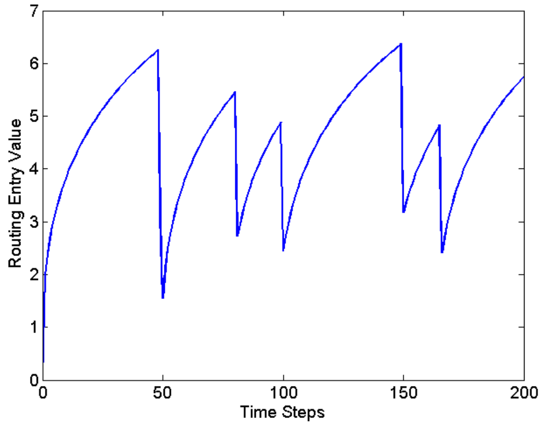


Figure 4: Snapshot of Routing Value Function

The drops of the values are the result of lost hello messages. Therefore, the decrease function is called two times after another.

5. SIMULATION RESULTS

It is clear from the definition of the protocol that it finds a path to a destination in the network if all nodes are fix. We want to simulate the capability of the protocol to deal with changing topologies. Thus, we place nodes along a manhatten grid such that a node is able to communicate with its direct neighbor along the grid. Then we add a mobile node that moves straight through the network with a constant speed. If the node reaches the border of the scenario it turns around and moves on with the same speed. A node at the border of the network tries to transmit packets to the moving node. The inter-arrival time of the packets is chosen according to an exponential distribution with a mean value of one second. The size of the packet is chosen such that the traffic load of the network is very low. The protocol is developed as part of our WSN simulation framework within the OPNET Modeler [5].

In the first scenario we simulate the end-to-end reliability of the protocol depending on the speed of the moving node. The duration of the Hello Message Interval is set to one second to minimize the reaction time of the protocol. Furthermore, we set the Decrease Routing Value Interval to 1.25 seconds. These values ensure that the increase function is at least called once before the decrease function. However, this is only the case if the hello messages are not lost. It has to be kept in mind the duration of these intervals have to be considered when choosing the increase and decrease functions. Equations 1 and 2 are used to modify the entries in the routing tables. To simulate the handover performance we vary the speed of the moving node from 2 meters per second to 20 in steps of 2. The transmission range of the nodes is set to 140 meters. The grid length is 125 meters. Thus, communication between direct neighbors along the grid is possible.

Figure 5 shows the end-to-end reliability of the traffic between the source node and the mobile destination depending on its speed.

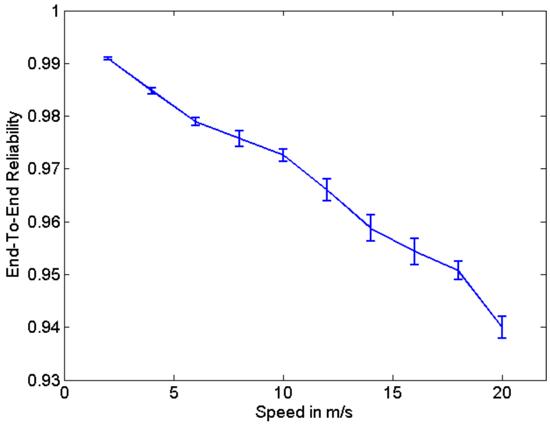


Figure 5: End-To-End Reliability

The intervals are the 99 percent confidence intervals. The results of Figure 5 point out that the end-to-end reliability decreases linearly for speed values between 2 and 20 meters per second. However, the averages of lost packets per handover depending on the mobile node speed which are presented in Figure 6 show a different trend.

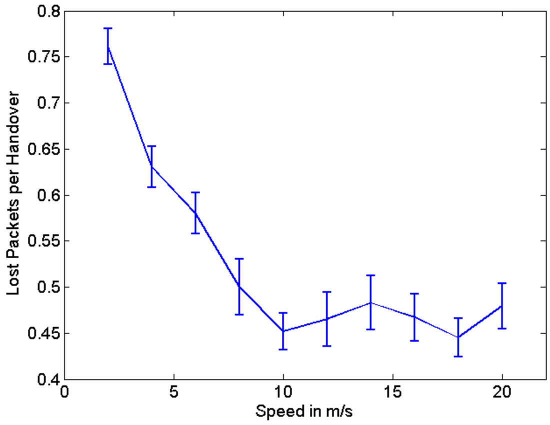


Figure 6: Number of Lost Packets per Handover

The number of lost packets per handover is higher for speeds lower than 10 ms^{-1} compared to the results from simulations with a higher mobile node speed. A slow node remains a long time within the range of its supporting node. Thus, the routing entry value has much time to reach a high value. As a consequence, more than one hello message is required by the protocol to recognize that another node has to be chosen as supporter for the mobile node. It has to be kept in mind that nodes with higher values are considered to be more reliable than those with a smaller value. The constant value of 0.5 for speeds higher than 10 ms^{-1} is the result of the Hello Message Interval duration of one second.

6. FUTURE WORK

Due to its simplicity Statistic-Based Routing should be considered as alternative to existing routing protocols in networks where nodes have limited capabilities. Our future studies include the simulation of sensor nodes with sleep times using the Statistic-Based approach. In addition, we will analyze the capability of the protocol to save energy in WSNs through data dissemination.

7. REFERENCES

[1] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR) RFC 3626", IETF Network Working Group, October 2003.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin, “Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", Proceeding of ACM MobiCom `00, pp. 56-67, Boston, MA, 2000.

[3] F. Ye, A. Chen, S. Liu, and L. Zhang, “A Scalable Solution to Minimum Cost Forwarding in Large Sensor Networks”, Proceeding of the tenth International Conference on Computer Communications and Networks (ICCCN), pp. 304-309, 2001.

[4] Y. Zhang, L.D. Kuhn, and M.P.J. Fromherz, “Improvements on Ant Routing for Sensor Networks”, Proceeding of the fourth International Workshop ANTS 2004, pp. 154-165, Brussels, Belgium, September, 2004.

[5] OPNET Modeler, OPNET University Program: http://www.opnet.com/services/university/.

Optimizing TDMA Design for Real-Time Applications in Wireless Sensor Networks

Nicos Gollan and Jens Schmitt
TU Kaiserslautern
Distributed Computer Systems Lab
p.o. box 3049
67653 Kaiserslautern, Germany
gollan@informatik.uni-kl.de
schmitt@informatik.uni-kl.de

ABSTRACT

Many wireless sensor networks (WSNs) are used to collect and aggregate data from potentially hostile environments. Catering to this, early application scenarios did not put tight constraints on performance properties like delay, but rather focused on ruggedness and energy conservation. Yet, there is a growing number of scenarios like e.g. production monitoring, intrusion detection, or health care systems which depend on the sensor network to provide performance guarantees in order to be able to act upon the phenomena being sensed in a timely fashion. Nevertheless, these applications still face the traditional issue of energy-efficiency. In this paper, we present means to find energy-efficient medium assignments in time-slotted multi-hop networks that satisfy given real-time constraints. Specifically, we present a way to find the optimal length of time slots and periods in TDMA schemes. We also present a software to compute those values for typical sink-tree WSNs.

General Terms

Wireless Sensor Networks, Energy-Efficiency, Real-time guarantees, Optimal TDMA, Network calculus

1. INTRODUCTION

Wireless sensor networks have requirements and characteristics that are considerably different from those of common computer networks. Issues include low energy reserves, limited processing power, uncontrolled environments and other adverse factors. The main attention has been put on meeting these restrictions, so there has been much research on minimizing energy usage by introducing sleep times, reducing the amount of transmitted data, etc. While those topics remain important, other aspects have been neglected. With long sleep times come long delays, which can grow rapidly with the size of the network, depending on its topology.

Traffic flows as well as scheduling regimes have so far often been considered fluid, neglecting aspects like packetizing or TDMA and describing them only by their sustained maximum or average rate and modifiers like a burst for incoming traffic or latencies for services, which is a good approximation for fast and relatively fine-grained data streams. When looking at very slow data streams however, that model loses precision. With slow medium data rates and processing, data sent over such a network loses its fluid characteristics. This becomes a concern in TDMA systems, where participants only receive intermittent service during their assigned time slots, and need to be quiet outside those slots. Modeling such a system with fluid models fails to take this into consideration.

We present the optimization problem of finding the most energy-preserving frame length in a TDMA system while still meeting worst-case delay constraints, and we show an analytical approach to compute that value in generic sink-trees. We also present an implementation using the existing DISCO Network Calculator framework [7], allowing us to analyze the impact of discrete models on worst-case delay bounds. This provides us with a means to compute bounds for TDMA parameters in sink-tree sensor networks.

The problem we address is different from other approaches that try to find optimally fast solutions, like e.g. [4], in that we find the *minimum* medium allocation that still allows the WSN to work within the given limits. Other related work covers aspects of TDMA networks, like time synchronization [9] or resource allocation and reuse in spatially spread out networks [1 5]. The general tenor is to maximize performance of a network, whereas we aim for the minimum performance at which quality of service requirements, in this case delay, are still fulfilled. thus achieving energy-efficiency.

This work is founded on the basic network calculus as described in [2]. Furthermore, it draws upon the extensions towards sensor networks which were introduced in [6] and further elaborated in [8]. Based on that, the DISCO network calculator has been created which has been extended for the numerical part of this paper.

2. OPTIMAL TDMA DESIGN

When designing a TDMA system, a choice has to be made for how long the repetitive TDMA frame as well as the individual slot sizes of each participating node are. Since the advantage of TDMA systems against concurrent medium access lies in the fact that each participant obtains exclusive use of the medium, it has to be ensured that each participant gets assigned enough time to perform its tasks. For some network nodes, that just requires a short slot in which they can send collected data, and perhaps receive an acknowledgment from an upstream node. However, in multi-hop systems, some nodes act as routers, and have higher bandwidth requirements for forwarding other nodes' data, while perhaps collecting and sending data themselves. Aside from avoiding contention, using TDMA also reduces energy consumption by making it possible for nodes to power down in periods without relevant traffic.

Since in wireless sensor networks, two main concerns are minimizing power consumption and meeting delay bounds, while transmission bandwidth requirements tend to be low, we want to maximize the frame length, giving the sensor nodes the opportunity to disable their radio transceivers or even go into deep sleep modes.

2.1 General TDMA Design Problem

From those requirements, we formulate the TDMA design problem as an optimization problem for a tree network with n nodes where from each node a flow is originating:

$$\begin{array}{lll} \text{max.} & Z = \min_{1 \leq i \leq n} \{f - s_i\} & \\ \text{s.t.} & \sum_{i=1}^{n} s_i \leq f & \text{(TDMA integrity)} \\ & \forall i: \; d_i(f, \vec{s} | r, b, C) \leq D & \text{(Delay)} \\ & \forall i: \; \frac{s_i}{f} \cdot C \geq F_i r & \text{(Rate)} \\ & \forall i: s_i \geq 0, \; f \geq 0 & \text{(Non-negativity)} \end{array}$$

Here, f is the length of the repetitive TDMA frame, s_i is the amount of time devoted to node i for sending (slot size of node i). These constitute the decision variables. For the parameters of the problem we further have, D as the maximum permissible delay that may be incurred by any flow in the sensor field, $d_i(f, \vec{s}|r, b, C) = h(\gamma_{r,b}, \beta^i_{eff})$ as the actual maximum delay incurred by flow i (which is computed based on the effective service curve β^i_{eff} computed by a PMOO analysis [8] of the network), F_i as the number flows carried by node i (including the flow originating at node i), C as the medium rate ("capacity"), and r as the (maximum) sustained rate for any flow as well as b as the maximum burst of a flow. Note that we assume the sensors to have identical arrival curves $\gamma_{r,b}$ as well as an identical delay requirement D, which for most practical situation will be no restriction and makes the further analysis more tractable.

The objective function reflects the fact that the minimum sleeping period over all nodes in the field should be maximized, thus achieving a maximum lifetime of the network. The TDMA integrity constraint captures the fact that all slot sizes together must fit into the TDMA frame. Obviously the delay constraints should be met for all flows, which is captured by the delay constraints, as well as all the rate constraints must be met in order not to obtain infinite delay bounds for the flows. Of course, we also have non-negativity constraints for the decision variables. As a remark, in the objective function there is a hidden assumption on the relative energy costs for switching between different states like transmission, reception, and idle. It is assumed that those state transitions have roughly the same cost, as it is the case for many transceiver architectures (see e.g. [10]), such that by minimizing the sleep period of a node with respect to sending will in fact coincide with minimizing the amount of energy consumed for transmission *and* reception by maximally batching data before forwarding them (within the delay constraints).

Unfortunately, this general modeling of the TDMA design problem results in a very hard to solve non-linear programming problem with $n+1$ decision variables and $3n+2$ constraints. The non-linearity is exhibited in the objective function as well as in the delay constraints. Hence, the only viable approach is to simplify the problem structure if a solution shall be found for larger instances of the TDMA design problem. There are two intuitive approaches towards relaxing the problem:

1. *Equal Slot Sizing (ESS):* the assignment may be made such that inside a fixed time slot length, each node can transmit enough data to fulfill all requirements.
2. *Traffic-Proportional Slot Sizing* (TPSS): slots may be assigned such that each node only claims the resources necessary to fulfill its own duties, depending on the input bandwidth and forwarded data streams.

While the second relaxation approach may appear more efficient, it is also harder to set up. The first approach requires rather little information – the number of nodes and the bandwidth requirements of the node serving the highest number of flows –, the second method requires good knowledge of the topology, which may not always be at hand.

In both cases two variables need to be controlled: The overall frame length f and the individual slot length s, where however for ESS this slot length is for each node, while for TPSS each node obtains a multiple of that slot length depending on the number of flows it has to carry. Obviously in both cases, with an increasing frame length, a node may sleep longer between transmission or reception phases, but delay is increased at the same time. For a given f in a network with n nodes, s is limited to values between an upper bound $\frac{f}{n}$ and a lower bound that is given by the minimum bandwidth requirements.

Interestingly, despite the more intuitively more appealing nature of the TPSS relaxation, the ESS relaxation achieves better results with respect to maximizing the minimum sleep period in the field. As it is also the easier one to solve we further on focus on the ESS relaxation for solving the TDMA design problem:

$$\begin{array}{ll} \text{max.} & Z = f - s \\ \text{s.t.} & s \leq \frac{f}{n} \\ & d(f, s|r, b, C) = \max_{1 \leq i \leq n} d_i(f, s|r, b, C) \leq D \\ & \frac{s}{f} \cdot C \geq F_{max} r \\ & f \geq 0 \end{array}$$

Fig. 1 shows a simple network and the resulting graphical representation of the optimization problem for $C = 10$, $r = 1$, $b = 1$, $D = 1$. It can be seen that the optimum must be taken on at the lower border of the feasible region. In fact, we can derive a closed form of the the border constituted by the delay constraint because the delay constraint is a quadratic form in s and f which can be solved for s with two real solutions of which we take the larger one as it results in a more binding constraint. A moment's consideration exhibits that, with g as the solution of the delay constraint for s, $\forall f: \frac{\partial g}{\partial f} < 1$ since otherwise an increase in frame size would result in a larger increase of the slot size, which obviously cannot be the case. On the other hand the partial derivative of the objective function after f is 1, which means that the optimum must be taken on at the corner point of the feasible region where delay constraint and TDMA integrity constraint intersect. In other words, the *TDMA design problem under ESS* can be reduced to *matching the delay with the TDMA integrity constraint.*

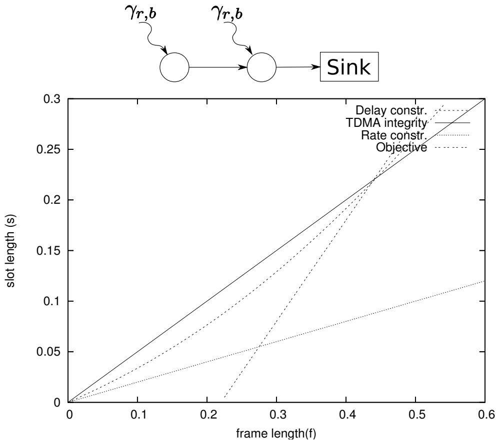


Figure 1: A two-node sample network and the graphical illustration of the resulting optimization problem for ESS. The feasible region is above the delay constraint and below the TDMA integrity constraint. The rate constraint does not affect the feasible region in this example.

