13. GI / ITG
Fachgespräch Sensornetze

Bettina Schnor
Institut für Informatik und Computational Science

Berichte Institut für Informatik und Computational Science
UP, ISSN 0946-7580, TR-2014-1

Programmkommittee

Oleksandr Artemenko

Steffen Christgau

Waltenegus Dargie

Michael Engel

Sebastian Fudickar

Jörg Hähner

Matthias Hollick

Reinhardt Karnapke

Peter Langendörfer

Stefan Lohs

Sebastian Porombka

Bettina Schnor

Andre Sieber

Inhaltsverzeichnis

Session 1: Sensor-Netze - Quo Vadis?

Wireless Sensor Networks and Their Applications: Where Do We Stand? And Where Do We Go?
Andreas Reinhardt, Sebastian Zöller und Delphine Christin 1

Development of a Contiki border router for the interconnection of 6LoWPAN and Ethernet
Viktor Eichmann und Thomas Scheffler 5

langOS - A Low Power Application-specific Configurable Operating System
Oliver Stecklina, Andreas Krumholz und Stephan Kornemann 9

Session 2: Algorithmen

Self-Stabilizing Aggregation- and Reduction-Structures for Wireless Sensor Networks
Sandra Beyer, Stefan Lohs, Reinhardt Karnapke und Jörg Nolte 13

Advanced Timestamping for pairwise Clock Drift Detection in Wireless Sensor/Actuator Networks
Marcel Baunach 17

Session 3: Zuverlässigkeit und Energiemanagement

Anwendungsmöglichkeiten softwarebasierter Selbstreparaturtechniken für Prozessoren in Sensorknoten
Mario Schölzel 21

From Energy Accounting to Energy Management
Andre Sieber, Reinhardt Karnapke und Jörg Nolte 25

Synchronisierte Messung durch Trigger-Broadcast und weitere Funktionen für drahtlose Batteriesensorik
Valentin Roscher, Matthias Schneider, Phillip Durdaut, Nico Sassano, Sergej Pereguda, Eike Mense und Karl-Ragmar Riemschneider 29

Session 4: Anwendungen

Quadrotor-based DT-WSNs for Disaster Recovery
Felix Büsching, Keno Garlichs, Johannes van Balen, Yannic Schröder, Kai Homeier, Ulf Kulau, Wolf-Bastian Pöttner, Stephan Rottmann, Sebastian Schildt, Georg von Zengen und Lars Wolf 33

Less GHz is More - On Indoor Localisation Accuracies and Device runtimess
Sebastian Fudickar 36

A Validated Simulation Model for Communicating Paragliders
Juergen Eckert, Christoph Sommer und David Eckhoff 41

Wireless Sensor Networks and Their Applications: Where Do We Stand? And Where Do We Go?

Andreas Reinhardt
School of Computer Science and Engineering
The University of New South Wales
Sydney, Australia
andreasr@cse.unsw.edu.au

Sebastian Zöller
Multimedia Communications Lab
Technische Universität Darmstadt
Darmstadt, Germany
zoeller@kom.tu-darmstadt.de

Delphine Christin
Privacy and Security in Ubiquitous Computing
University of Bonn
Bonn, Germany
christin@cs.uni-bonn.de

***Abstract*—Research on wireless sensor networks has been ongoing for more than 15 years. As a result, an enormous number of novel ideas have been proposed in academic and industrial research since then. These comprise the design of new hardware components, novel communication and processing regimes, and the realization of systems that would have been unimaginable before wireless sensor networks came into existence. The resulting application areas are broad, ranging from deployments of a few low-cost sensor nodes to the installation of large numbers of highly specialized sensing systems. In this paper, we summarize wireless sensor network application trends and point out future directions and emerging novel application domains that bear high research potential.**

I. Introduction

In 1999, the notion of *mote*s has been introduced in [1]. This visionary idea of combining sensing, computation, and communication capabilities into minuscule systems that can easily be deployed to sense environmental parameters has since been taken up by innumerable researchers around the world. Consequently, many facets of the resulting wireless sensor networks (*WSN*s) have been investigated to date, ranging from designs for hard- and software to novel application scenarios. In fact, virtually no part of motes and their applications has been left untouched by researchers in search for optimization potential, new research directions, and beyond.

Strong ongoing research activities confirm the topicality of WSN research. However, at the same time the ubiquity of research in this domain naturally elicits the question whether new research is still possible and meaningful. In this paper, we thus present our vision of future research directions in wireless sensor networks. Although a large spectrum of potential open challenges still exists, we specifically focus on application scenarios for WSN technology. Novel applications directly implicate the need for research on many underlying aspects, e.g., hardware platforms, processing algorithms and communication protocols as well as sensor data collection and interpretation.

In this paper, we first survey existing WSN applications in Sec. II, in order to delineate emerging trends in sensor networks from the state-of-the-art. Subsequently, we highlight our visions for sensor network applications in Sec. III and outline selected required research contributions. Finally, we summarize the core findings of this paper in Sec. IV.

II. Applications of Wireless Sensor Networks

In an approach to highlight the breadth of WSN application areas, we categorize the deployments presented in twelve survey publications (cf. [2–13]) in Table I. We discuss the characteristics of the nine resulting categories and summarize representative deployments for each category as follows.

A. Environmental monitoring

Environmental monitoring is one of the oldest application areas for WSN technology. WSNs provide the opportunity for the unobtrusive monitoring of areas that are difficult to access for humans, e.g., natural animal habitats. One of the earliest WSN deployments has been the deployment within the Great Duck Island project [14], where the natural habitat of Leach's Storm Petrels was monitored. Another prominent environmental monitoring deployment is the PermaSense project [15], in which WSN technology is applied to monitor a hard-to-reach permafrost area in the Swiss Alps.

B. Disaster control

The prevention of disasters and proper reactions to disasters where prevention is not possible is a second application area for WSNs. An application example is the usage of motes on chemical drums [16, 17] to monitor that a maximum quantity of chemicals allowed to be stored together in a certain area is not exceeded. Structural health monitoring, e.g., of bridges, constitutes another example for applying WSNs in the field of disaster control. It serves the purpose to estimate the current state of a structure and detect relevant state changes so that critical states can be identified and countermeasures taken in time to prevent disasters. One such WSN has, e.g., been deployed on the Golden Gate Bridge in San Francisco [18].

C. Smart spaces

Ambient intelligence, or *smart spaces*, can be realized by continually monitoring the environment and taking actuation decisions to improve the users' comfort and safety. Currently, many applications focus on the user-oriented control of heating, ventilation, and air conditioning systems [19]. Another application example for WSN technology in the context of smart space realizations is the usage of motes for monitoring electrical energy consumption [20], targeting building energy efficiency by reducing energy consumption.

D. Object tracking and monitoring

Thanks to their small size and unobtrusive wireless operation capabilities, motes can easily be attached to everyday objects. This allows these objects to be monitored, e.g., with regard to their location and environment. A prominent application area for the resulting tracking capabilities is the use of WSN technology in logistics. The enablement of tracking assets, in particular high-valued goods, in transport processes where defined transportation routes and object integrity need to be continually ensured is presented in [21, 22].

E. Human-centric WSNs

Similar in their nature to the aforementioned smart spaces, human-centric WSNs comprise unobtrusive sensors collecting a huge range of parameters about humans. The collected data are subsequently being evaluated and combined to serve humans and their wellbeing and learning. In this context, applications in the medical and healthcare domains are prominent examples for the beneficial application of WSN technology. Example applications range from monitoring and supporting hospitalized patients to enabling new possibilities for extensive medical field studies [23].

F. Traffic control

Intelligent parking management systems constitute one prominent example for the beneficial application of WSN technology in the domain of traffic control [24, 25]. In this context, WSN technology can be employed to detect and identify vehicles with the goal to monitor vehicles in a parking lot and thus being able to provide information for example on the number and location of free sparking spaces. Similar concepts can also be applied to freeways, intersections, and many other traffic entities within the scope of realizing smart cities. WSN technology can substantially support traffic surveillance systems in this context as well (cf. [26]).

G. Security

In particular in military sensing, security applications constitute another one of the oldest application domains for WSN technology. The detection of snipers on a battlefield with the help of WSN technology has, e.g., been presented in [27]. Furthermore, WSN technology can be beneficially employed in the context of surveillance systems with the goal to autonomously detect intruders, track their movements, and classify them [28].

H. Industrial process monitoring and control

WSNs can also be employed to monitor the correct execution of process steps in industrial deployments by providing the operator with means to adapt process parameters on demand [29]. Machine surveillance and maintenance is another huge application field in this context. Here, WSNs can be employed for condition-based maintenance of machines exploiting the capability of local data processing and providing monitoring data in real time in order to enhance the utilization and lifetime of the monitored equipment [30].

I. Diverse other application areas

Manifold other application domains exist that cannot be unambiguously assigned to one of the application areas. For example, enhancing the efficiency of aircrafts during their flights constitutes a general engineering task, which can as well benefit from the application of WSN technology [31].

III. Future Research Directions

After having highlighted the broad range of existing domains, the identification of novel fields appears challenging. While most of the previously introduced solutions, however, solve well-defined problems by applying WSN technology, we highlight future research directions at a more generic scale. As follows, we list selected research challenges which we expect to play a vital and integral role in future WSN deployments.

A. Enabling the Internet of Things

The emerging vision of the Internet of Things (*IoT*) entails many research challenges to ensure its success. While today's WSN deployments are commonly designed and operated by a single stakeholder and rely on hardware of one particular type only, the billions of networked devices envisioned in the IoT cannot be assumed to follow this tradition. Novel means for cross-platform address allocation, device addressing, unicast and multicast routing, energy efficiency, and interoperability between applications are essential for the successful realization of the IoT. Besides these more technical challenges, novel applications and business models are also strongly required to make the IoT a success and cater to the creation of smart buildings, smart cities, and beyond.

B. Component re-use and smart data processing

The prevalent majority of existing WSNs have been tailored to application-specific use cases. While component modularization plays a crucial role in other software engineering-related disciplines, WSN applications are still often developed from scratch. The definition of re-usable components and corresponding interfaces to simplify and streamline application development is still an open issue. This especially applies to data processing components, which are generally developed from the ground up for each new application scenario despite their potential re-usability in other areas.

C. Validation of results through practical experimentation

Newly proposed algorithms and protocols for WSNs are often only validated by means of analytical and/or simulation studies. While this allows for the simple evaluation of the devised algorithms at scale, real-world effects are implicitly not considered. The widely observed discrepancies between simulations and real-world experiments, however, strongly motivate more practical experimentation in WSN research. Due to the availability of embedded sensing system hardware in many varieties and the large number of publicly accessible testbed sites, practical research is easily possible and essential to demonstrate the viability of any newly proposed solution.

IV. Conclusions

In this paper, we have briefly summarized the broad range of current application domains for wireless sensor networks. We have compiled a short list of representative applications for each category and thus highlighted the versatility of WSN technology. From the broad range of existing applications, however, the question emerges whether further research is still necessary and worthwhile. We agree that indeed many WSN implementations have been presented in the last 15 years to solve existing real-world challenges in unprecedented novel ways. However, following our overview of the state-of-the-art, we have also identified several future research directions and methodologies that we expect to bear significant potential. Besides continuing to deploy WSNs and gain more practical experiences, most identified challenges are of a more generic nature. Once viable solutions to these challenges have been found, we strongly expect them to be enabling technologies for the widespread use of WSNs in the future. Especially as WSNs play an integral role for the emerging Internet of Things, their *raison d'être* will be given for many years to come.