2.2 Analytical Solution for ESS in General Sink Trees

What remains to do in general sink trees compared to the two-hop network in the previous setting is to show that the delay constraint again takes on a quadratic form. This then allows to easily express the slot size as a function of the frame size and the same arguments as in the two hop case will lead to the conclusion that the optimum solution is given at the point where delay and TDMA integrity constraint are matched. Hence let us discuss the delay constraint in a general sink tree network:

We assume a general sink tree network with each node offering a service curve $\beta_{\frac{s}{f}C,f-s}$ and flows starting from each node constrained by arrival curve $\gamma_{r,b}$. Looking at a particular flow we have a situation as depicted in Fig. 2. Applying the PMOO analysis results in the following effective service curve for the flow of interest:

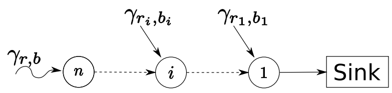


Figure 2: Flow in a general sink tree

$$\begin{aligned}\beta^n_{eff} &= \left[[\beta_{R,T} - \gamma_{r_1,b_1}]^+ \otimes \beta_{R,T} - \gamma_{r_2,b_2}\right]^+ \otimes \dots \\ &= \beta_{R-\sum_{i=1}^{n-1} r_i,\, \frac{T\left(nR-\sum_{i=0}^{n-1}\sum_{j=1}^{i} r_j\right)+\sum_{i=1}^{n-1} b_i}{R-\sum_{i=1}^{n-1} r_i}}\end{aligned}$$

with $R = \frac{s}{f}C$ and $T = f-s$ and $r_i = a_i r$ and $b_i = c_i b + d_i r T$. For the latter expressions the parameters $a_i, c_i, d_i \in \mathcal{N}$ are depending on the topology. The delay constraint for flow n (i.e the one originating at node n) can thus be expressed as a quadratic form in f and s which can be recast to s where we can ignore the smaller right hand side version as the larger one is the physically meaningful. Hence, we can again reason that the optimal solution must be taken on at the cornerpoint of the feasible region where TDMA integrity and delay constraint are matched.

3. NUMERICAL APPROACH IN THE DISCRETE SETTING

For the numerical approach, we extended the DISCO Network Calculator [7] to handle affine curves as described in [3], which allows us to model discrete service and arrival curves and perform network analysis with such. We specifically model the service curves like β shown in Figure 3. The arrival curve is modeled as a simple token bucket, mostly to reduce computing time. It also resembles the behavior of a sensor that is constantly doing measurements, like a low-bandwidth audio stream or some other task that requires frequent sampling.

The TDMA design problem, based on the insight from the previous section that TDMA integrity and delay constraint have to be matched, ultimately boils down to a root-finding problem: a black-box function $d(f, s|r, b, C, D)$ returns the delay incurred for the parameters. Since we have seen that the optimum is found at $s = \frac{f}{n}$, this is a function with one variable $d(f|r, b, C, D)$. Since we want to maximize the frame length f, we seek a solution f' for which $d(f'|.) = D$, which is a root for $d(f|.) - D = 0$. We further know that $\forall 0 < f < f': d(f|.) - D < 0$ and $\forall f > f': d(f|.) - D > 0$. Thus, we can use interval bisection to find the root. We are specifically using an algorithm that approaches the root from below, making sure that the calculated value is smaller than f', ensuring it is inside the feasible region.

3.1 Numerical Examples

Using that procedure, we set up a number of sample networks, among them the one from the previous section and fully populated binary trees of depth 3 and 5 and analyzed those in the enhanced DISCO Network Calculator. The results presented in Table 1 show that a discrete analysis has an advantage over an analysis in the fluid domain. The magnitude of that advantage depends on the scenario. The

Figure 3 explains the improvement. The latency bound is computed as the maximum horizontal distance between the arrival curve α and the service curve. In the fluid setting, the service curve models the average medium rate $\frac{s}{f}C$ available to a node, as shown by the curve β'; in the discrete setting, a node periodically gets the full rate C, which makes the discrete service curve β "jump" above β'. Because of this, it holds that $\forall \alpha \in \mathcal{F}: h(\alpha, \beta) \leq h(\alpha, \beta')$.

					fluid	discrete
Setting	C	r	b	D	f	f
2 nodes	10	1	1	1	0.4444	0.7368
Binary tree, 3 deep	5000	1	1	10	3.5426	3.5863
Binary tree, 5 deep	5000	1	1	10	1.4920	1.6863

Table 1: Comparison of numerical results in fluid and discrete settings

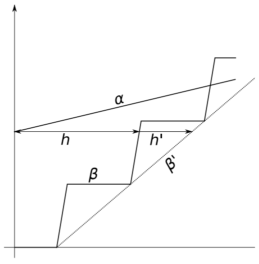


Figure 3: Illustration of the improvement

It is apparent from the numerical results that for larger networks, the relative advantage of a discrete analysis becomes smaller. The same holds for service curves with a lower step height. This is because, due to the nature of the effective end-to-end service curve, the step size remains the same, but the latency grows with each hop and interfering flow. When referring to Figure 3, h grows faster than $h' - h$.

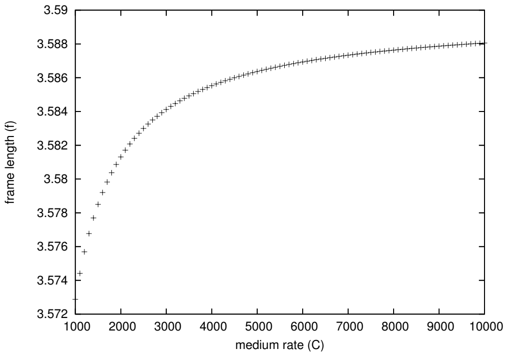


Figure 4: Frame lengths for varying medium speed

It is also notable that in our test-cases, the optimal value for f scaled linearly with D, so for a given network with all other parameters fixed, the frame length can be easily extrapolated from a few computed values. The behavior with changing C is potentially more interesting, since a modified service curve is accompanied with it. However, when looking at a sample data set in Figure 4, which was created by varying C from 1000 to 10000 in steps of 100 with $r = 1$, $b = 1$ and $D = 10$, one notices that the gains in the possible frame lengths are minimal and even diminishing for larger rates. Such small gains in frame length are likely outweighed by the rising energy requirements imposed by the hardware necessary to achieve a faster transmission speed.

4. REFERENCES

[1] Patrik Björklund, Peter Värbrand, and Di Yuan. Resource Optimization of Spatial TDMA in Ad Hoc Radio Networks: A Column Generation Approach.

[2] Jean-Yves Le Boudec and Patrick Thiran. *Network calculus: a theory of deterministic queuing systems for the internet.* Springer-Verlag New York, Inc., New York, NY, USA, 2001.

[3] Anne Bouillard and Éric Thierry. An algorithmic toolbox for network calculus. Technical report, Unité de recherche INRIA Rennes, 2007.

[4] Shuguang Cui, A. Madan, R.and Goldsmith, and S. Lall. Energy-delay tradeoffs for data collection in TDMA-based sensor networks. In *ICC 2005 Vol. 5.* IEEE, May 2005.

[5] S.C. Ergen and P. Varaiya. TDMA scheduling algorithms for sensor networks. Technical report, University of California, Berkley, 2005.

[6] Jens Schmitt and Utz Roedig. Sensor Network Calculus - A Framework for Worst Case Analysis. In *Proceedings of IEEE/ACM International Conference on Distributed Computing in Sensor Systems (DCOSS'05), Marina del Rey, USA*, pages 141–154. Springer, LNCS 3560, June 2005. ISBN 3-540-26422-1.

[7] Jens B. Schmitt and Frank A. Zdarsky. The DISCO Network Calculator - A Toolbox for Worst Case Analysis. In *Proceedings of the First International Conference on Performance Evaluation Methodologies and Tools (VALUETOOLS'06), Pisa, Italy.* ACM, November 2006.

[8] 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.

[9] M. Sichitiu and C. Veerarittiphan. Simple, accurate time synchronization for wireless sensor networks. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC) 2003.* IEEE, 2003.

[10] A.Y. Wang and C.G. Sodini. A simple energy model for wireless microsensor transceivers. In *Global Telecommunications Conference 2004.* IEEE, IEEE, November 2004.

Enabling the Sleep Mode in Non-beaconed 802.15.4 Multihop Networks - A Simulative Investigation

Barbara Staehle, Tobias Hossfeld,
Matthias Kuhnert
University of Würzburg
Institute of Computer Science
bstaehle@informatik.uni-wuerzburg.de

Norbert Vicari

Siemens AG
Corporate Technology
norbert.vicari@siemens.com

ABSTRACT

Large wireless sensor network deployments used for environmental monitoring or cargo tracking, require energy efficient mesh topologies. This implies duty cycling of sensor nodes to be coordinated with the routing protocol. Staying in the context of ZigBee, we simulate the combination of the sleep enabled non-beaconed mode of 802.15.4 and AODV routing and compare the duty cycling effects of synchronized and unsynchronized sleep scheduling. We consider two different link layer feedback schemes for AODV, denoted as *regular* and *smooth* AODV.

1. INTRODUCTION

The large number of purposes, a Wireless Sensor Network (WSN) can be dedicated to, are as different as habitat monitoring, animal tracking, environmental surveillance, forest fire detection, cargo tracking, industrial automation, home automation or intrusion detection. All those situations have specific requirements and challenges, thus the deployed hardware, radio communication techniques and protocols are manifold.

The use of a standardized communication layer would simplify and enable the interoperability of different WSN deployments. Among the existing IEEE wireless communication standards, 802.15.4 [2] seems to be the most suitable. It specifies the PHY and MAC layer for low rate Wireless Personal Area Networks (LR-WPANs) and promises to enable cheap wireless networking for applications with limited battery power and small throughput requirements. ZigBee is a set of network, security and application layer protocols, that were specified upon 802.15.4 to create a universal platform for use cases like home, building and industrial automation [9].

In these situations, one hop star topologies, where a single (PAN) coordinator broadcasts beacons to synchronize surrounding devices, are most suitable. Thus, much work has been dedicated to the performance analysis of those so called *beacon-enabled* 802.15.4 networks. If however, large scale multihop 802.15.4 WSNs with battery powered nodes should be established, these have to be realized as *non-beacon-enabled* PANs. The most outstanding issue in such networks is their size: they are too large to be synchronized by a single PAN coordinator but nodes have also to switch to sleep state to increase the battery lifetime. It is proposed to use an AODV [4] like routing algorithm for non-beaconed PANs [9], hence, sophisticated distributed sleep scheduling strategies have to be deployed to make such a routing algorithm work. In this work we investigate non-beaconed 802.15.4 networks and especially the interdependency of multihop routing and sleeping sensor nodes. As most previous work has focused on beaconed 802.15.4, these problems have not yet been considered.

This work is structured as followed: In Section 2 we review related work. In Section 3, we describe the simulation setup in *ns-2* [7], we used for our performance evaluation, whereof we show results in Section 4. In Section 5 we conclude and give an outlook on future work.

2. RELATED WORK

One of the first performance evaluations of 802.15.4 is reported in [8]. The authors used *ns-2* to investigate the general performance of 802.15.4 and inquired various aspects like the efficiency of slotted and non-slotted CSMA/CA during the contention access period (CAP) of a 802.15.4 superframe more deeply. The authors did however not consider the problem of establishing a multihop routing topology and assumed, that their nodes were always on. To analyze the CAP and the influence of radio-shutdowns on the energy consumptions more deeply, the authors of [5] implemented a more realistic energy and node state model. More modifications to the code have been made by the authors of [6], who examined the influence of the number of backoff periods used for CSMA/CA in the CAP on the network efficiency.

The requirements for a WSN MAC protocol comprise more than just channel access. In order to guarantee a maximal lifetime and nevertheless maintain a certain transmission delay and throughput, the MAC layer is responsible for duty cycling the sensor node's radio unit [3]. As the challenges are manifold, more than 50 different proposals for MAC WSN protocols exist. The 802.15.4 standard, however, does only specify the channel access and does not consider sleep scheduling. As this is mandatory, if 802.15.4 shall be used for sensor networks, we examine the performance of two straightforward duty cycling concepts, namely synchronized and randomly scheduled sleep periods.

If 802.15.4 shall be used for a multihop WSN, a routing topology has to be established. The applicability of the classical AODV and several of its modifications has been investigated in a 802.15.4 testbed consisting of 15 nodes [1]. The results showed, that both the existing and the newly proposed protocols are not suitable for sensor networks, as the routing overhead consumes too much energy. We therefore extended the existing *ns-2* AODV implementation by some modifications discussed in the ZigBee specification [9], which concern the broadcast mechanism and the link layer feedback handling.

3. IMPLEMENTATION DETAILS

3.1 Sleep Scheduling

Minimizing energy consumption is a key challenge for any WSN deployment. Thus, in a non-beacon-enabled 802.15.4 WSN sleep scheduling has to be managed. As this is not specified for the non-beacon-enabled mode [2, 9], this problem has also not been considered by the existing *ns-2* implementations [5, 8]. The most intuitive idea of letting each node independently wake up for sending packets and going to sleep afterwards can't be applied, as in multihop networks most nodes have to relay packets for other nodes. We therefore decided upon another simple method and implemented the sensor node duty cycle as an on-off-process. Each node in the network is awake for the same fraction p_w of a constant time interval T and spends the rest in sleep state. $p_w = 100\%$ corresponds to an always-on node. To decide, *which* part of T the node is sleeping, we considered two different strategies: The *random* scheduling strategy starts the simulation by letting the nodes go active for $p_w T$ at randomly distributed times, then sleep for $(1 - p_w)T$ and so on. The *synchronized* scheduling strategy assumes, that the entire network is either on or off.

This fixed duty cycle is kept, expect for one situation: If a node is on the point of going to sleep, but still has a packet in the send queue, this packet is not discarded but sent, and therefore some time of the sleep period is cut off. Note that, in the random system, especially if $p_w < 0.5$, it can happen, that some nodes will not be able to communicate, as they are never awake, when their neighbors are. Moreover, the usage of randomly distributed starting points of sleep and wake phases introduces changing routes during the initialization process. This is not the case under the synchronized schedule, but the collision probability is increased in this case, as all nodes try to send during the same small activity period.

3.2 Routing

For routing in a multihop ZigBee mesh network, the ZigBee specification proposes an algorithm which is very similar to AODV [9]. For a first analysis, we thus take the existing *ns-2* AODV implementation [7] and include the proposed modification to jitter the route request broadcast. Moreover, link layer feedback (LLF) which is given after three not received acknowledgments, thus failed MAC layer retransmissions, is used to announce failed transmissions, upon which the link is considered as broken and the route error mechanism is started. This consists basically of starting possibly a local repair and sending out an error message to the neighboring nodes. However, the cause of the outstanding ACKs could not only be a dead, but also a sleeping destination or a packet collision which are quite frequent in dense or low duty WSNs. Thus, if messages are broadcasted after each LLF, these messages can easily flood the system, and decrease the system performance. We propose therefore *smooth* AODV with a modified LLF handling, as indicated within the ZigBee specification [9], and summarize this mechanism in Fig. 1. In contrast to the *regular* AODV, *smooth* AODV only assumes a link failure, if the number of LLFs n_c during a certain *guard interval* g does not exceed the *LLF threshold* L_T. We introduced these two parameters to tolerate occasional transmission failures and found, that they improve the system performance significantly.