References

[1] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for 'Smart Dust'," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom)*, 1999, pp. 271–278.

[2] T. Arampatzis, J. Lygeros, and S. Manesis, "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks," in *Proceedings of the 13th Mediterranean Conference on Control and Automation (MED)*, 2005, pp. 719–724.

[3] E. H. Callaway Jr., *Wireless Sensor Networks: Architectures and Protocols*. Boca Raton, FL, USA: Auerbach Publications, 2004.

[4] C.-Y. Chong and S. Kumar, "Sensor Networks: Evolution, Opportunities, and Challenges," *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1247–1256, 2003.

[5] D. Culler, D. Estrin, and M. Srivastava, "Guest Editors' Introduction: Overview of Sensor Networks," *Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[6] M. Haenggi, "Opportunities and Challenges in Wireless Sensor Networks," in *Smart Dust: Sensor Network Applications, Architecture, and Design*, I. Mahgoub and M. Ilyas, Eds. Boca Raton, FL, USA: CRC Press Taylor & Francis Group, 2006, ch. 1, pp. 1–14.

[7] H. Karl and A. Willig, *Protocols and Architectures for Wireless Sensor Networks*. Chichester, UK: John Wiley & Sons, 2007.

[8] I. Khemapech, I. Duncan, and A. Miller, "A Survey of Wireless Sensor Networks Technology," in *Proceedings of the 6th Annual Postgraduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNET)*, 2005, pp. 1–6.

[9] L. Oliveira and J. Rodrigues, "Wireless Sensor Networks: A Survey on Environmental Monitoring," *Journal of Communications*, vol. 6, no. 2, pp. 143–151, 2011.

[10] K. Sohraby, D. Minoli, and T. Znati, *Wireless Sensor Networks: Technology, Protocols, and Applications*. Hoboken, NJ, USA: John Wiley & Sons, 2007.

[11] R. Verdone, D. Dardari, G. Mazzini, and A. Conti, *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*. Oxford, UK: Academic Press, 2008.

[12] Q. Wang, H. Hassanein, and K. Xu, "A practical perspective on wireless sensor networks," in *Smart Dust: Sensor Network Applications, Architecture, and Design*, I. Mahgoub and M. Ilyas, Eds. Boca Raton, FL, USA: Taylor & Francis, 2006, ch. 7, pp. 1–28.

[13] F. Zhao and L. J. Guibas, *Wireless Sensor Networks: An Information Processing Approach*. Amsterdam: Morgan Kaufmann, 2004.

[14] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2002, pp. 88–97.

[15] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel, "PermaDAQ: A Scientific Instrument for Precision Sensing and Data Recovery in Environmental Extremes," in *Proceedings of the International Conference on Information Processing in Sensor Networks (IPSN)*, 2009, pp. 265–276.

[16] M. Strohbach, H.-W. Gellersen, G. Kortuem, and C. Kray, "Cooperative Artefacts: Assessing Real World Situations with Embedded Technology," in *Proceedings of the 6th International Conference on Ubiquitous Computing (UbiComp)*, 2004, pp. 250–267.

[17] U. Kubach, C. Decker, and K. Douglas, "Collaborative Control and Coordination of Hazardous Chemicals," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, p. 309.

[18] S. Kim, S. Pakzad, D. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, "Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks," in *Proceedings of the International Symposium on Information Processing in Sensor Networks (IPSN)*, 2007, pp. 254–263.

[19] E. Arens, C. Federspiel, D. Wang, and C. Huizenga, "How Ambient Intelligence will Improve Habitability and Energy Efficiency in Buildings," in *Ambient Intelligence*, W. Weber, J. M. Rabaey, and E. Aarts, Eds. Berlin, Heidelberg: Springer, 2005, pp. 63–80.

[20] C. Kappler and G. Riegel, "A Real-World, Simple Wireless Sensor Network for Monitoring Electrical Energy Consumption," in *Wireless Sensor Networks*, ser. Lecture Notes in Computer Science, H. Karl, A. Wolisz, and A. Willig, Eds. Berlin, Heidelberg: Springer, 2004, vol. 2920, pp. 339–352.

[21] L. Ruiz-Garcia, P. Barreiro, J. Rodriguez-Bermejo, and J. I. Robla, "Monitoring the Intermodal, Refrigerated Transport of Fruit Using Sensor Networks," *Spanish Journal of Agricultural Research*, vol. 5, no. 2, pp. 142–156, 2007.

[22] F. Valente, G. Zacheo, P. Losito, and P. Camarda, "A Telecommunications Framework for Real-Time Monitoring of Dangerous Goods Transport," in *Proceedings of the 9th International Conference on Intelligent Transport Systems Telecommunications (ITST)*, 2009, pp. 13–18.

[23] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless Sensor Networks for Healthcare," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1947–1960, 2010.

[24] J. Chinrungrueng, U. Sunantachaikul, and S. Triamlumlerd, "Smart Parking: An Application of Optical Wireless Sensor Network," in *Proceedings of the International Symposium on Applications and the Internet Workshops (SAINT)*, 2007, p. 66.

[25] J. Gu, Z. Zhang, F. Yu, and Q. Liu, "Design and Implementation of a Street Parking System Using Wireless Sensor Networks," in *Proceedings of the 10th IEEE International Conference on Industrial Informatics (INDIN)*, 2012, pp. 1212–1217.

[26] S. Y. Cheung, S. C. Ergen, and P. Varaiya, "Traffic Surveillance with Wireless Magnetic Sensors," in *Proceedings of the 12th World Congress on Intelligent Transport Systems (ITS World Congress)*, 2005, pp. 1–13.

[27] G. Simon, M. Maróti, A. Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton, "Sensor Network-Based Countersniper System," in *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, 2004, pp. 1–12.

[28] 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*, vol. 46, no. 5, pp. 605–634, 2004.