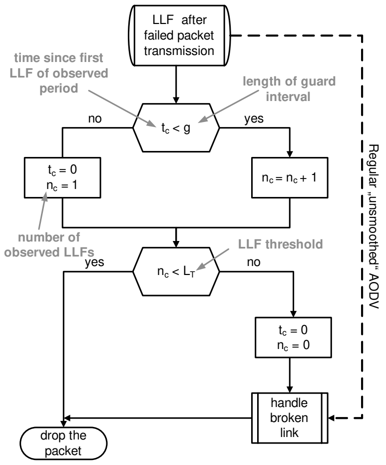


Figure 1: Modified LLF handling in smooth AODV

3.3 Simulation Setup

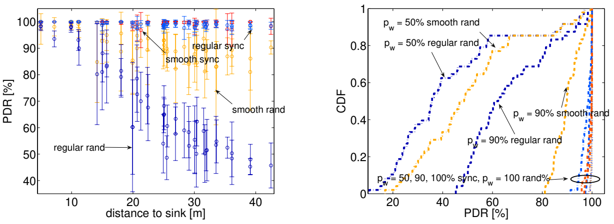
To investigate the performance of smooth AODV in a non-beaconed 802.15.4 multihop WSN, we used a grid layout of 49 nodes with an inter-node spacing of 5 meters. The PAN coordinator, also playing the role of the traffic sink, is in one of the grid's corners. We limited the radio range to 12 meters and assumed failure free transmissions, as our research is targeted on the interaction of MAC and routing layer. We used a simplified sensor node life cycle model and assumed, that each node is either transmitting, receiving which is equivalent to listening or sleeping and consumes a power of 35.28 mW, 31.32 mW or 0.144 μW respectively. These values have been taken from [5] and are representative for state-of-the art hardware. The duration of one simulation run was $t_{sim} = 10000$ sec, and we used $T = 1$ sec, $g = 12$ sec and $L_T = 3$. We assumed furthermore, that each node tries every $\Delta t = 50$ sec to send a data packet of size $s = 50$ byte to the sink.

4. SIMULATION RESULTS

The goal of this section is to investigate the performance of *regular* and *smooth* AODV for both *random* and *synchronized* sleep scheduling. The impact of the wakeup ratio p_w on the performance is investigated for the four different possible scenarios a) regular rand, b) regular sync, c) smooth rand, and d) smooth sync.

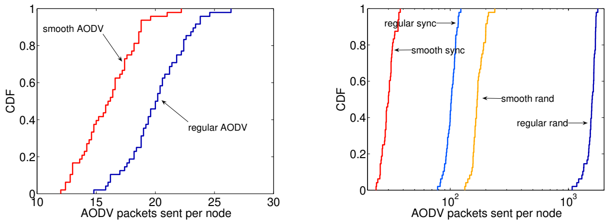
In order to quantify the performance of the system, we use the packet delivery ratio PDR which is defined as the ratio of received application datagrams at the sink and the sent application datagrams of a node, $PDR = N^{app}_{rcvd}/N^{app}_{sent}$. The resulting overhead required to find a path from a node to the sink is expressed by the number of sent AODV packets, N^{AODV}_{sent}. In this context, the used energy E_b per successfully received bit and the end-to-end (e2e) delay D are used. The latter one is the time from starting to send the application datagram until the time of successful reception.

Fig. 2 shows the packet delivery ratio for the four considered scenarios. In Fig. 2(a), the PDR for each sensor node in the spatial network layout is plotted depending on its distance to the sink for a wakeup ratio of p_w=90%. Each simulation run was repeated five times and the corresponding confidence intervals at a significance level of 95% are illustrated as errorbars around the average PDR values. If the



(a) p_w=90% online ratio (b) p_w = 50%, 90%, 100%

Figure 2: Packet delivery ratio PDR



(a) p_w=100% online ratio (b) p_w=90% online ratio

Figure 3: Sent AODV packets

nodes are synchronized, i.e. regular sync or smooth sync, the PDR is nearly 100% for each node independent of the node's distance to the sink, but smooth AODV shows a marginally better performance than regular AODV. If the nodes are active in an unsynchronized fashion, the PDR drastically changes. While smooth AODV still leads to a PDR larger than 80%, the PDR using regular AODV drops below 50%, although the nodes are online 90% of the time.

Considering different online times, expressed by a different p_w, yields the same result. Fig. 2(b) shows the cumulative distribution function (CDF) of the average PDR values for each node, as given by the dots in the previous scatter plot. We now vary p_w from 100%, to 90% and 50%. If the nodes are always on, i.e. p_w=100%, or the nodes are synchronized, then the PDR is about 100% with only slight differences. For p_w=90% the difference between smooth and regular AODV is very large with 23% on average. For shorter online periods with p_w=50%, the difference of the PDR between regular and smooth AODV narrows and leads to very small PDRs in the unsynchronized case. To challenge this problem, the nodes in the sensor network should try to coordinate and in the best case to synchronize their wake times.

An explanation for this dramatic decrease of the PDR is given by the number N_{sent}^{AODV} of sent AODV packets per node, as depicted in Fig. 3. First, we consider the CDF of N_{sent}^{AODV} when the nodes are always active. In that case, the synchronized and the unsynchronized scheduling scheme lead to the same results. As soon as a node has found a route to the sink, there is no need to change it anymore. Nevertheless, there is already a small difference between regular and smooth AODV. This is caused by dropped datagrams, which is indicated by dropped AODV packets, cf. Table 1. Smooth AODV does not try to find a new route for each failed data transmission and uses additionally the guard interval to smoothen its reaction. Reasons for dropped AODV packets are packet collisions, a bad link quality according to a too low link quality indicator (LQI), or system drops because of elapsed time-to-live counters.

Fig. 3(b) shows the CDF of the number of sent AODV packets for each of the 48 nodes in the network. Note that the x-axis is scaled logarithmically in this case. The first observation is that N_{sent}^{AODV} strongly varies between the four different scenarios: For route establishment using both regular AODV and smooth AODV, under the random sleep scheduling scheme, one order of magnitude more AODV packets than under the synchronized scheme are required. Observe, that smooth AODV decreases the number of sent AODV packets significantly. The average number of sent AODV packets per node is about a) 1557 for regular rand, b) 102 for regular sync, c) 173 for smooth rand, and d) 30 for smooth sync during $t_{sim} = 10000$ sec.

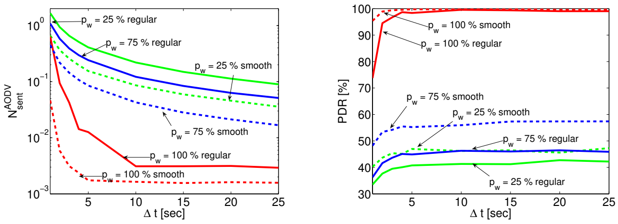
The second observation is that a high activity ratio of p_w=90% already requires a lot of AODV overhead if the nodes are unsynchronized and start their wake/sleep cycle at a random time instant. This means that for a random observer a node is offline with a probability of $1 - p_w$. For a route with H hops between source and sink, the probability that all nodes on the route are online is thus ${p_w}^H$. In the considered layout, we obtain for each path about 4 hops on average per node and about 7 hops at most. In the worst case, the packet is routed successfully with a probability of only ${p_w}^H$=0.48 for H=7 and p_w=0.90. Due to the emerging frequent route requests, the number of dropped AODV packets due to collisions or bad link quality increases which intensifies this effect even more, cf. Table 1. As a consequence, the packet delivery ratio will decrease even stronger in high load situations.

Next, we investigate this assumption by varying the offered load per node in the unsynchronized case and compare regular and smooth AODV in Fig. 4. Therefore, the interdeparture time Δt of two application datagrams at each sensor node is varied between 1 sec and 25 sec. Fig. 4(a) shows the normalized number $\hat{N}_{sent}^{AODV}$ of sent AODV packets depending on Δt for p_w=25%, 75%, 100%. The normalized number $\hat{N}_{sent}^{AODV}$ takes the online time of a node into account during which the AODV packets can be sent. It is $\hat{N}_{sent}^{AODV} = N_{sent}^{AODV} / (p_w \cdot t_{sim})$. Note that the y-axis is logarithmically scaled in Fig. 4(a). For p_w=25% and p_w=75%, $\hat{N}_{sent}^{AODV}$ is about 2.5 and 2.8 times larger for regular AODV than for smooth AODV, respectively. In both cases, the factor is independent of the time betwenn two packets, Δt. For p_w=100%, the smaller Δt, i.e. the higher the offered load, the larger is the difference between regular and smooth AODV. As expected, the normalized number of sent AODV packets is increasing, if p_w is decreasing. Fig. 4(b) shows the corresponding PDRs for the same scenarios. As mentioned above, the PDR is highly affected by the system load. For p_w=100%, smooth AODV softens significantly the impact of a high datagram frequency and reaches a PDR of 95%, while regular AODV results in a PDR of roughly 75%. For shorter online times, smooth AODV always outperforms regular AODV.

In Fig. 5, the sensitivity of smooth AODV is evaluated

Table 1: AODV packet collisions and LQI

		collisions		LQI	
		regular	smooth	regular	smooth
100%	un-/sync	991	733	752	577
90%	sync	5249	1958	4488	1006
	unsync	50215	9890	68693	6691
50%	sync	9369	2368	9276	1294
	unsync	22921	10690	32738	11936



(a) Sent AODV packets (b) PDR

Figure 4: Impact of traffic load

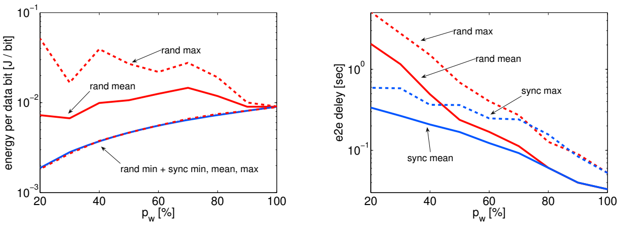
with respect to the wake time of a node. From application layer's point of view, the end-to-end delay D and the used energy E_b per successfully received bit are the important performance metrics. The latter one implicitly describes the PDR, as in our system the energy for sending and receiving is almost the same and therefore E_b mainly depends on the wake time $p_w T$ during the interval T. The energy required per successfully received bit can thus be estimated as

$$E_b = \frac{t_{sim} \cdot p_w}{PDR \cdot s \cdot t_{sim}/\Delta t} P. \tag{1}$$

The energy used during the online time is $t_{sim} p_w P$ with the average power consumption P for transmitting and receiving which is constant for all nodes with the same p_w. The number of successfully received bits at the sink is $PDR \cdot s \cdot t_{sim}/\Delta t$ with the packet size s.

Fig. 5(a) shows the minimum, maximum, and average E_b of all nodes in the layout. In the synchronized scenario, the PDR of each individual node is almost 100% and nearly all sent packets are received at the sink. Thus, there is no difference between minimum, maximum, and mean E_b. In the unsychronized scenario, the maximum PDR is almost 100% for nodes which are directly sending the application datagrams to the sink. However, there are datagramms which are routed over several hops before they reach the sink. Hence, the average and the maximum PDR over all nodes in the network depends on the actual wakeup ratio p_w. Accordingly, the used energy E_b differs for the individual nodes in the layout and the average and maximum value is much larger than in the sychnronized scenario.

The related e2e delays of the considered simulation scenarios are depicted in Fig. 5(b). The maximum and the average e2e delay of successfully delivered application datagrams is computed over all nodes in the system. As the wakeup ratio p_w impacts the amount of packet collisions and the resulting retransmissions of packets, the e2e delay decreases with an increasing activity ration p_w. For the same reason, the synchronized sleeping schedule shows a better performance than the unsynchronized one. A consequence of multi-hop routing is the difference between the average and the maximum delay. However, this difference vanishes with increasing p_w.



(a) Energy E_b per bit (b) End-to-end delay D

Figure 5: Impact of wakeup time on smooth AODV

5. CONCLUSIONS

In this paper, we simulated the combination of the sleep enabled non-beaconed mode of 802.15.4 and AODV routing. Two duty cycle schedules were compared: random scheduling and global synchronization. AODV used immediate and smooth - with several drops as trigger - link layer feedback for route maintenance. The performance of the synchronized system under low load was not seriously limited by the duty cycles, however, smoothing AODV reduced the routing overhead significantly. The unsynchronized system performance broke down to 50% already for sleeping only 25% of the time. Again, smoothing AODV improved the situation slightly. While the energy for a successfully transmitted information could be descreased with for a less active node in the synchronized case, the energy consumption in the unsynchronized case could not be reduced. We saw, that the variation of the energy consumption were even increased if the nodes are sleeping longer, i.e. some nodes may not be efficient enough to fulfill their role in the WSN if the duty cycle is cut down.

The results indicate the need for synchronization mechanisms for wireless sensor networks. Our future work will be dedicated to the evaluation of sleep scheduling mechanisms for 802.15.4 WSNs with regard to the overhead induced by in band signaling and the effects of concentrating the traffic load on a fraction of the bandwidth.

6. REFERENCES

[1] C. Gomez, P. Salvatella, O. Alonso, and J. Paradells. Adapting AODV for IEEE 802.15.4 Mesh Sensor Networks: Theoretical Discussion and Performance Evaluation in a Real Environment. In *WOWMOM '06*, Buffalo, NY, USA, June 2006.

[2] IEEE Computer Society. IEEE Standard 802.15.4: Wireless Medium Access Control and Physical Layer Specifications for Low-Rate Wireless Personal Area Networks, October 2003.

[3] K. Langendoen. Medium Access Control in Wireless Sensor Networks. In H.Wu and Y. Pan, editors, *Medium Access Control in Wireless Networks, Volume II: Practice and Standards*. Nova Science Publishers, 2007.

[4] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC3561, 2003.

[5] I. Ramachandran, A. K. Das, and S. Roy. Analysis of the Contention Access Period of IEEE 802.15.4 MAC. *ACM Trans. Sen. Netw.*, 3(1), 2007.

[6] V. Rao and D. Marandin. Adaptive Backoff Exponent Algorithm for Zigbee (IEEE 802.15.4). In *NEW2AN 2006*, St.Petersburg, Russia, May 2006.

[7] USC Information Sciences Institute. Network Simulator NS2. http://www.isi.edu/nsnam/ns.

[8] J. Zheng and M. Lee. Will IEEE 802.15.4 Make Ubiquitous Networking a Reality? *IEEE Comm. Mag.*, 42(6), June 2004.

[9] ZigBee Alliance. ZigBee Specification, December 2006.

Ratpack: Using Sensor Networks for Animal Observation

Jó Bitsch Link,
Klaus Wehrle
Distributed Systems Group
RWTH Aachen
first.last1.last2@rwth-aachen.de

Okuary Osechas,
Johannes Thiele,
Hanspeter Mallot
Cognitive Neuroscience
Department of Zoology
University of Tübingen
first.last@uni-tuebingen.de

ABSTRACT

The goal of this project is to describe the behaviour of rats. To study this behaviour, we will resort to the use of wireless sensor networks, monitoring various quantities that yield important information to complement current knowledge on the behavioural repertoire of rats. The challenges we face include data acquisition and processing on the one hand, as rat-borne sensor nodes will need to be small enough not to interfere with the rats' own activities, thus limiting the available memory and processing capabilities. Additionally, rats spend a significant amount of time underground, making data transmission and routing a very interesting challenge, for which we are currently developing novel strategies.

Categories and Subject Descriptors

C.3 [**Special-purpose and Application-based Systems**]: Real-time and embedded systems

Keywords

Sensor network, Animal observation, Rattus norvegicus, Sporadic connectivity

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. In this paper, we discuss our contribution to make this vision a reality. With the help of sensor networks, we plan to observe several aspects of rat behaviour.

Currently, we are equipping rats with standard sensor nodes (mica2dot) and a sensor suite consisting of light, audio, and acceleration sensors. Sensors are attached to laboratory rats with the help of 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. As an additional feature, it also has a reflective marker to allow optical tracking in a controlled lab setting, such as a maze. The long term goal is to attach (or even implant) the sensor nodes to wild rats; this will have consequences on the accessibility of the data.

In the wild, rats live in underground burrows and so radio propagation is very limited. Therefore, sensor nodes can only communicate when the rats carrying these sensors meet somewhere. As a result, the sensor nodes are only sporadically connected and the network topology is highly dynamic, making our deployment scenario significantly different from the typically envisioned static networks.

The remainder of this paper is structured as follows: first, section 2 discusses other deployments of sensor nodes and compares our deployment scenario to these. Section 3 describes the quantities we are interested in measuring and the type of information we hope to obtain from them, while section 4 discusses the impact of sporadic connectivity on sensor network algorithms and protocols. Finally, section 5 offers some concluding remarks on our work.

2. RELATED WORK

Recently, a considerable number of sensor networks have been deployed in the environment [8, 11, 13, 14]. Most of these deployments – except ZebraNet [8] – are static networks. In these, researchers placed sensor nodes at locations of interest and ensured that the nodes could communicate with each other and the base station.

The ZebraNet project equipped zebras with customised sensor nodes. Via GPS, the sensor nodes recorded the animal's position and other relevant quantities. Furthermore, the sensor nodes recorded when and where zebras met. From this data, biologists could evaluate the movements and social interactions of zebras.

Our deployment scenario is somewhat similar to ZebraNet, but has a number of interesting differences: (1) we cannot use GPS for the observation of rats, as they live under the surface; (2) due to the small size of the rats, we need to use standard sensor nodes without large batteries or several MB of storage space, as was the case in the ZebraNet project. The GPS signal in the ZebraNet project provides absolute and accurate time and location information. Slotted medium access and routing benefit heavily from this knowledge. As a result, many of the research challenges we discuss in this paper are not relevant in the ZebraNet project.

The DTAG project [7] has an application that relates to ours in certain aspects but again differs in others. A very interesting aspect is that whales spend most of their time (up to 95%) underwater, making constant radio communication impossible, and thus, the scenario is in a way similar to ours. Their approach was to build a tag that records the relevant data onto a digital tape; when the data storage is full, the tag separates from the whale and floats to the surface, where it is picked up by the research team. This solution is not viable for our purpose since our motes need to stay attached to the rats as the danger of losing them in the burrow system is too high. Furthermore, our approach has the advantage of automatic data collection (as opposed to the manual collection of the tapes), thus speeding up the availability of the collected data.

3. SENSING PRINCIPLES

The scheme we propose for studying the behaviour of rats differs from existing methods in that it allows us to monitor rats directly in their natural environment (as opposed to traditional laboratory experiments). One fascinating prospect is the possibility of studying the structure of a burrow with a minimally invasive method, as this was previously done by excavating existing ones, thus disturbing the natural course of its inhabitants. Should our scheme be accurate enough, it would allow us to describe burrows, not only without disrupting their everyday life (as, for example, in [1]), but also in near-real-time as they are being built.

3.1 Social Interaction

Norway Rats live in burrows, usually shared in groups, which naturally leads to the formation of social hierarchies [1]. The communication between these rats is partially based on ultrasound vocalisations Several scenarios of interaction between rats are already known (mother/child, resident/intruder). Our setup should allow us to verify these and find new situations. The main tool for this approach will be the analysis of the vocalisations emitted by rats.

3.2 Motion

The motion of a rat may enable us to describe its foraging habits, as well as the layout of its burrow. This may also allow us to draw conclusions as to the actual use of different sections of the burrow, in a non-destructive fashion. The first approach to this problem will be through inertial measurements (acceleration, turn rates), but we expect to require further sensing principles for more accurate descriptions of the motion paths.

3.3 Activity

Sleeping and eating habits could be of interest as indicators of energy consumption. For example instead of a description of the seasonal variations in the rats' metabolism, it should be possible to obtain a higher time resolution. Activity monitoring could be accomplished by complementing the motion information (see section 3.2) with heart rate and breathing frequency data.

3.4 Higher Level Description

From the behavioural aspects described above, more abstract concepts can be inferred; we expect to exploit the synergy between different types of data, so we can interpret behavioural patterns in more abstract concepts. For example, the detection of vocalisations from infant rats, followed by movements previously determined to be characteristic of a mother carrying a child, might hint to a fostering behaviour. These vocalisations could then be conferred a certain interpretation, in the previously described case they could be thought of as cries for help.

4. SPORADIC CONNECTIVITY

Currently the main research focus in the sensor network community is on continuously connected sensor nodes. Thus, although the network topology may vary slightly over time, for example, due to node failure or changing radio conditions, the network infrastructure mostly remains the same. Today's algorithms and protocols such as Medium Access Control (MAC), routing, and data aggregation focus on this static scenario. The requirements of our own scenario quickly made obvious that the available algorithms and implementations are not efficiently usable for the following reasons.

4.1 Medium Access Control

Medium Access Control in a sporadically connected network, especially in a sensor network, should have two modes of operation: (1) an ultra low power beacon mode and (2) a high throughput mode. In the beacon mode, two nodes can find each other by periodically sending beacon messages and listening for such messages from other nodes. Once two nodes find each other, they switch to the high throughput mode to exchange data. Existing MAC protocols such as [12, 15] do not provide this functionality.

4.2 Routing

As we do not expect to know all exits of a rat burrow and some rats may stay in the burrow for long durations, we need the sensor nodes to exchange their measurements. Today's tree-based routing protocols [10] or even new any-to-any versions [4] are not suitable for this purpose. Similar to delay tolerant networks [3], data should be relayed from one sensor node to another when their bearers, i.e. the rats, meet. We place a base station at one (or more) exits of the rat burrow. When a rat passes along this exit, all measurements, e.g. data collected by this rat as well as the data received from other rats, are transmitted to the base station.

4.3 Data Aggregation

Sensor nodes have very limited storage space, typically 4 kB of RAM and about 500 kB of additional flash space [6]. As discussed, it may take some time until a certain rat passes one of the base stations. Thus, its sensor node needs to store large amounts of measurement data – its own and those of the rats it has met. Efficient high-level data aggregation is necessary to reduce storage requirements and the communication overhead when two rats meet.

4.4 Time Synchronisation

Accurately synchronised clocks on the sensor nodes ensure consistent time stamps and measurements for the distributed observation of events. Nonetheless, typical time synchronisation algorithms [2] assume continuously connected nodes and are thus not applicable.

4.5 Reprogramming

The lessons learned during the deployment may result in changing application needs and therefore require flexible schemes for reprogramming sensor nodes. We expect modular and flexible communication protocols [9] and operating systems [5] to be very beneficial in our application scenario.

5. CONCLUSION

At first glance, the merits of a software architecture required for our deployment may not be very obvious – implementing a software that can (1) record when two rats meet and (2) record some additional sensor readings of temperature and motion seems to be straightforward. However, when looking at the presented scenario, it becomes obvious that the necessary communication paradigms for sporadically connected networks are missing in the sensor network community.

In this paper, we discussed the features such a communication paradigm should provide for efficient and energy-aware animal observation. Currently, our ongoing work focuses on designing and implementing the required features. The main deployment scenario is rat observation. However, we think that this architecture can be easily adapted to the observation of many other species such as Flying Foxes or Naked Mole Rats (Heterocephalus glaber), as their social interaction is highly complex. Newly available platforms have become sufficiently small to make it seem plausible to even study smaller bats.

6. ACKNOWLEDGEMENT

We would like to thank Microsoft Research for the financial support of our project as part of their programme "Towards 2020 Science".

7. REFERENCES

[1] *The Ecology and Sociobiology of the Norway Rat.* Bethesda, Maryland, 1962.

[2] J. Elson and K. Römer. Wireless Sensor Networks: A New Regime for Time Synchronization. *ACM SIGCOMM Computer Communication Review*, 2003.

[3] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proc. of the conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM)*, 2003.

[4] R. Fonseca, S. Ratnasamy, D. Culler, S. Shenker, and I. Stoica. Beacon Vector Routing: Scalable Point-to-Point in Wireless Sensornets. In *Proc. of Symposium on Networked Systems Design and Implementation (NSDI)*, 2005.

[5] C.-C. Han, R. K. Rengaswamy, R. Shea, E. Kohler, and M. Srivastava. SOS: A dynamic operating system for sensor networks. In *Proc. of the International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.

[6] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. of conference on Architectural support for programming languages and operating systems (ASPLOS)*, 2000.

[7] M. Johnson and P. Tyack. A Digital Acoustic Recording Tag for Measuring the Response of Wild Marine Mammals to Sound. *IEEE Journal of Oceanic Engineering*, 2003.

[8] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with zebranet. In *Proc. of conference on Architectural support for programming languages and operating systems (ASPLOS)*, 2002.

[9] O. Landsiedel, J. Bitsch, K. Denkinger, and K. Wehrle. Modular Communication Protocols for Sensor Networks. In *Proc. European Workshop on Wireless Sensor Networks (EWSN)*, 2006.

[10] P. Levis, S. Madden, D. Gay, J. Polastre, R. Szewczyk, A. Woo, E. A. Brewer, and D. E. Culler. The emergence of networking abstractions and techniques in tinyos. In *Proc. of Symposium on Networked Systems Design and Implementation (NSDI)*, 2004.

[11] J. Paek, K. Chintalapudi, J. Caffrey, R. Govindan, and S. Masri. A Wireless Sensor Network for Structural Health Monitoring: Performance and Experience. In *Proc. of Workshop on Embedded Networked Sensors (EmNets)*, 2005.

[12] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proc. of Conference on Embedded Networked Sensor Systems (SenSys)*, 2004.

[13] R. Szewczyk, J. Polastre, A. Mainwaring, and D. Culler. Lessons from a Sensor Network Expedition. In *Proc. of European Workshop on Wireless Sensor Networks (EWSN)*, 2004.

[14] G. Werner-Allen, J. Johnson, M. Ruiz, J. Lees, and M. Welsh. Monitoring Volcanic Eruptions with a Wireless Sensor Network. In *Proc. of European Workshop on Wireless Sensor Networks (EWSN)*, 2005.

[15] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proc. of conference on Computer Communications (InfoCom)*, 2002.

Wireless Sensor Networks for Environmental Noise Monitoring

Silvia Santini
Institute for Pervasive Computing
ETH Zurich, Switzerland
santinis@inf.ethz.ch

Andrea Vitaletti
Department of Computer Science
University of Rome “La Sapienza”, Italy
andrea.vitaletti@dis.uniroma1.it

ABSTRACT

While environmental issues keep gaining increasing attention from the public opinion and policy makers, several experiments demonstrated the feasibility of wireless sensor networks to be used in a large variety of environmental monitoring applications. Focusing on the assessment of environmental noise pollution in urban areas, we provide qualitative considerations and preliminary experimental results that motivate and encourage the use of wireless sensor networks in this context.

Keywords

Wireless sensor networks, environmental noise, noise indicators

1. INTRODUCTION

In 1996, the European Community estimated in about 80 millions the number of its citizens that were exposed to unacceptable levels of environmental noise, while another 170 millions suffered serious annoyances from high noise pollution during daytime [2]. Directive 2002/49/EC of the European Parliament has since made the avoidance, prevention, and reduction of environmental noise a prime issue in European policy, requiring member states to firstly determine the exposure of its citizens to environmental noise, and secondly, to ensure that information on environmental noise and its effects is made available to the public [1].

According to the directive, Member States are required to provide an accurate mapping of environmental noise exposure in urban areas like public parks, schools, hospitals, and other noise-sensitive zones, starting from June 2007. While current noise maps are mostly based on sparse data and ad-hoc noise propagation models, a recent position paper by the Commission [6] has stressed that *“every effort should be made to obtain accurate real data on noise sources,”* since *“detailed noise modelling/mapping and noise exposure assessment may have to be undertaken in order to produce detailed local action plans.”* The demand for accurate data about noise exposure levels will likely increase dramatically, as this statement makes its way into mandatory regulation. We believe that wireless sensor networks will be able to satisfy this demand by providing noise measurements with an accuracy and cost-efficiency that current noise assessment procedures cannot afford.

Wireless sensor networks have already been used in a large variety of environmental monitoring applications, e.g., to monitor bird habitats and habits [8, 11], to investigate the growth model of redwood trees [5], or to study the influence of environmental parameters on the quality of agricultural products [9]. Wireless sensor networks also allow monitoring pollution parameters in urban areas at an accuracy and scale that were previously unreachable, e.g., within the CitySense[1] testbed, which plans to use a fixed network of 100 line-powered wireless sensors to collect fine-grained air pollution data and deliver it in real-time to the users. To the best of our knowledge, however, wireless sensor networks have so far not been used to perform fine-grained measurements of noise pollution levels.

This article presents a preliminary assessment for using wireless sensor nodes to measure noise exposure levels in urban settings. After briefly summarizing today’s environmental noise assessment procedures, we give an overview of the various quantities involved in measuring environmental noise. We close with initial experimental results of using a sensor node to compute these quantities.

2. NOISE POLLUTION ASSESSMENT TODAY

Today’s noise measurements in urban areas are mainly carried out by designated officers that collect data in a location of interest for successive analysis and storage, using a sound level meter or similar device. This manual collection method using expensive equipment does not scale as the demand for higher granularity of noise measurements in both time and space increases. Instead, a network of cheap wireless sensor nodes deployed over the area of interest could collect noise pollution data over long periods of time, and autonomously report it to a central server though the sensor’s on-board radio, requiring human intervention only to install and subsequently remove the sensing devices. Moreover, since sensor nodes are typically equipped with several different sensors, they can label the collected noise data with additional information like, e.g., the temperature and humidity values registered as the noise measurements were collected. This information must indeed be provided for any properly collected set of noise exposure data, along with other meteorological parameters like wind speed and direction.

Collected noise data is typically stored in a land register and used, together with additional information about exist-

---

[1] http://www.citysense.net/

ing noise sources, to feed computational models that provide extrapolated noise exposure levels for those areas for which real data is unavailable. Even if this assessment procedure is still compliant with European regulations, today's computational models often fail to provide accurate estimations of the real noise pollution levels[2]. Indeed, while the free propagation properties of noise generated from typical noise sources[3] are well understood, shadowing and reflection effects hinder accurate estimation of noise levels in complex urban settings. For instance, estimated noise levels on internal buildings façades (e.g., facing a courtyard) are typically unreliable, and this inaccuracy may become critical if noise exposure data is used to drive decisions about construction planning or to elaborate local noise abatement policies. The accuracy of estimated noise levels could be easily verified and improved by installing a wireless sensor network at those locations for which computational models are likely to provide inaccurate estimations. In these settings, noise assessment points must be closely spaced (about every 2 to 3 meters), and measurements should be taken simultaneously at all assessment points in the presence of sound from a noise source. While this distributed sensing setup is extremely hard to realize with current measurement procedures, it is a "natural" setup for wireless sensor networks.

Wireless sensor networks may bring significant improvements also in the assessment of noise pollution due to vehicular traffic on urban roads. The current procedure requires estimating, for several different vehicle classes, the average number of units passing-by at daytime, evening and night and the average noise level for each vehicle pass-by [4]. This estimation is either performed through computation, with the drawbacks and problems outlined above, or it is performed manually, i.e., by a designated officer standing nearby the road and annotating the type and number of vehicles passing-by. Wireless sensor networks have already proved their ability to detect and classify vehicles [3] and could therefore be used in this context to automate the vehicle counting procedure and, at the same time, record the corresponding noise levels.

3. MEASURING NOISE

Acoustic waves are pressure fluctuations, usually caused by a solid vibrating surface, that propagate through an appropriate medium like air or water. Sound is the sensation induced at the human ear by incident acoustic waves that are captured and converted into neurological stimuli by the hearing system. Similarly, a microphone converts pressure fluctuations into an equivalent electrical signal, that can be post-processed to compute the loudness of the noise source that generated the acoustic wave. Average loudness levels over long periods of time are commonly used as *noise indicators.* For instance, the European directive 2002/49/EC requires Member States to apply the L_{den} and L_{night} indicators for the preparation and revision of strategic noise mapping [1]. Before getting to the formal definition of these indicators, we need to explain how the equivalent sound pressure level of a noise source can be computed from the output signal of a microphone [4].