[29] H. N. Koivo and E. Blomqvist, "Security in Sensor Networks: A Case Study," in *Proceedings of the 12th Mediterranean Conference on Control and Automation (MED)*, 2004, pp. 1–6.

[30] A. Tiwari, P. Ballal, and F. L. Lewis, "Energy-Efficient Wireless Sensor Network Design and Implementation for Condition-Based Maintenance," *ACM Transactions on Sensor Networks (TOSN)*, vol. 3, no. 1, pp. 1–23, 2007.

[31] K. Bur, P. Omiyi, and Y. Yang, "Wireless Sensor and Actuator Networks: Enabling the Nervous System of the Active Aircraft," *IEEE Communications Magazine*, vol. 48, no. 7, pp. 118–125, 2010.

TABLE I
OVERVIEW OF APPLICATION AREAS FOR WSN TECHNOLOGY AS IDENTIFIED IN RESPECTIVE LITERATURE.

		Arampatzis et al. [2]	Callaway Jr. [3]	Chong and Kumar [4]	Culler et al. [5]	Haenggi [6]	Karl and Willig [7]	Khemapech et al. [8]	Oliveira and Rodrigues [9]	Sohraby et al. [10]	Verdone et al. [11]	Wang et al. [12]	Zhao and Guibas [13]
Environmental Monitoring	Environment Detection and Monitoring									x		x	
	Environment Control and Biodiversity Mapping						x						
	Agriculture and Environmental Monitoring		x			x							
	Environmental Monitoring	x						x			x		x
	Soil Moisture and Temperature Monitoring								x				
	Environmental and Habitat Monitoring			x	x								
	Habitat Monitoring				x			x	x				
	Ecophysiology				x								
	Weather Forecasting								x				
	Scientific Exploration											x	
	Urban Terrain Mapping				x								
Disaster Control	Disaster Prevention and Relief				x		x					x	
	Emergency Response				x								
	Chemical Hazardous Detection								x				
	Fire and Civil Structures Deformations Detection								x				
	Flooding Detection								x				
	Earthquake Detection								x				
	Volcano Eruption								x				
	Structural Monitoring				x								
Smart Spaces	Building and Office Control								x				x
	Intelligent Buildings / Home Intelligence		x				x	x		x	x	x	
	Industrial Applications (e.g., Smart Store)										x		
	Industrial Control and Monitoring		x										
	Facility Management						x						
	Indoor Climate Control				x								
Monitoring	Logistics						x				x		
	Telematics						x						
	Tracking (Inventory System)							x					
	Asset and Warehouse Management												x
	Asset Tracking and Supply Chain Management		x		x								
Human	Medicine and Health Care	x	x		x	x	x	x		x	x	x	x
	Mood-Based Services										x		
	Entertainment										x		
	Interactive Surroundings											x	
Traffic	Transportation										x		
	Automotive												x
	Traffic Monitoring			x					x				
	Vehicle Tracking							x					
Security	Military Sensing	x	x	x		x		x		x		x	x
	Surveillance			x	x								x
	Intelligent Alarms				x								
	Treaty Verification				x								
Process	Industrial Process Control												x
	Manufacturing Process Control				x								
	Machine Surveillance and Maintenance			x	x		x				x		
Others	General Engineering					x							
	Civil Engineering					x							
	Ubiquitous Computing Environments				x								
	Precision Agriculture				x		x						
	Positioning and Animals Tracking										x		
	Robotics	x											

Development of a Contiki border router for the interconnection of 6LoWPAN and Ethernet

Viktor Eichmann, Thomas Scheffler
Beuth Hochschule für Technik Berlin
University of Applied Sciences
Luxemburger Str. 10, 13353 Berlin, Germany
Email: eichmann-viktor@web.de, scheffler@beuth-hochschule.de

***Abstract*—This paper presents implementation details of our prototypical 6LoWPAN border router. The border router is built around a 32-bit ARM Cortex M3 microcontroller and runs the network-enabled operating system Contiki in version 2.6. It uses a wireless network interface based on the IEEE 802.15.4 standard as well as an IEEE 802.3 Ethernet PHY. The network layer uses IPv6 and Layer-3 forwarding between these different link-layer technologies. The paper gives a description of the necessary adaptations to the Contiki operating system and provides an outlook on future implementation options.**

I. Introduction

Many researchers and market participants expect that the next revolution of the Internet comes from the interconnection of many, so called *smart objects*. These are inconspicuous electronic devices, equipped with sensors or actuators, a microcontroller, and a communication device that will form the backbone of the *Internet of Things* [1], [2].

These smart objects have to be integrated into existing network infrastructures mainly through a wireless link layer. A good candidate is the Low-Power WPAN *IEEE 802.15.4* [3], which has been designed to provide low-bitrate network connectivity efficiently and at minimal cost. It can be used with different network stacks such as Zigbee [4] or IPv6 [5].

The Internet of Things will have to use IPv6, which has a pool of 2^{128} addresses and offers more than enough room to grow the Internet into the physical world. In order to run IPv6 on IEEE 802.15.4 links, with a frame size of 127 octets, an adaptation layer is necessary that provides link-specific fragmentation and reassembly, as well as header compression. This service is provided by *6LoWPAN* [6], [7].

6LoWPANs are usually stub-networks that work with resource constrained devices (memory, processing power and energy). It is therefore important to reduce packet overhead, and bandwidth consumption, as well as processing requirements. RFC 6568 introduces the concept of a *LoWPAN Border Router* (LBR) [8] that is responsible for network coordination, address configuration and network interconnection.

II. Routing in 6LoWPANs

IEEE 802.15.4 network devices use low power radios, that imply a typical signal range in tens of meters, and even less in noisy and obstructed environments. Therefore the standards foresee the support for mesh scenarios, where two devices do not require direct reachability in order to communicate. However, neither IEEE 802.15.4 nor RFC 4944 define mechanisms for the operation and management of such mesh networks.