The instantaneous sound pressure level (SPL) of a sound is usually expressed in logarithmic units with respect to a given reference pressure level and is computed according to the following equation:

$$L_p(t) = 10\log_{10}\frac{p(t)^2}{p_{ref}^2} = 10\log_{10} p(t)^2 - 10\log_{10} p_{ref}^2 \quad \text{(dB)} \tag{1}$$

in which $p(t)$ represents the instantaneous pressure of an acoustic wave impinging the membrane of the microphone. The standard reference pressure p_{ref} is $20\mu Pa$ and conventionally represents the minimum audible sound. Substituting this value into equation 1, one obtains:

$$L_p(t) = 10\log_{10} p(t)^2 + 94 \quad \text{(dB)}. \tag{2}$$

If $E(t)$ is the microphone output voltage induced by an incident acoustic wave $p(t)$, equation 2 may be rewritten as:

$$L_p(t) = 10\log_{10} E(t)^2 + 94 - S \quad \text{(dB)} \tag{3}$$

The sensitivity S of the microphone that appears in equation 3 defines how the microphone responds to a certain pressure input and is typically expressed in decibel with respect to a reference level. The following equation holds for the sensitivity S of a microphone:

$$S = 20\log_{10}\frac{Ep_0}{E_{ref}p} \quad \text{(dB)} \tag{4}$$

It is common practice to set $E_{ref} = 1V$ and $p_0 = 1Pa$ and thus to express the sensitivity as a (negative) value with respect to the reference value: 0 dB = 1V/Pa.

Since noise typically fluctuates significantly even over short periods of time, the instantaneous sound pressure level as defined in 3 is of little practical relevance. The loudness of a given noise source is therefore better represented by the average of the time-varying sound pressure level $L_p(t)$ over a given period of time T:

$$L_{eq} = \frac{1}{T}\int_0^T 10^{\frac{L_p(t)}{10}}\,dt \quad \text{(dB)} \tag{5}$$

The equivalent sound level pressure L_{eq} defined above is the quantity that is typically measured by a sound level meter and that drives the computation of most commonly used noise indicators. For instance, the L_{day}, $L_{evening}$ and L_{night} noise indicators are the equivalent sound levels averaged over day, evening and night periods[4]. The L_{den} (day-evening-

---

[2] The authors are indebted to Hans Huber and Fridolin Keller of the department for environmental noise protection of the city of Zurich, who pointed this out in a personal in-depth interview.

[3] Typical noise sources are, e.g., human activities, motor vehicles, railways, aircrafts or industrial machinery.

[4] Accurate definition of these indicators is provided in ISO

night) level is accordingly defined as:

$$L_{den} = 10\log_{10}\left[\frac{1}{24}\left(12\cdot 10^{\frac{L_d}{10}} + 4\cdot 10^{\frac{L_e+5}{10}} + 8\cdot 10^{\frac{L_n+10}{10}}\right)\right] \tag{6}$$

where we abbreviated L_{day}, $L_{evening}$ and L_{night} to L_d, L_e and L_n, respectively[5]. For the purpose of noise mapping, Member States of the European Community must provide noise pollution data in terms of the L_{den} and L_{night} indicators [1].

Even if there are several different standard procedures for the computation of the above defined noise indicators, a few important issues like, e.g., the spatial distribution of the measurement points or the necessary acoustic data processing, must always be carefully considered when measuring sound pressure levels. Indeed, since the human ear does not respond equally to all the frequencies in the audible range, measured sound levels must be adequately weighted (in the frequency domain) to take into account this selective behavior of the human hearing system. Among the available standard weighting methods the European regulation requires the use of the A-weighting function, originally defined in [7] and then adopted in numerous international standards. Even if A-weighting should better be performed using an analog filter before sampling, digital post-sampling filtering is also tolerated, even if typically less accurate. The A-weighted sound level pressures and noise indicators are indicated in A-weighted decibels or dB(A).

When assessing noise indicators, the location of the measuring devices must follow clearly defined rules [1, 6]. In particular, for the purpose of noise mapping near to buildings, the assessment points must be 4.0 ± 0.2 m above the ground and at the most exposed façade. If necessary, other heights may be used but they shall never be less than $1,5$ m above the ground, and results should be corrected in accordance with an equivalent height of 4 m. This requirement assumes particular importance since it disqualifies portable hand-held devices like mobile phones to be used for noise pollution measurements.

4. PRELIMINARY RESULTS

To be used as noise pollution sensing devices, wireless sensor nodes must be able to compute the noise indicators defined in the previous section. We therefore performed a preliminary study to understand the feasibility of currently available sensor networks platforms to compute such indicators. We used the Tmote Sky prototyping platform[6], equipped with the SBT80 multi-modality sensor board available from EasySen[7]. This sensor board features, among other sensors, the EM6050P-423 omni directional condenser microphone, which we used to capture audio signals from the environment. The output voltage of the microphone is quantized using the Tmote Sky's 12 bits analog to digital converter (ADC). The voltage levels recorded by the microphone may be reconstructed from the ADC samples using the simple formula: $E = \frac{E_{ADC}}{4096}\cdot V_{ref}$, where V_{ref} is set to 2.5V for the Easysen sensor board and E_{ADC} are the ADC samples. Figure 1 reports the (A-weighted) voltage response of the microphone to a synthetically generated 250Hz sine wave stimulus, whose amplitude has been progressively increased to bring the microphone to saturation. The raw signal samples, read from the ADC at a a 2kHz rate, has been sent from the node to the pc through the serial port. We post-processed the data in Matlab to compute the A-weighted equivalent SPL, which is 90 dB(A).

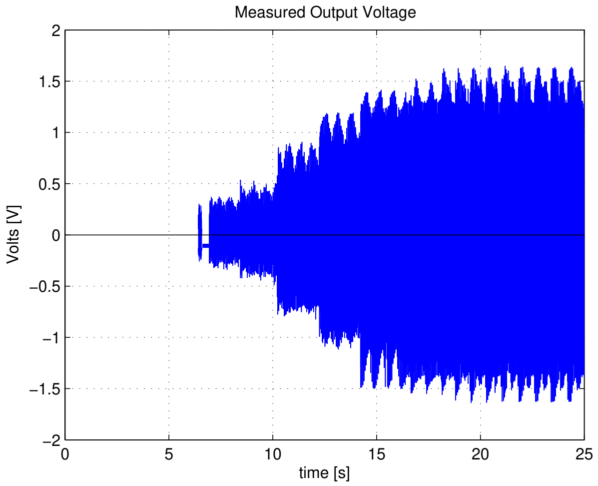


Figure 1: Microphone response to a 250Hz sine wave stimulus of increasing amplitude.

The first consideration we can derive by observing the sample data in figure 1 is the high level of background noise. The EM6050P-423 has indeed a self-noise level of 54 dB(A) SPL[8], which makes sounds corresponding to SPL levels below 54 dB(A) to be indistinguishable from electrical background noise[9]. Since this high level of background noise seriously limits the applicability of the EM6050P-423 for our purposes, we are considering using alternative acoustic sensors like e.g., the Tmote Invent[10] on-board microphone or a custom made sound level meter. The EM6050P-423 has indeed further suboptimal characteristics. First, its frequency response start deviating from linearity already at 5kHz and distorts significantly harmonic components above 10 kHz. Second, the EM6050P-423 datasheet does not report the maximum SPL the microphone can measure without significant total harmonic distortion (THD). This parameter is nevertheless necessary to define the upper bound of the dynamic range of the microphone and without a specification it is unclear up to which SPL level the microphone may

---

1996-1:2003 (that very recently replaced the currently withdrawn ISO 1996-2:1987 standard).

[5] L_{den}, L_d, L_e and L_n are all expressed in dB.

[6] www.moteiv.com

[7] www.easysen.com

[8] The self-noise (or equivalent noise) level may be computed subtracting the nominal signal to noise ratio (S/N) from the reference sound pressure level of 94dB. For the EM6050P-423 the S/N is 40 dB.

[9] 54 dB(A) is the SPL level that corresponds to the noise produced during a normal conversation taking place at about 1 m distance from the microphone.

[10] www.moteiv.com

provide a reliable response.

Besides the electrical and acoustic characteristics of the microphone there are other issues influencing the quality of the performed noise measurements like the rate at which voltage samples are read from the ADC. Since the human ear can only perceive acoustic waves whose frequencies range from 20 Hz to 20 kHz, sampling the output signal of a microphone must occur at at least 40 kHz [4,10]. In the context of noise pollution measurements this sampling frequency may be reduced 32 kHz since the hearing system of adult humans cannot perceive frequencies above 16 kHz [4, 8]. Nevertheless, this sampling rate seems prohibitively high for sensor network platforms, which typically rely only on limited computational resources. However, as long as no radio communication is involved, the Tmote Sky is able to support the required sampling rate. Since noise indicators are are long-term SPL averages, transmitting raw data back to a central sensor is not necessary as the aggregated noise indicators are of interest and not the time-varying pressure levels. Furthermore, sensor nodes must not necessarily continuously sample the acoustic levels, but they can apply intelligent data collection techniques to estimate noise indicators in accordance to pre-specified accuracy requirements. Limiting radio communication and data acquisition, the power consumption of the sensor nodes can be adequately controlled to allow long-term, unattended network operation.

In order to rapidly get first quantitative data we neglected calibration issues, that should otherwise be considered carefully. Sensor nodes needs indeed to be opportunely calibrated in order to be used as noise pollution sensors. The calibration procedure may be carried out using a pistonphone, i.e., a device that generates (in anechoic conditions), well-defined sound pressure levels. Measuring discrepancies between the effective microphone response and the expected (ideal) response would allow to adequately tune individual sensor nodes gains to make them provide reliable SPL measurements. Furthermore, in-network calibration methods could be investigated to ensure long-term correct network operation.

Even if the way to the first "wireless noise sensing network" must still pass through several important milestones, the considerations and the preliminary results reported in this paper show that using wireless sensor networks for environmental noise monitoring is not only technically possible, but would also bring significant advantages with respect to the current assessment procedures. We will therefore concentrate our future work in building a reliable hardware/software prototype, and in testing it extensively in the real, noisy world.

5. REFERENCES

[1] Directive 2002/49/EC of the European Parliament and of the Council of 25 June 2002 relating to the Assessment and Management of Environmental Noise.

[2] Future Noise Policy. European Commission Green Paper. COM (96) 540 final, November 1996.

[3] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y. Choi, T. Herman, S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A Line in the Sand: A Wireless Sensor Network for Target Detection, Classification, and Tracking. *Computer Networks (Military Communications Systems and Technologies)*, 46(5):605–634, December 2004. Military Communications Systems and Technologies.

[4] D. A. Bies and C. H. Hansen. *Engineering Noise Control: Theory and Practice.* Spon Press (Taylor &Francis Group), London and New York, 3rd edition, 2003.

[5] P. Buonadonna, D. Gay, J. M. Hellerstein, W. Hong, and S. Madden. TASK: Sensor Network in a Box. In S. B. Erdal Cayirci and P. Havinga, editors, *Proceedings of the Second IEEE European Workshop on Wireless Sensor Networks and Applications (EWSN'05)*, Istanbul, Turkey, February 2005.

[6] European Commission Working Group Assessment of Exposure to Noise (WG-AEN). Good Practice Guide for Strategic Noise Mapping and the Production of Associated Data on Noise Exposure, January 2006.

[7] H. Fletcher and W. A. Munson. Loudness, its Definition, Measurement and Calculation. *Journal of the Acoustic Society of America*, 4(2):82–108, 1933.

[8] B. Greenstein, C. Mar, A. Pesterev, S. Farshchi, E. Kohler, J. Judy, and D. Estrin. Capturing High-Frequency Phenomena Using a Bandwidth-Limited Sensor Networks. In *Proceedings of the 4th ACM Intl. Conf. on Embedded Networked Sensor Systems (SenSys'06)*, Boulder, Colorado, USA, November 1–3 2006.

[9] Sensor Network in a Vineyard. GoodFood EU Integrated Project: Food Safety and Quality Monitoring with Microsystems. Project Website: `www3.unifi.it/midra/goodfood/`.

[10] C. E. Shannon. Communication in the Presence of Noise. *Proc. Institute of Radio Engineers*, 37(1):10–21, January 1949.

[11] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. An Analysis of a Large Scale Habitat Monitoring Application. In *Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems (SenSys'04)*, Baltimore, Maryland, USA, November 3-5 2004.

Security in Pervasive Healthcare

Oscar Garcia Morchon, Heribert Baldus
Philips Research Europe
Weisshaussstrasse 2
52062 Aachen, Germany
firstname.lastname@philips.com

Tobias Heer, Klaus Wehrle
Distributed Systems Group
RWTH Aachen University
52074 Aachen, Germany
lastname@cs.rwth-aachen.de

1. INTRODUCTION

Pervasive healthcare is an emerging application in which wireless sensor networks (WSNs) are going to be used in medical environments. In this context, patients will be outfitted with wearable wireless sensors, forming a body sensor network (BSN) that will sense, process, and transmit patients' vital signs to their caregivers.

In a first step, this technology will be introduced in medical centers, replacing existing wired telemetry systems and improving the flexibility of health monitoring systems. Later, pervasive healthcare will extend its scope to home monitoring, allowing applications such as monitoring of elderly people. Finally, pervasive healthcare may enable ubiquitous patient monitoring. In this vision, which is pursued by health alliances, such as Continua [1], everybody will carry a BSN which may enable real time vital sign monitoring, independently of the positioning of patients and caregivers.

Security is a fundamental part of this pervasive healthcare architecture. The demand for security is driven by the responsibility of the caregiver for the patient as well as by legal laws such as the Health Insurance Portability and Accountability Act (HIPAA) in the USA [1] or the European Directive 95/46 [3] on the protection of personal data. In this context, the provision of basic security services, such as, e.g., confidentiality, authentication, integrity and availability depends upon the definition of a consistent key management architecture as it manages the cryptographic secrets that are necessary to enable secure communication between the sensor nodes and monitoring devices. However, the design of a key-management architecture in the framework of pervasive healthcare is not straightforward due to inherent technical restrictions and operational requirements.

On the one hand, the resource-constrained nature of medical sensor nodes, e.g., the MICAz platform [3], which is used in some pervasive healthcare projects (e.g. Codeblue [5]), presents computational, communication, memory and energy constrains.

On the other hand, pervasive healthcare architectures introduce novel operational requirements that make key management challenging.

- Access to infrastructure like a centralized key-management-service cannot be guaranteed at all times, and thus, sensor nodes must be able to establish a secure communication in an ad hoc manner. However, whenever sufficient network coverage is provided (as it would be the case in a hospital), timeliness and reliability are crucial.
- Pervasive healthcare architectures need to be highly scalable. Consequently, key management solutions must be scalable as well in order to guarantee global coverage and to support typical workflows and patient-caregiver relationships in medical facilities.
- Security solutions must take into account the distinct ways of communication in a BSN scenario: Intra-BSN communication between the nodes of a BSN and inter-BSN communication with nodes of other BSNs or medical monitors.
- Pervasive healthcare architectures must also address the problem of bridging the distance between several BSNs and healthcare monitors whenever patients leave medical facilities. Other network technologies need to be utilized in order to allow remote monitoring.
- Privacy concerns appear and aggravate due to the ubiquitous nature of pervasive healthcare applications whenever a patient leaves the medical facility.

In this context, the definition of an appropriate key management architecture for pervasive healthcare must address four specific problems, namely (1) key distribution within a BSN, (2) scalability, (3) secure BSN management, (4) secure ubiquitous connection of BSNs with medical facilities, and (5) protection of the patients' privacy. This paper addresses all of these aspects and discusses challenges and solutions in respect of the particular conditions in pervasive healthcare.