Given the principal design of network nodes in 6LoWPANs, several challenges exist: devices may be battery powered and implemented on microcontrollers with just a few kByte of RAM. RFC 6606 cites the following challenges for mesh networking in 6LoWPANs [9]:

- low overhead on data packets
- low routing overhead
- minimal memory and computation requirements
- support for sleeping nodes (saving battery)

There exist two principal approaches for packet routing in 6LoWPANs: *route-over* and *mesh under* [10]. The route-over approach routes packets on the IP level which has the implication, that intermediate nodes in the network have to make forwarding decisions. The mesh-under approach treats the 6LoWPAN mesh as a single IP hop, similar to network technologies such as Ethernet or WiFi, that provide a single broadcast domain.

Both approaches have benefits and drawbacks. The most prominent drawback of the mesh-under approach is the high overhead associated with the provisioning of a multicast service that supports the IPv6 neighbor discovery protocol and the hidden topology of the network, which prevents the IP- and application layer to optimize performance. Because of these limitations, almost all current implementations (including ours) implement a route-over solution.

The route-over approach also provides some challenges. The 6LoWPAN is a single IPv6 subnet and traditional prefix-based forwarding does not work. Intermediate routers might experience temporary link loss due to changing channel conditions, node mobility or sleeping devices. Packet forwarding in an 6LoWPAN is usually done over a single wireless interface. Devices have very little memory and any routing protocol will have to minimize state.

The IETF developed the *Routing Protocol for Low-Power and Lossy Networks* (RPL) for networks with severely constrained resources [11]. It works under the assumption that a few administratively chosen devices form the root of a *directed acyclic graph* (DAG). Routes exist as up and down routes and also intermediate nodes in the 6LoWPAN can act as routers

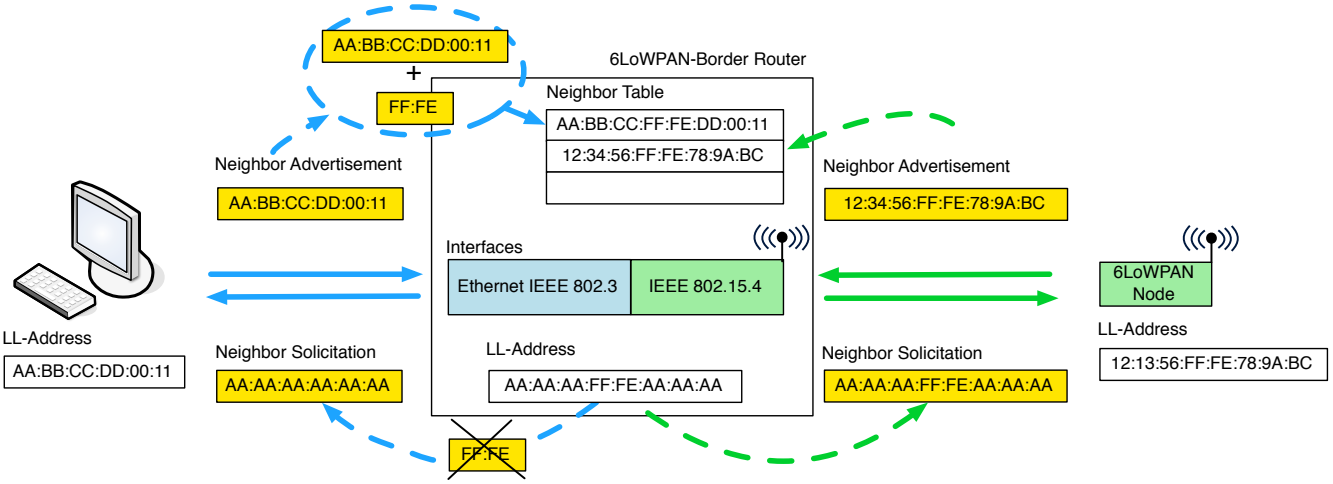

Fig. 1. Packet forwarding and address rewriting at the RPL border router

to provide multi-hop forwarding beyond the radio coverage of the DAG root.

Depending on the memory situation of the devices, RPL can be run in storing or non-storing mode. Intermediate nodes in non-storing mode do not keep downward routes. They only have enough memory to keep state about their connected parents and propagate information about connected children periodically in the form of a *destination advertisement object* (DAO) to the root node. The root node collects all the individual parent information and calculates reachability information for the whole 6LoWPAN. When a packet needs to be send to a node within the 6LoWPAN, the root inserts appropriate source-routes into the packet so that intermediate nodes can make adequate forwarding decisions without having to maintain a full routing table.

In storing mode the intermediate nodes keep downward routing information in their routing table and propagate only the destination addresses and prefixes for which a node has routes. Upward routes are handled as default routes chosen during the parent selection process.

The *Contiki OS* [12] is an open source operating system that supports a wide variety of devices and has a tiny memory footprint, which allows it to run on 8-bit microcontrollers with 8 kByte RAM and upwards of 30 kByte ROM. The Contiki OS is completely written in C, so it becomes relatively easy to port it to alternative hardware platforms and reuse existing application code. Since version 2.6 the Contiki OS supports the RPL routing protocol, which is operating in storing mode.

III. Related Work

Currently, there does not exist any freely available and usable implementation of a 6LoWPAN stack for desktop operating systems such as Linux or Windows. The Linux ZigBee project[1] aims to provide a complete implementation of the IEEE 802.15.4 and 6LoWPAN protocol. However, progress has been relatively slow and only recently[2] has it become possible to exchange basic IPv6 messages between the Linux stack and Contiki.