2. KEY DISTRIBUTION

Key distribution refers to the mechanisms used to distribute secrets and to set up trust relationships in a network. Hence, key distribution is of paramount importance in order to enable basic security services in pervasive healthcare applications. The design of a key distribution approach must consider both technical restrictions and operational requirements.

Research on key distribution for WSNs shows that the use of public key based solutions in WSNs is feasible at the cost of high memory, energy, and computational requirements. Experimental results [6] show that efficient public key cryptosystems based on elliptic curve cryptography (ECC) still require about 0,81 second for a 160-bit ECC point multiplication. This result might be satisfactory in static wireless sensor networks, but not in pervasive healthcare in which key agreement takes place very frequently due to the high mobility of sensor nodes. A public key based solution would rapidly exhaust sensor node's batteries and lead to unacceptable delays in the transmission of patient's vital signs.

Online trust centre alternatives, such as, e.g., SPINS [7] or S2RP [8] are also not feasible, since continuous access to infrastructure is not guaranteed in most BSN scenarios. Key pre-distribution schemes (KPSs) [9] seems to be one of the most promising solution to enable key management in pervasive healthcare due to their low resource requirements and operational flexibility.

In [13], we proposed the use of a key-distribution scheme that allows any pair of sensor nodes to generate a pair-wise key from a relatively small set of keying material. The keying material consists of a set of polynomials, which is distinct for every node. Two nodes can generate a pair-wise key from their sets of polynomials that cannot be reproduced by other nodes that own a different set of polynomials. This key distribution and key generation scheme forms a suitable foundation for securing the inter- and intra-BSN communication. The key distribution scheme is scalable and allows to securely connecting millions of nodes without the utilization of a centralized key management service. Nevertheless, its resiliency must be improved and needs to be enriched with suitable identification, authentication, and privacy protection features.

3. SECURE GROUP MANAGEMENT

BSNs are small clusters of sensors that connect in an ad hoc manner. These clusters collaborate to optimize measurements, detect intruders, aggregate data, and minimize energy consumption [8]. Hierarchical clustering of sensor nodes into different domain allows reducing the overall complexity of the system and makes the relationships between sensor nodes manageable. Therefore, clustering is an important step towards a scalable and manageable system.

Key distribution techniques allow sensor nodes, which are picked from a pool of nodes, to establish a secure link. However, that is just a part of the key management architecture as according to the definition of key management in [10], key management must support the establishment and the maintenance of keying relationships between authorized parties, including initialization of nodes in a security domain (SD), distribution, update, revocation and storage of keying material. Therefore, solutions to enable the dynamic formation of a BSN, in the sense of an independent SD, are required. This must include solutions to allow the entrance, leaving and the revocation of misbehaving devices within a BSN.

Current solutions such as BLIG [11] require human intervention to establish a BSN at network layer. Distributed device and key revocation solutions [17] are not applicable since they have been specially designed to revoke nodes (and keys) in systems based on a random pairwise KPS. Therefore, they do not perfectly fit other key distribution approaches. Additionally, they work under strong assumptions such as static networks, or the instantaneous deployment of nodes.

4. BRIDGING BSNs

BSNs are not only useful when used in a controlled environment like hospitals and medical practices. Due to the small size and low weight of the sensors, these networks are especially suited to be worn outside of medical facilities during a patient's everyday life. The sensors can record the patients' vital signs during their daily activity and, therefore, provide detailed information that cannot be obtained in a hospital. However, ubiquitous patient monitoring raises several technical problems.

As the communication range of the sensors in a BSN is very limited, direct communication is not always suitable for inter-BSN communication. Hence, other networking technologies are required to connect a BSN to a remote monitor in a transparent way. A special mobile gateway device that receives data from the sensors and transmits it over other wireless access technologies can be used to bridge the potentially large distance between a remote patient and the healthcare monitor on-site.

Modern mobile phones can use various access technologies like GSM, UMTS, and Wi-Fi to connect to the Internet and, thus, to a remote healthcare monitor. Therefore, these devices are suited as mobile gateways. Seamless handovers between different networking technologies and, therefore, between different addressing and routing paradigms are required to allow flexible and reliable monitoring of patients. GSM, for example, may not be available in all parts of a medical facility but IEEE 802.11-based wireless networks may be present. In urban areas, UMTS is an option for transmitting patients' vital signs with high bandwidth while a GSM network might be the only option in rural areas. Moreover, with the requirement for high reliability in mind, utilizing these technologies in parallel makes traditional networking protocols inapplicable.

Moreover, secure communication requires strong authentication of a communication peer. A BSN must be sure that it is communicating with an authorized healthcare monitor while the monitor must be able undoubtedly to verify that its data source is a certain patient's BSN regardless of the networking technology or a certain network address which is used to transfer the data. Therefore, a strong end-to-end trust relationship must be established and maintained between the monitor and the mobile gateway device.

The Host Identity Protocol (HIP) [12] is a secure networking protocol that enables end-host mobility and multihoming. HIP allows hosts to change their point of network attachment and to use several networking interfaces at the same time. Hence, HIP allows connecting mobile BSNs to a stationary healthcare monitor over various networking technologies while keeping the same identifier. Moreover, HIP allows authenticating hosts with strong cryptography. Thus, it allows the gateway device to verify the identity of a healthcare monitor and vice versa. However, HIP has not been designed for the use in medical environments and, thus, does not comply with the rigid requirements for medical devices such as timeliness and reliability. Although additional research must be conducted in order to allow the use of HIP in a medical environment, HIP is a compelling platform for connecting mobile BSNs.

5. PRIVACY PROTECTION

In the second phase of medical BSN deployment, more and more BSNs will operate off-site. Therefore, identity management and privacy protection will become a serious problem. CASPIAN, Consumers Against Supermarket Privacy Invasion and Numbering, is an example of an initiative restraining the deployment of the ubiquitous technologies due to privacy concerns [18]. CASPIAN is a real proof of the necessity of solving privacy concerns in order ubiquitous technologies, such as, e.g., pervasive healthcare, to be deployed.

Privacy protection in pervasive healthcare is relatively new, being only addressed in the context of data protection within the hospital framework [19]. However, the vision of pervasive healthcare is more ambitious as it enables ubiquitous monitoring, making inter-BSN communications especially worrying from the privacy point of view, as for instance, patient's identity or context information might be exposed. This fact leads to new privacy threats. For instance, a medical sensor node using a unique identifier can allow attackers to track people, in the same manner an RFID tag does. Another example emerges when access control issues are not considered leading to uncontrolled misuses of patients' digital identity. For instance, health insurance companies might be interested in knowing the patient health state before taking out an insurance policy.

Key exchange and authentication protocols like SIGMA [14] allow for privacy protection in the Internet. However, these protocols make use of computationally expensive public-key cryptography that is not suitable for resource constrained sensor nodes. P3P and PawS [15] aim at protecting the users' privacy in the Internet and ubiquitous computing environments respectively. Although, these protocols do not completely fit the requirements of pervasive healthcare, some guiding principles, such as, e.g., notice and disclosure, choice and consent, anonymity and pseudonymity, and proximity and locality, can be reused to guarantee an adequate privacy level in pervasive healthcare.

In this context, privacy protection must be addressed in a different manner depending on the kind of communication. On the one hand, intra-BSN communication must ensure that information and identifiers are not recognizable, so that patients cannot be tracked. For instance, by forcing the healthcare monitor to authenticate before the BSN reveals its identity allows protecting the BSNs identity from being exposed to an unauthorized party. On the other hand, inter-BSN communications must ensure that patient's identity and related medical information is only transmitted to authorized parties according to legal laws. Another peculiarity of BSNs in the context of pervasive healthcare is that the BSNs need identity control while the healthcare monitors do not require this protection.

Typically healthcare monitors are not as resource constrained as mobile sensor nodes. Therefore, it is possible to compute public key algorithms on them. This imbalance in processing power nicely matches the imbalance of computational cost of a public key signature and encryption algorithm. RSA signatures are typically costly to generate but cheap to verify and RSA decryption requires much more computation than RSA encryption. Watro et al. [16] proposed a system that exploits this imbalance to negotiate shared keys but this imbalance can also be used to authenticate the healthcare monitor and to encrypt the BSNs identity. Elliptic curve cryptography also offers promising possibilities to reduce the communicational and computational overheads of public-key cryptography.

6. CONCLUSION: OPEN ISSUES IN KEY MANAGEMENT

The provision of basic security services in pervasive healthcare requires the definition of an appropriate key management architecture. However, the definition of a suitable key management architecture is only feasible by overcoming previous problems. To this end, the following specific open issues must be solved:

On the one hand, further research into key distribution is required in order to enable efficient and secure key distribution in pervasive BSNs. Firstly, KPSs with a higher resiliency level are required in order to improve the security of current KPSs. Secondly, scalability of KPSs must be taken into consideration in order to make feasible the secure deployment of highly scalable pervasive healthcare systems. Thirdly, new KPSs must enable easy implementation of security services such as authentication and access control. Finally, developed KPSs must ensure security interoperability in highly mobile healthcare systems, and in short, make key distribution feasible in pervasive healthcare systems.

In the context of secure group management, a fully distributed security solution for BSNs is necessary, so that BSNs are able to maintain its own security relationships in a distributed manner. To this end, approaches to securely form a BSN, understood as an independent SD, must be developed. For instance, each person's BSN will be transformed into an independent SD that is able to manage the admittance and leaving of nodes, the distribution of secrets within the SD, as well as the detection and revocation of malfunctioning or intruder devices.

Remote monitoring requires utilizing other network technologies besides the sensor-to-sensor communication to bridge the physical distance between the BSN and the monitoring devices. These technologies must be combined and adapted to meet the strong security and reliability requirements of medical applications. Solutions must be able to seamlessly combine several access technologies in order to allow continuous secure monitoring in situations that cannot be covered with a single access technology.

Finally, further research is compulsory in order to address privacy concerns that emerge in different healthcare application scenarios. Firstly, new identification models are required for sensor nodes, person's BSNs and doctors in order to avoid patient tracking and protect patient and doctor digital identities. Secondly, authorization and access control systems must be designed to guarantee that patient's vital signs and context cannot be accessed by unauthorized personal.

7. REFERENCES

[1] Continua Health Alliance, Website, May 20th 2007. URL: - http://www.continuaalliance.org/home

[2] Health Insurance Portability and Accountability Act, Website, May 20th 2007, URL: http://www.hipaa.org/

[3] European Directive 95/46, Website, May 20th 2007. URL: http://www.cdt.org/privacy/eudirective/EU_Directive_.html

[4] Datasheet of the MICAz mote MPR2400CA – *Document part number: 6020-0060-04 Rev A. Available online. URL:* http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[5] Shnayder, V. and Chen, B. and Lorincz, K. and Gulford-Jones, T.R.F. and Welsh, M.: Sensor Networks for Medical Care, *Harvard University Technical Report* TR-08-05, April 2005.

[6] Gupta, V. and Wurm, M. and Zhu, Y. and Millard, M. and Fung, S. and Gura, N. and Eberle, H. and Chang Shantz, S.: Sizzle: A Standars-based end-to-end Security Architecture for the Embedded Internet. In *PERCOM'05: Procedings of the third IEEE International Conference on Pervasive Computing and Communications*, 2005.

[7] Perrig, A. and Szewczyk, R. and Tygar, JD and Wen, V. and Culler, D.E.: SPINS: Security suite for Sensor Networks. In

Proc. of the Seventh Annual International Conference on Mobie Compution and Networking (MOBICOM-01) New York, July 2001

[8] Dini, G. and Savino, M.: S2RP: a Secure and Scalable Rekeying Protocol for Wireless Sensor Networks. In *Proc. of Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference*, 2006.

[9] Çamtepe, S. A. and Yener, B.: Key Distribution Mechanisms for Wireless Sensor Networks: a Survey, *Rensselaer Polytechnic Institute* TR-05-07, March 2005.

[10] Menezes, A. J. and van Oorschot, P.C and Vanstone, S.A.: *Handbook of Applied Cryptography*. CRC Press 1996.

[11] Andersen, J. and Bardram, J. E.: BLIG: A New Approach for Sensor Identification, Grouping and Authorization in Body Sensor Networks. *In Proc of 4th International. Workshop on Wearable and Implantable Body Sensor Networks (BSN 2007)*, March 2007.

[12] Moskowitz, R. and Nikander P.: Host Identity Protocol (HIP) Architecture. *RFC4423*, May 2006

[13] Garcia, O and Baldus, H. and Sanchez, D.: Resource-Efficient Security for Medical Body Sensor Networks. *In Proc. of 3th International. Workshop on Wearable and Implantable Body Sensor Networks (BSN 2006)*. 2006.

[14] Krawczyk, H.: SIGMA: The "SIGn-and-MAc" approach to authenticated Diffie-Hellman and its use in the IKE protocols. *Advances in Cryptology—CRYPTO, Vol. 2729,* pp. 400-425, 2003

[15] Langheinrich, M.: Personal Privacy in Ubiquitous Computing, *Ph.D. Thesis*, ETH Zürich, CH, 2005

[16] Watro, R. and Kong, D. and Cuti, S. and Gardiner, C. and Lynn, C. and Kruus, P.: Securing Sensor Networks with Public Key Technology. *In Proc. of 2nd ACM Workshop on Security of Ad hoc and Sensor Networks*, Washingon, 2004

[17] Chan, H. and Gligor, V.D. and Perrig, A. and Muralidharan, G.: In *proc of IEEE Transactions on Dependable and Secure Computing* Vol. 2 No.3, July-September 2005.

[18] CASPIAN - Consumers Against Supermarket Privacy Invasion and Numbering, Website, May 20th. 2007. URL: http://www.nocards.org/

[19] Tentori, M. and Favela, F and Rodriguez, M. D.: Privacy-Aware Autonomous Agents for Pervasive Healthcare. *IEEE Intelligent Systems*, vol. 21, no. 6, pp. 55-62, Nov/Dec, 2006

Reliable Data Transport in Wireless Sensor Networks

[Extended Abstract]

Mesut Günes, Christoph Bürger, Martin Wenig, Ulrich Meis
Department of Computer Science, Informatik 4
RWTH Aachen University, Germany
{guenes, buerger, wenig, meis}@cs.rwth-aachen.de

Keywords
Transport protocols, Wireless Sensor Networks

1. MOTIVATION

In the future we expect to live in a riched internetworked environment which will be created by the integration of various wired and wireless networks. To realize such an internetworked environment different wireless networks will be used to accomplish particular tasks, e.g. *mobile ad-hoc networks* (MANET), *wireless mesh networks* (WMN), and *wireless sensor networks* (WSN). WMNs will provide network connections to large areas, MANETs may provide access to the Internet and WMN for particular situations, and WSN will be deployed for collecting data from the environment.

Wireless sensor networks [4] consist of a large number of sensor nodes. These measure data and send it over wireless links to a sink which may be several hops away. They can be deployed in a random way and organize themselves. The sink processes the received data and reacts on it if necessary. Application areas for wireless sensor networks are environmental monitoring, military applications, healthcare, and medicine.

To transport the sensed data from the sensor nodes to the sink is the responsibility of transport protocols. Although, wireless sensor networks does not exhibit the layered architecture of ISO/OSI and TCP/IP models, it is common to use the same terminology. In fact, wireless sensor networks have a cross-layer architecture. In wireless sensor networks it is often assumed that the source node is one of the plenty existing sensor nodes and the destination node is the sink. The reason for this assumption lies in the consideration, that WSNs will be established by application specific sensor nodes which are distributed once, operate until the energy is decayed totally and the sensor node will not be collected anymore.

However, there are also applications which do not need as many sensor nodes as in the aforementioned application scenarios and the sensor nodes can be accessed without any problems. Additionally, it is desired that the application of the sensor nodes can be modified. In that case, the sensor nodes need to be re-configurable or re-programmable. For such sensor networks, support from the transport protocol is required. It has to work in both ways from the sensor node to the sink as well as from the sink to the sensor node.

The remainder of the paper is as follows. Characteristics of transport protocols and existing transport protocols for wireless sensor networks are discussed in section 2. Subsequently the *Two-way Reliable Transport Protocol* is presented in section 3. The paper closes with some remarks and conclusions in section 4.

2. RELATED WORK

2.1 Characteristics of Transport Protocols

In this section we discuss some characteristics which are expected from transport protocols for wireless sensor networks. These characteristics can also be understood as "requirements" [10], however it is typically not feasible to support all of them in one protocol.

The data streams in a wireless sensor network can be classified by different aspects. The first aspect refers to whether the data stream is *continuous* or *event* based. Another aspect refers to the direction of the data stream namely from the sink to the sensor nodes (downstream) or vice versa from the sensor nodes to the sink (upstream). Continuous data streams are characterized by transmitting data periodically from the source to the destination. Event based data streams transmit only data when a defined event occurs. There are typically continuous and event based data streams from the sensor nodes to the sink and only event based data streams from the sink to the sensor nodes, e.g., management of data streams and reprogramming of the nodes. The support of both continuous and event based data streams is important for many applications, since the nature of the sensed information may command the type of the generated data stream.

Another aspect in classifying transport protocols is the reliability. Reliability on the transport protocol layer refers to the transmission of packets from the source to the destination. Either the reliability is guaranteed, i.e., 100 % or it is stochastic. In wireless sensor networks there is no need for 100 % reliability and very often it is enough to reach only p % reliability. Reliability is typically defined as the ratio of received packets and sent packets. To achieve reliability at least two problems have to be overcome by the WSN namely packet loss due to the wireless channel and due to congestion. The sensor nodes will be densely distributed. However, they have to transmit the data to the sink. This creates a funnel shape towards the sink. Therefore, congestion will typically occur in the vicinity of the sink. To prevent packet loss due to congestion a special arrangement has to be made.

The transport protocol should also support dynamics in wireless sensor networks. We refer to application dynamics of the wireless sensor network, i.e., the kind of communication should be reprogrammable when it is needed. This requires that the application and data communication is not fixed in the sensor nodes and can be changed later.

2.2 Existing Transport Protocols

The transport protocols used in the Internet, TCP [2] and UDP [1], are not suitable for wireless sensor networks due to several reasons. First of all, they incorporate too much overhead, the kind of connection establishment is not needed, the performance of TCP in wireless networks is poor, the reliability provided is not necessary [4, 10].

The focus of *Pump Slowly Fetch Quickly* (PSFQ) [9] is the reprogramming of one or more sensor nodes. This task requires the reliable transmission of all packets belonging to the reprogramming code from the sink to the sensor nodes. The protocol consists of three operations. In the *pump* operation the packets of the reprogramming code are transmitted in intervals, which are also cached in the relaying sensor nodes. The *fetch* operation deals with lost packets which can be restored from the caching nodes. With the last operation, *report*, the sink collects status information from the sensor nodes.

The goal of the *Reliable Multi-Segment Transport* protocol (RMST) [8] is the transport of data from the sensor nodes to the sink and it works on top of *Directed Diffusion* [5]. When a packet does not arrive at the sink, which is recognized by timer expiration, the packet is requested by a NACK. It may happen that the requested packet is in the cache of intermediate sensor nodes which can send the packet directly to the sink. The protocol is suitable for continuous data streams, but not for event based streams.

The goal of the *Event-to-Sink Reliable Transport* (ESRT) [7, 3] protocol is to provide a reliable transport of event based data from the sensor nodes to the sink. The occurrence of an event is accepted by the sink if it is reported by a specified number of sensor nodes. For this each sensor node reports with the frequency f, which is adapted by the sink. ESRT supports also congestion detection based on the queue level of the sensor nodes. When a sensor node recognizes congestion it informs the sink by setting the CN-bit of relayed packets. Subsequently, the sink reduces the report frequency of the sensor nodes.

The *Sensor TCP* (STCP) [6] protocol supports continuous as well as event based data streams from the sensor nodes to the sink. The sink initiates the transmission of data streams with a *Session Initiation Packet*. The reliability and transfer rate of continuous streams are tracked by the sink and requested with a NACK if a packet does not arrive. The reliability of event based packets are tracked by the source sensor node and resent if required.

3. TWO-WAY RELIABLE TRANSPORT PROTOCOL FOR WSN

In this section we present the *Two-way Reliable Transport Protocol* (TRTP) which is designed for reliable communication for both upstream and downstream traffic. It supports continuous and event based data streams, session management, and congestion control.

All the presented transport protocols were designed for a particular application. They either support upstream traffic or downstream traffic. In contrast to typical application scenarios with potentially millions of sensor nodes, we consider application scenarios with a moderate number of sensor nodes which are also accessable. Thus, energy considerations are not a main issue in these applications. Applications in the area of healthcare, medicine, and building management are in our focus. In these applications there is a demand for dynamic configuration of available data streams and reprogramming of sensor nodes.

3.1 Design Criteria for TRTP

Data streams

TRTP is designed to support continuous as well as event based data streams. The affiliation of a packet is indicated by a particular bit in the packet header. The packets of event based data streams have a higher priority than the packets of continuous data streams. Therefore, when a sensor node receives an event packet, it schedules this packet before the continuous ones. Furthermore, each data stream has a priority which is assigned by the sink, i.e., the priority of a data stream is set by the user on the application layer which is propagated by the sink when it requests data streams from sensor nodes. Packets of the same kind (continuous, event based) are handled according to their priority which is also indicated in the packet header.

The recognition of packet loss is done by the sink and the sensor nodes and is performed by exchanging ACKs and NACKs. Packet loss in continuous data streams is recognized by the sink due to timer expiration. The sink requests the packet by sending a NACK to the sensor node. In the case of event based data streams the sensor node recognizes a packet loss if it does not receive an ACK from the sink and resends the packet.

Reliability

In the context of TRTP reliability is defined by the user. The reliability of continuous data streams is tracked by the sink. When the reliability of a continuous data stream falls below the specified reliability, the sink requests packets from the source node to increase it. Reliability is interdependent with packet loss. When packet loss occurs, the sink only requests packets if the reliability falls below the specified value. In the case of event based data streams the reliability is tracked by the source sensor node. When the sensor node sends an event based packet it calculates the reliability when this packet is not received by the sink. In that case this packet is stored to be able to resend it later.

Dynamic configuration

TRTP supports dynamic configuration with a session management concept which is realized by *Session Management*

	PSFQ	ESRT	RMST	STCP	TRTP
Direction	DS	US	US	US	DS, US
Loss Detection	NACK		NACK	ACK/NACK	ACK/NACK
Loss Recovery	HbH		HbH, E2E	E2E	E2E
Reliability	Packet	Event	Packet	Packet, Event	Packet, Event
Congestion Control		X		X	X

DS = Downstream, US = Upstream, HbH = Hop-by-hop, E2E = End-to-End

Figure 1: Characteristics of transport protocols

Packets (SMP). The sink can request, delete, and modify data streams from the sensor nodes. Each SMP can concern many data streams.

Congestion Control

Congestion control in TRTP is done in three phases. i) Congestion Detection (CD), ii) Congestion Notification (CN), and iii) Rate Adjustment. Congestion detection is based on the queue level of the sensor nodes. When the queue level reaches a threshold q_c the sensor node assumes a congestion. It informs the sink by setting the CN bit of relayed packets and the source sensor node by a management packet. In this way both the sink and the source sensor nodes are informed and can react. The source sensor node decreases the transfer rate of the data stream and the sink adapts the transfer rate also. This is done only once. Further, changes in transfer rate have to be performed over the session management and with feedback from the application layer.

4. CONCLUSIONS

In this paper we introduced a new transport protocol for wireless sensor networks which provides reliable communication for both upstream and downstream traffic. It supports continuous and event based data streams and has a congestion control concept. Furthermore, the configuration of the wireless sensor network can be changed dynamically, i.e., the number and kind of data streams can be modified, new data streams can be requested and existing data streams can be deleted.

This functionality is required for applications where the sensor network is deployed manually or semiautomatically and is accessable. In these scenarios energy considerations are not the main issue. Figure 1 depicts the characteristics of the discussed transport protocols and TRTP. The most similar transport protocol in functionality to TRTP is STCP. However, STCP considers only upstream traffic and does not support the dynamic configuration of data streams.

5. REFERENCES

[1] RFC 768: User Datagram Protocol, August 1980.

[2] RFC 793: Transmission Control Protocol, September 1981.

[3] O. B. Akan and I. F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. *IEEE/ACM Transactions on Networking*, 13(5):1003–1016, October 2005.

[4] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38:393–422, 2002.

[5] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks. In *MobiCom '00: Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 56–67, New York, NY, USA, 2000. ACM Press.

[6] Y. G. Iyer, S. Gandhalm, and S. Venkatesan. STCP: A Generic Transport Layer Protocol for Wireless Sensor Networks. In *Proceedings of the 14th International Conference on Computer Communications and Networks (ICCCN 2005)*, pages 449–454, 2005.

[7] Y. Sankarasubramaniam, O. B. Akan, and I. F. Akyildiz. ESRT: Event-to-Sink Reliable Transport in Wireless Sensor Networks. In *MobiHoc '03: Proceedings of the 4th ACM International Symposium on Mobile ad hoc Networking & Computing*, pages 177–188, New York, NY, USA, 2003. ACM Press.

[8] F. Stann and J. Heidemann. RMST: Reliable Data Transport in Sensor Networks. In *Proceedings of the First International Workshop on Sensor Net Protocols and Applications* , pages 102–112, Anchorage, Alaska, USA, April 2003. IEEE.

[9] C.-Y. Wan, A. T. Campbell, and L. Krishnamurthy. PSFQ: A Reliable Transport Protocol for Sensor Networks. *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, 23:862 – 872, 2005.

[10] C. Wang, K. Sohraby, B. Li, and M. Y. H. Daneshmand. A Survey of Transport Protocols for Wireless Sensor Networks. *Network, IEEE*, 20:34–40, May-June 2006.

A 6lowpan Implementation for TinyOS 2.0

Matúš Harvan
Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen, Germany
m.harvan@jacobs-university.de

Jürgen Schönwälder
Computer Science
Jacobs University Bremen
Campus Ring 1
28759 Bremen, Germany
j.schoenwaelder@jacobs-university.de

1. INTRODUCTION

Traditionally, wireless sensor networks have used custom, light-weight network protocols such as Active Messages. However, given the common presence of an 802.15.4 radio interface [4] on the motes and the 6lowpan adaptation layer [6] allowing the exchange of IPv6 packets over 802.15.4 links, enabling IPv6 connectivity in wireless sensor networks and connecting them to the global Internet becomes feasible. By natively supporting the IPv6 protocol, these devices would become first-class Internet citizens capable of communication with any other IPv6-enabled host and benefit from the standardized and already estabilished technology and a plethora of readily available applications. To this end a 6lowpan/IPv6 stack [3] has been implemented for TinyOS 2.0 [2], an embedded operating system commonly used in wireless sensor networks.

The rest of this document is structured as follows. The 6lowpan adaptation layer is briefly introduced in Section 2, the implementation is discussed in Section 3, related work is described in Section 4 and the document concludes in Section 5.

2. 6LOWPAN

The 6lowpan adaptation layer allows to transport IPv6 packets over 802.15.4 links. To meet the IPv6-required MTU of at least 1280 bytes with the 802.15.4 layer offering at most 102 bytes of payload per frame, a fragmentation mechanism below the IP layer is specified using an optional Fragmentation Header before the actual IPv6 header. Support for mesh networking is provided by the optional Mesh Addressing and Broadcast Headers. The former carries the Originator and Final Destination link-layer addresses while the latter contains a sequence number used to detect duplicated frames. Both are carried at the beginning of the 802.15.4 payload. Furthermore, mechanisms for compressing the IPv6 header from 40 down to 2 bytes and the UDP header from 8 down to 4 bytes, in the ideal case, are specified. To distinguish between a compressed and uncompressed header, a 1-byte dispatch value is prepended before the header. The optional 6lowpan headers mentioned earlier also start with a dispatch value allowing the recipient to determine what types of headers are present. The format of the 6lowpan adaptation layer is specified in [6].

3. IMPLEMENTATION

The goal for the implementation was to support replying to an ICMP echo request message (ping) and exchanging of UDP datagrams. Only the bare minimum necessary for meeting that goal was implemented.

The main restriction of the implementation was the amount of RAM available on the target platform, i.e., 4 KB on the MicaZ. Although aiming for an embedded implementation, easily readable and maintainable code was preferred over optimizing to squeeze into the least possible amount of memory at the cost of hard to understand programming constructs, hacks or munging of code into a few large functions for saving space on the stack. Each network layer and protocol are handled by a separate function. This allows to easily add more functionality in the future.

3.1 6lowpan for Linux

Most PCs today do not have an 802.15.4 interface and common operating systems such as Linux or the BSDs do not include a 6lowpan implementation. To allow for exchanging packets between the motes and a Linux PC, a tunneling daemon has been developed to use a mote as an 802.15.4 interface. The scenario is shown in Figure 1.

The mote runs the TinyOS sample application *BaseStationCC2420* forwarding traffic between the 802.15.4 and the USB interface of a mote. This mote is connected to the PC via the USB interface.

The translating daemon on the PC is a C program exchanging packets between the USB interface and a tun interface. The latter is a virtual network interface allowing a user space process to read and write packets to it. The daemon decapsulates the 6lowpan-encapsulated IPv6 packets received from the mote and 6lowpan-encapsulates the plain IPv6 packets received on the tun interface. This allows to use standard IPv6 applications on Linux for communication with the motes without modifying the Linux kernel. Furthermore, by enabling IPv6 forwarding on the PC, the motes can be connected to the global Internet.

3.2 Evaluation

The implementation was tested using a scenario as shown in Figure 1. A TelosB mote with the BaseStationCC2420 application was connected to a Linux PC running the translating daemon. Two other TelosB motes and a MicaZ mote were flashed with the 6lowpan implementation. The motes were

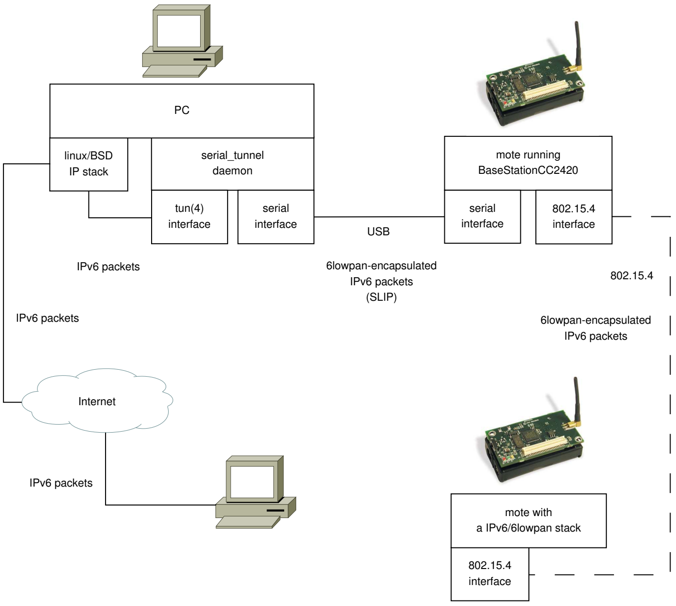


Figure 1: The motes and the Internet

successfully replying to ICMP echo requests initiated from the PC as well as exchanging UDP datagrams. Both short unfragmented packets as well as large, fragmented packets of size up to 1280 bytes were successfully exchanged.