The existing, prototypical RPL implementation for Linux *SimpleRPL*[3] has only limited functionality and is not yet able to communicate successfully with other 6LoWPAN implementations due to kernel issues. Therefore, all known RPL border router implementations so far use the Contiki 6LoWPAN stack in different configurations:

A possible solution for the creation of a 6LoWPAN border router requires the use of an RZRaven USB stick[4] from Atmel, which runs a complete Contiki installation. The stick can be installed as a network interface on a computer running a desktop operating system such as Linux or Windows. The device then acts as a bridge between the WPAN and the global IPv6 Internet [13]. Limitations are the low number of possible RPL routes that this solutions supports (due to the low memory size of 8kByte on the Atmega AT90USB1287) and the cumbersome network configuration that requires the addition of explicit MAC-addresses into the neighbor table for certain operating systems.

Beerli and Fischer [14] developed a solution, where they ported Contiki to a platform that has native Ethernet interface support and a microcontroller with more memory and processing power than the Atmel MCU. However, this design is not freely available and uses Contiki to bridge packets between the Ethernet and the IEEE 802.15.4 network, instead of routing them at Layer 3.

A third implementation approach for a 6LoWPAN border router is chosen by the 6lbr project [15]. They use the Contiki OS compiled as a native process on Linux to handle the border router task. This approach gives the most freedom from resource constraints, however it also requires a much

[1] http://sourceforge.net/projects/linux-zigbee/

[2] https://archive.fosdem.org/2014/schedule/event/deviot04/

[3] https://github.com/tcheneau/simpleRPL

[4] http://www.atmel.com/Images/doc8120.pdf

more powerful hardware platform to run. The code could also be run on a microcontroller, but would require a porting effort, especially since their work very much focuses on the provisioning of multiple border routers to provide redundant points of attachment for the 6LoWPAN.

We choose to follow the path taken by Beerli and Fisher and develop our own hardware platform. However, we also want to make the necessary modifications to Contiki to support packet forwarding and routing between interfaces with different hardware address sizes. We also plan to open-source our implementation, so that it can be used by other projects that need a stand-alone RPL border router.

Figure 1 shows the network layout and the packet rewriting process that will be explained in the following section. Devices to the left of the border router are normal IPv6 network nodes that connect via Ethernet with the border router. Devices to the right of the border router run a RPL-enabled 6LoWPAN stack. We have successfully tested our prototype with several sensor nodes running Contiki in a multi-hop configuration.

IV. Design of the Contiki border router

Our solution of a prototypical 6LoWPAN border router uses a 32-bit ARM Cortex M3 CPU from ST-Microelectronics with 64 kByte RAM, 256 kByte FLASH memory and integrated Ethernet MAC (ST32F107RCT6)[5]. The border router has a 10/100 Mbit IEEE 802.3 Ethernet and an IEEE 802.15.4 interface, operating in the 2.4 GHz band. Figure 2 shows a picture of the final device.

IEEE 802.15.4 and Ethernet use different L2 address sizes. IEEE 802.15.4 uses EUI-64 addresses, whereas Ethernet uses EUI-48 addresses. There exists a defined mapping from the EUI-48 to the EUI-64 format. The EUI-48 address is split after the first 24 bit and FF:FE$_{hex}$ is inserted. The Contiki OS uses this mechanism to provide Ethernet-compatible L2 addresses for 6LoWPAN nodes. The border router implementation therefore needs to strip of, or insert the FF:FE$_{hex}$ as frames are exchanged between networks.

Figure 1 shows the packet rewriting between the *IEEE 802.3* Ethernet and an *IEEE 802.15.4* WPAN, using 6LoWPAN as an adaption layer to enable global communication via IPv6. Contiki stores link-layer addresses in the EUI-64 format in the neighbor table. We implemented an address rewriting procedure that converts between the EUI-64 and the EUI-48 format for incoming and outgoing Ethernet frames.

V. Implementation Details

Our implementation is based on Contiki OS in version 2.6. The current packet forwarding code in the Contiki OS uses a single neighbor table to store important information about reachable targets such as the IP address, L2 address and others.

Typical router implementations require information about the outgoing interface for a network route. However, Contiki in version 2.6 supports only single interface routing, so there is no place to store the interface information. In order to facilitate

[5]http://www.st.com/web/en/catalog/mmc/FM141/SC1169/SS1031

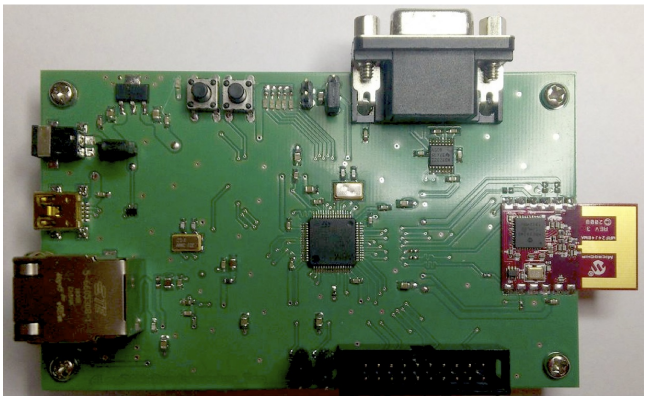

Fig. 2. A picture of the final hardware implementation

forwarding between different IPv6 networks, we extended the data structure of the neighbor table with a new entry `com_medium` that will be used to store the interface type. The 8-bit variable `com_medium` has currently three defined values: `2` - for Ethernet, `1` - for 802.15.4 and `0` - for situations where the interface type has not yet been determined. We only store the interface type and no interface identifier, because we assume that Contiki routers will have few interfaces with distinct types, so that it is not necessary to explicitly number them. Figure 3 shows this extended neighbor table.

We modified the `output()` function of the Contiki IP stack, so that before the IP packet is handed over to the device driver, our own output function takes over, retrieves the `com_medium` entry from the neighbor table and makes any address modifications as necessary (cf. Figure 1). Similar code has been added to the Ethernet input function, that extends a received L2 address to make it compatible with the Contiki data structures and updates the neighbor table.