The main limitation to interoperability with other 6lowpan implementations is the absence of a proper 802.15.4 stack in TinyOS 2.0. Although the implementation supports the ICMP echo mechanism and the UDP protocol, many features required for IPv6 implementations are missing. Among others, the Neighbor Discovery has not been implemented and packets are broadcasted on the link-layer, IPv6 extensions headers are not processed, IPv6 fragmentation is not supported and ICMP error messages are not generated. While many of these could be added, it is unclear whether they make sense in an embedded system. For example, is one willing to trade decreased battery life for regular neighbor advertisements or neighbor unreachability detection? Or if an error is encountered while processing a received packet, should a 1280 bytes long ICMP error message be sent back? Should one be sent back at all?

4. RELATED WORK

Several 6lowpan implementations for wireless sensor networks have been announced while this project was in progress.

The *Arch Rock* company has announced a commercial 6lowpan implementation *Primer Pack/IP* in March 2007. As this is a commercial implementation, technical information is scarce.

The *Sensinode* company has released a GPL-licensed 6lowpan implementation called *NanoStack v0.9.4* in April 2007. It is claimed to be up to date with version 12 of the 6lowpan format draft and to include IEEE 802.15.4 Beacon-mode support. The source code, however, does not seem to include 6lowpan fragmentation support and UDP checksumming.

uIP[1] is a TCP/IPv4 stack written by Adam Dunkels. It runs on 8-bit controllers with a few hundred bytes of RAM. It has been ported to TinyOS 1.1 by Andrew Christian from the Hewlett-Packard Company. It is available in the `tinyos-1.x/contrib/handhelds/tos/lib/UIP/` directory of the TinyOS 1.1 distribution.

While an IP stack can be implemented on the motes, it is also possible to use a proxy-based scheme. In this case a special proxy server is employed as a gateway separating the sensor network and the IP network. This allows to freely choose the communication protocol used within the sensor network. Although limited to IPv4, the Sensor Internet Protocol (SIP) [5] is an example of such a proxy scheme.

5. CONCLUSION

A 6lowpan/IPv6 stack has been implemented for the TinyOS 2.0 operating system and was tested on the TelosB and MicaZ hardware platforms. Using the translating daemon on the PC and a mote as the base station, it is possible to exchange IPv6 packets between the motes and a PC without an 802.15.4 interface. In case IP forwarding is set up on the PC and a properly assigned and routable global IPv6 prefix is used, the motes can be connected to the global Internet. More information about the implementation can be found in [3] and it can be downloaded from [7]. Furthermore, discussions have started to include it in the TinyOS distribution.

As future work it would be useful to add a proper 802.15.4 stack. Instead of broadcasting on the link-layer, Neighbor Discovery could be implemented. Various mesh routing algorithms could be investigated and the Mesh Addressing and Broadcast Headers could be used for mesh networking. Finally, SNMP could be implemented on top of the 6lowpan stack for collecting sensor values.

6. REFERENCES

[1] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[2] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler. The nesC Language: A Holistic Approach to Networked Embedded Systems. In *PLDI03*. ACM, June 2003.

[3] M. Harvan. Connecting Wireless Sensor Networks to the Internet - a 6lowpan Implementation for TinyOS 2.0. Master's thesis, School of Engineering and Science, Jacobs University Bremen, May 2007.

[4] IEEE. IEEE Std. 802.15.4-2003, Oct. 2003.

[5] X. Luo, K. Zheng, Y. Pan, and Z. Wu. A TCP/IP implementation for wireless sensor networks. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2004.

[6] G. Montenegro, N. Kushalnagar, D. E. Culler, and J. W. Hui. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. Internet-Draft Version 13, IETF, April 2007. Work in progress.

[7] `http://www.eecs.iu-bremen.de/users/harvan/files/6lowpan.tar.gz`.

Aachener Informatik-Berichte

This list contains all technical reports published during the past five years. A complete list of reports dating back to 1987 is available from http://aib.informatik.rwth-aachen.de/. To obtain copies consult the above URL or send your request to: Informatik-Bibliothek, RWTH Aachen, Ahornstr. 55, 52056 Aachen, Email: `biblio@informatik.rwth-aachen.de`

2001-01 * Jahresbericht 2000
2001-02 Benedikt Bollig, Martin Leucker: Deciding LTL over Mazurkiewicz Traces
2001-03 Thierry Cachat: The power of one-letter rational languages
2001-04 Benedikt Bollig, Martin Leucker, Michael Weber: Local Parallel Model Checking for the Alternation Free mu-Calculus
2001-05 Benedikt Bollig, Martin Leucker, Thomas Noll: Regular MSC Languages
2001-06 Achim Blumensath: Prefix-Recognisable Graphs and Monadic Second-Order Logic
2001-07 Martin Grohe, Stefan Wöhrle: An Existential Locality Theorem
2001-08 Mareike Schoop, James Taylor (eds.): Proceedings of the Sixth International Workshop on the Language-Action Perspective on Communication Modelling
2001-09 Thomas Arts, Jürgen Giesl: A collection of examples for termination of term rewriting using dependency pairs
2001-10 Achim Blumensath: Axiomatising Tree-interpretable Structures
2001-11 Klaus Indermark, Thomas Noll (eds.): Kolloquium Programmiersprachen und Grundlagen der Programmierung
2002-01 * Jahresbericht 2001
2002-02 Jürgen Giesl, Aart Middeldorp: Transformation Techniques for Context-Sensitive Rewrite Systems
2002-03 Benedikt Bollig, Martin Leucker, Thomas Noll: Generalised Regular MSC Languages
2002-04 Jürgen Giesl, Aart Middeldorp: Innermost Termination of Context-Sensitive Rewriting
2002-05 Horst Lichter, Thomas von der Maßen, Thomas Weiler: Modelling Requirements and Architectures for Software Product Lines
2002-06 Henry N. Adorna: 3-Party Message Complexity is Better than 2-Party Ones for Proving Lower Bounds on the Size of Minimal Nondeterministic Finite Automata
2002-07 Jörg Dahmen: Invariant Image Object Recognition using Gaussian Mixture Densities
2002-08 Markus Mohnen: An Open Framework for Data-Flow Analysis in Java
2002-09 Markus Mohnen: Interfaces with Default Implementations in Java
2002-10 Martin Leucker: Logics for Mazurkiewicz traces
2002-11 Jürgen Giesl, Hans Zantema: Liveness in Rewriting
2003-01 * Jahresbericht 2002
2003-02 Jürgen Giesl, René Thiemann: Size-Change Termination for Term Rewriting
2003-03 Jürgen Giesl, Deepak Kapur: Deciding Inductive Validity of Equations

2003-04 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Improving Dependency Pairs

2003-05 Christof Löding, Philipp Rohde: Solving the Sabotage Game is PSPACE-hard

2003-06 Franz Josef Och: Statistical Machine Translation: From Single-Word Models to Alignment Templates

2003-07 Horst Lichter, Thomas von der Maßen, Alexander Nyßen, Thomas Weiler: Vergleich von Ansätzen zur Feature Modellierung bei der Softwareproduktlinienentwicklung

2003-08 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, Stephan Falke: Mechanizing Dependency Pairs

2004-01 * Fachgruppe Informatik: Jahresbericht 2003

2004-02 Benedikt Bollig, Martin Leucker: Message-Passing Automata are expressively equivalent to EMSO logic

2004-03 Delia Kesner, Femke van Raamsdonk, Joe Wells (eds.): HOR 2004 – 2nd International Workshop on Higher-Order Rewriting

2004-04 Slim Abdennadher, Christophe Ringeissen (eds.): RULE 04 – Fifth International Workshop on Rule-Based Programming

2004-05 Herbert Kuchen (ed.): WFLP 04 – 13th International Workshop on Functional and (Constraint) Logic Programming

2004-06 Sergio Antoy, Yoshihito Toyama (eds.): WRS 04 – 4th International Workshop on Reduction Strategies in Rewriting and Programming

2004-07 Michael Codish, Aart Middeldorp (eds.): WST 04 – 7th International Workshop on Termination

2004-08 Klaus Indermark, Thomas Noll: Algebraic Correctness Proofs for Compiling Recursive Function Definitions with Strictness Information

2004-09 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Parameterized Power Domination Complexity

2004-10 Zinaida Benenson, Felix C. Gärtner, Dogan Kesdogan: Secure Multi-Party Computation with Security Modules

2005-01 * Fachgruppe Informatik: Jahresbericht 2004

2005-02 Maximillian Dornseif, Felix C. Gärtner, Thorsten Holz, Martin Mink: An Offensive Approach to Teaching Information Security: "Aachen Summer School Applied IT Security"

2005-03 Jürgen Giesl, René Thiemann, Peter Schneider-Kamp: Proving and Disproving Termination of Higher-Order Functions

2005-04 Daniel Mölle, Stefan Richter, Peter Rossmanith: A Faster Algorithm for the Steiner Tree Problem

2005-05 Fabien Pouget, Thorsten Holz: A Pointillist Approach for Comparing Honeypots

2005-06 Simon Fischer, Berthold Vöcking: Adaptive Routing with Stale Information

2005-07 Felix C. Freiling, Thorsten Holz, Georg Wicherski: Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks

2005-08 Joachim Kneis, Peter Rossmanith: A New Satisfiability Algorithm With Applications To Max-Cut

2005-09 Klaus Kursawe, Felix C. Freiling: Byzantine Fault Tolerance on General Hybrid Adversary Structures

2005-10 Benedikt Bollig: Automata and Logics for Message Sequence Charts

2005-11 Simon Fischer, Berthold Vöcking: A Counterexample to the Fully Mixed Nash Equilibrium Conjecture

2005-12 Neeraj Mittal, Felix Freiling, S. Venkatesan, Lucia Draque Penso: Efficient Reductions for Wait-Free Termination Detection in Faulty Distributed Systems

2005-13 Carole Delporte-Gallet, Hugues Fauconnier, Felix C. Freiling: Revisiting Failure Detection and Consensus in Omission Failure Environments

2005-14 Felix C. Freiling, Sukumar Ghosh: Code Stabilization

2005-15 Uwe Naumann: The Complexity of Derivative Computation

2005-16 Uwe Naumann: Syntax-Directed Derivative Code (Part I: Tangent-Linear Code)

2005-17 Uwe Naumann: Syntax-directed Derivative Code (Part II: Intraprocedural Adjoint Code)

2005-18 Thomas von der Maßen, Klaus Müller, John MacGregor, Eva Geisberger, Jörg Dörr, Frank Houdek, Harbhajan Singh, Holger Wußmann, Hans-Veit Bacher, Barbara Paech: Einsatz von Features im Software-Entwicklungsprozess - Abschlußbericht des GI-Arbeitskreises “Features”

2005-19 Uwe Naumann, Andre Vehreschild: Tangent-Linear Code by Augmented LL-Parsers

2005-20 Felix C. Freiling, Martin Mink: Bericht über den Workshop zur Ausbildung im Bereich IT-Sicherheit Hochschulausbildung, berufliche Weiterbildung, Zertifizierung von Ausbildungsangeboten am 11. und 12. August 2005 in Köln organisiert von RWTH Aachen in Kooperation mit BITKOM, BSI, DLR und Gesellschaft fuer Informatik (GI) e.V.

2005-21 Thomas Noll, Stefan Rieger: Optimization of Straight-Line Code Revisited

2005-22 Felix Freiling, Maurice Herlihy, Lucia Draque Penso: Optimal Randomized Fair Exchange with Secret Shared Coins

2005-23 Heiner Ackermann, Alantha Newman, Heiko Röglin, Berthold Vöcking: Decision Making Based on Approximate and Smoothed Pareto Curves

2005-24 Alexander Becher, Zinaida Benenson, Maximillian Dornseif: Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks

2006-01 * Fachgruppe Informatik: Jahresbericht 2005

2006-02 Michael Weber: Parallel Algorithms for Verification of Large Systems

2006-03 Michael Maier, Uwe Naumann: Intraprocedural Adjoint Code Generated by the Differentiation-Enabled NAGWare Fortran Compiler

2006-04 Ebadollah Varnik, Uwe Naumann, Andrew Lyons: Toward Low Static Memory Jacobian Accumulation

2006-05 Uwe Naumann, Jean Utke, Patrick Heimbach, Chris Hill, Derya Ozyurt, Carl Wunsch, Mike Fagan, Nathan Tallent, Michelle Strout: Adjoint Code by Source Transformation with OpenAD/F

2006-06 Joachim Kneis, Daniel Mölle, Stefan Richter, Peter Rossmanith: Divide-and-Color

2006-07 Thomas Colcombet, Christof Löding: Transforming structures by set interpretations

2006-08 Uwe Naumann, Yuxiao Hu: Optimal Vertex Elimination in Single-Expression-Use Graphs

2006-09 Tingting Han, Joost-Pieter Katoen: Counterexamples in Probabilistic Model Checking

2006-10 Mesut Günes, Alexander Zimmermann, Martin Wenig, Jan Ritzerfeld, Ulrich Meis: From Simulations to Testbeds - Architecture of the Hybrid MCG-Mesh Testbed

2006-11 Bastian Schlich, Michael Rohrbach, Michael Weber, Stefan Kowalewski: Model Checking Software for Microcontrollers

2006-12 Benedikt Bollig, Joost-Pieter Katoen, Carsten Kern, Martin Leucker: Replaying Play in and Play out: Synthesis of Design Models from Scenarios by Learning

2006-13 Wong Karianto, Christof Löding: Unranked Tree Automata with Sibling Equalities and Disequalities

2006-14 Danilo Beuche, Andreas Birk, Heinrich Dreier, Andreas Fleischmann, Heidi Galle, Gerald Heller, Dirk Janzen, Isabel John, Ramin Tavakoli Kolagari, Thomas von der Maßen, Andreas Wolfram: Report of the GI Work Group “Requirements Management Tools for Product Line Engineering”

2006-15 Sebastian Ullrich, Jakob T. Valvoda, Torsten Kuhlen: Utilizing optical sensors from mice for new input devices

2006-16 Rafael Ballagas, Jan Borchers: Selexels: a Conceptual Framework for Pointing Devices with Low Expressiveness

2006-17 Eric Lee, Henning Kiel, Jan Borchers: Scrolling Through Time: Improving Interfaces for Searching and Navigating Continuous Audio Timelines

2007-01 * Fachgruppe Informatik: Jahresbericht 2006

2007-02 Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl: SAT Solving for Termination Analysis with Polynomial Interpretations

2007-03 Jürgen Giesl, René Thiemann, Stephan Swiderski, and Peter Schneider-Kamp: Proving Termination by Bounded Increase

2007-04 Jan Buchholz, Eric Lee, Jonathan Klein, and Jan Borchers: coJIVE: A System to Support Collaborative Jazz Improvisation

2007-05 Uwe Naumann: On Optimal DAG Reversal

2007-06 Joost-Pieter Katoen, Thomas Noll, and Stefan Rieger: Verifying Concurrent List-Manipulating Programs by LTL Model Checking

2007-07 Alexander Nyßen, Horst Lichter: MeDUSA - MethoD for UML2-based Design of Embedded Software Applications

2007-08 Falk Salewski and Stefan Kowalewski: Achieving Highly Reliable Embedded Software: An empirical evaluation of different approaches

2007-09 Tina Kraußer, Heiko Mantel, and Henning Sudbrock: A Probabilistic Justification of the Combining Calculus under the Uniform Scheduler Assumption

2007-12 Uwe Naumann: An L-Attributed Grammar for Adjoint Code

2007-13 Uwe Naumann, Michael Maier, Jan Riehme, and Bruce Christianson: Second-Order Adjoints by Source Code Manipulation of Numerical Programs

2007-14	Jean Utke, Uwe Naumann, Mike Fagan, Nathan Tallent, Michelle Strout, Patrick Heimbach, Chris Hill, and Carl Wunsch: OpenAD/F: A Modular, Open-Source Tool for Automatic Differentiation of Fortran Codes

* These reports are only available as a printed version.
Please contact `biblio@informatik.rwth-aachen.de` to obtain copies.