When Contiki is configured as a router, the normal IPv6 network autoconfiguration is disabled, because it is assumed, that the node will be configured via RPL. This makes sense for the wireless interface, but less so for the Ethernet side. We wanted the 6LoWPAN router to behave just like a normal network node in the Ethernet-Segment, using *Stateless Address Autoconfiguration* (SLAAC) for the prefix and default router-configuration on the Ethernet interface. Doing so required a few changes in several source files that are related to the processing of ICMPv6 Router Solicitation and Router Advertisement messages.

uip_ds6_nbr_cache[9]	{...}
isused	1
ipaddr	{...}
lladdr	{...}
reachable	{...}
sendns	{...}
last_lookup	72412
nscount	1
isrouter	0
state	1
com_medium	2

Fig. 3. Neighbor table in the Contiki OS

VI. Summary and outlook

We developed a practical solution that extends the current routing functionality found in the Contiki operating system and implemented it in a working prototype that can be directly attached to an Ethernet segment. Our border router might be used to easily build meshed WPANs that can incorporate a large number of smart objects within a geographic area and make them globally reachable. We are currently investigating the long term stability and actual resource requirements of our solution. We expect this solution to scale much better than current solutions, based on the Atmel Raven USB stick (cf. Section III), mainly because the ARM Cortex M3 provides 8-times more RAM than the Atmel MCU. We also expect our solution to be much more energy efficient than a Linux or Windows-based installation.

Further on, we want to automate 6LoWPAN deployment by implementing the DHCP Prefix Delegation mechanism on the router [16]. The current 6LoWPAN prefix is statically configured on the border router and a static route for the 6LoWPAN needs to be installed on the upstream router. DHCP PD would make the network deployment much more robust and automatic. A stand-alone RPL border router with these features can be used to easily set-up networks of smart objects that can provide secure and robust automatisation solutions, as described in [17].

The IETF *Home Network*[6] working group is currently developing a set of standards that aid the configuration of routed networks in residential homes. We monitor these standards and investigate if they could be implemented on the 6LoWPAN border router.

Finally, we also investigate security issues. It might be very easy for an attacker to overpower the network, because 6LoWPANs have very limited resources. We currently evaluate security requirements and investigate how message filtering and rate-limiting on the gateway router can protect the smart objects and the surrounding network from resource exhaustion attacks.

References

[1] J.-P. Vasseur and A. Dunkels, *Interconnecting Smart Objects with IP: The Next Internet*. Morgan Kaufmann, 2010.

[2] IPSO, "IP for Smart Objects (IPSO) Alliance." [Online]. Available: http://ipso-alliance.org/

[3] J. Gutierrez, M. Naeve, E. Callaway, M. Bourgeois, V. Mitter, and B. Heile, "IEEE 802.15.4: A developing standard for low-power low-cost wireless personal area networks," *IEEE Network Magazine*, vol. 15, no. 5, pp. 12–19, September/October 2001.

[4] ZigBee Allicance. [Online]. Available: http://www.zigbee.org

[5] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," Internet Engineering Task Force, RFC 2460, Dec. 1998. [Online]. Available: http://tools.ietf.org/html/rfc2460

[6] N. Kushalnagar, G. Montenegro, and C. Schumacher, "IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals," Internet Engineering Task Force, RFC 4919, Aug. 2007. [Online]. Available: http://tools.ietf.org/html/rfc4919

[7] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks," Internet Engineering Task Force, RFC 4944, Sep. 2007. [Online]. Available: http://tools.ietf.org/html/rfc4944

[8] E. Kim, D. Kaspar, and J. Vasseur, "Design and Application Spaces for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)," Internet Engineering Task Force, RFC 6568, Apr. 2012. [Online]. Available: http://tools.ietf.org/html/rfc6568

[9] E. Kim, D. Kaspar, C. Gomez, and C. Bormann, "Problem Statement and Requirements for IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing," Internet Engineering Task Force, RFC 6606, May 2012. [Online]. Available: http://tools.ietf.org/html/rfc6606

[10] J. Hui and D. Culler, "IPv6 in Low-Power Wireless Networks," *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1865 –1878, November 2010.

[11] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks," Internet Engineering Task Force, RFC 6550, Mar. 2012. [Online]. Available: http://tools.ietf.org/html/rfc6550

[12] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," in *Proceedings of the 29th Annual IEEE International Conference on Local Computer Networks*, ser. LCN '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 455–462.

[13] A. Dunkels, "Jackdaw RNDIS RPL border router," Tech. Rep., 2012. [Online]. Available: https://github.com/contiki-os/contiki/wiki/Jackdaw-RNDIS-RPL-border-router

[14] U. Beerli and S. Fischer, "Ethernet to WPAN gateway with IPv6 support," Züricher Fachhochschule, Tech. Rep., 2009. [Online]. Available: https://ba-pub.engineering.zhaw.ch/WebPublication/Flyer.pdf?version=Bachelorarbeit2009&code=BA09_FS_mema_1

[15] L. Deru, S. Dawans, M. Ocaña, B. Quoitin, and O. Bonaventure, "Redundant Border Routers for Mission-Critical 6LoWPAN Networks," in *Proceedings of the Fifth Workshop on Real-World Wireless Sensor Networks*, 2013.

[16] O. Troan and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6," Internet Engineering Task Force, RFC 3633, Dec. 2003. [Online]. Available: http://tools.ietf.org/html/rfc3633

[17] S. Zehl and T. Scheffler, "Secure access and management of Smart Objects with SNMPv3," in *Proceedings of the Third IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin)*, P. Friedrich, H. L. Cycon, B. Wolf, and J. Clauss, Eds. IEEE, 2013, pp. 366–370.

[6] http://datatracker.ietf.org/wg/homenet/charter/

langOS - A Low Power Application-specific Configurable Operating System

Oliver Stecklina, Andreas Krumholz and Stephan Kornemann
IHP
Im Technologiepark 25
15236 Frankfurt (Oder), Germany
Email: {stecklina, krumholz, kornemann}@ihp-microelectronics.com

The implementation of an application for a wireless sensor node is driven by various factors. This may be the application-specific requirements first, but also the chosen platform, the constrained resources of the used micro controller, the preferred or given network protocol as well as security demands. In contrast to the broad variety of requirements a software developer wants to use a single operating system (OS) for all the different applications. Due to the fact that a full-featured OS cannot be implemented on a sensor node a configurable solution is demanded. In this paper we present langOS, which is a compile-time configurable OS. A developer can easily configure the required feature-set and will have a minimal run-time overhead. Due to a human-readable, non-language specific configuration file it is very easy to use. Furthermore, langOS is implemented in a standard programming language without any extensions. Therefore, we are convinced that by using langOS the implementation of tailor-made sensor node application becomes very easy.

I. Introduction

The acronym langOS stands for low power application-specific configurable operating system. In this paper we will shortly introduce the basic idea and the key features of langOS. Although many OSs for wireless sensor networks (WSNs) are already implemented, we will argue that our approach is different from these one. To achieve an application-specific configuration with a minimal memory footprint and run-time overhead, langOS uses a predecessor tool to create a tailor-made set of OS features. During compilation only the selected features are integrated and the interfaces are configured statically.

It is a fact that for a wireless sensor node (mote) in order to become more efficient, reliable and secure in real world application scenarios the software architecture should be highly configurable. Furthermore, due to the limited resources of a mote only a specific set of functionality can be implemented at one time. In recent years various OSs were designed with dynamic configuration schemes and some of them support a partial online update as well. But a dynamic configuration set and an update scheme cause an additional overhead. By using a static run-time configuration this overhead can be mostly avoided. Moreover, in cyber-physical systems (CPSs) where security and reliability are key such a static configuration may be a better option to achieve a demanded security level.

In the following section we give a brief overview of mote OSs and their configuration schemes. In Section III we describe the basic ideas of langOS and list some of the implemented features. We conclude this paper with an outlook on further work and a short summary.

II. Mote operating systems

The basic idea of the langOS configuration process was inspired by the configuration schemes of Linux and eCos [1]. Both provide a menu-driven user interface, which allows a user to choose the feature-set of their systems. A predecessor tool generates include files for the build process and header files, which are used within the source files by the C preprocessor. A similar approach is provided by the Atmel Software Framework (ASF) [2]. The ASF provides software drivers and libraries to build applications for Atmel devices. These tools simplify the configuration process and solve dependencies automatically. But all these solutions do not fit to the requirements of OSs of low power wireless sensor nodes with their restricted resource. Linux as well as eCos are designed for powerful devices and the ASF does not include OS capabilities.

Within the area of WSNs TinyOS is still the state-of-the-art mote OS. It provides an event-driven operating environment and uses a component model for design [3]. The components of TinyOS are written in the nesC language. The language is an extension to C and was designed to embody the structuring concepts and execution model of TinyOS [4]. The behavior of a component is specified by a set of interfaces. A TinyOS application is built out of components written in nesC, which are wired by interfaces to form the program. During the build process optimized C code is generated and the components are statically linked. This increases the runtime performance and minimizes the memory footprint of an application. A fully aspect driven approach is implemented by the CiAO operating system [5]. The OS follows the aspect-orient programming (AOP) and provides a highly configurable system that takes the different aspects into account. CiAO is written in AspectC++ a source-to-source weaver that transforms AspectC++ sources into ISO C++ code [6].

Within the last years several OSs for wireless sensor nodes were developed. Most of them provide a dynamic update scheme, e.g. SOS [7], Contiki [8] and RETOS [9]. SOS consists of dynamically loaded modules and a common kernel. Like TinyOS, SOS is an event-driven OS and uses a component module design. The kernel implements messaging, dynamic memory allocation and module loading and unloading. Basic memory protection techniques, watchdog timers and garbage collection are implemented as well. Contiki provides dynamic loading and unloading of programs and services. It supports an optional preemptive multi-threading. Furthermore, Contiki multi-threading is implemented as a library, which is linked to the system on demand. RETOS is also a multi-threaded OS. It separates the kernel from user applications and supports loadable kernel modules for a dynamic reconfiguration. Other OSs for embedded systems are focused on multi-threading (MANTIS [10]), real-timer scheduler (Reflex [11] and FreeRTOS [12]) or security (t-kernel [13]).

But all these systems are configured within the source code files. We are convinced that an highly and dynamic configurability is contradictorily. Rather, it must be considered during the design of the OS. Furthermore, we have the experience that a developer will faster familiar with an OS that is written in a standard programming language. The entry level for understanding the source code is much lower and the common acceptance is significantly higher.

III. LANGOS

langOS was designed to simplify the tailor-made implementation of mote's applications. We already used the OS for implementing a biochemical sensor capsule [14], a secure wakeup receiver [15] and a multi-hop routing protocol for energy efficient WSNs [16]. Although all these applications are tightly coupled with WSNs their specific demands are so different that a one-fits-all solution will be over-sized and inefficient. During the design of langOS a highly configurability by using a standard programming language was key. Furthermore, a minimal memory footprint and extended low power capabilities were focused.

A. Source code organization

The langOS sources are arranged in subdirectories where each subdirectory contains modules of a specific type. As shown in Figure 1 the `dev` directory contains for example all the device drivers. Other directories include protocol, service, storage or infrastructure modules. Among the modules dependencie exists, which must be taken into account by a developer.

The current version of langOS supports the MSP430 processor family only. Nevertheless, a port to other processor families will be easy. The OS is completely written in C, which simplifies a port to alternative platforms. The hardware dependent and the hardware independent program code are separated in different subdirectories. The modules of the hardware abstraction layer (HAL) are located in the subdirectory `hal`. Only these modules are specific for the microprocessor

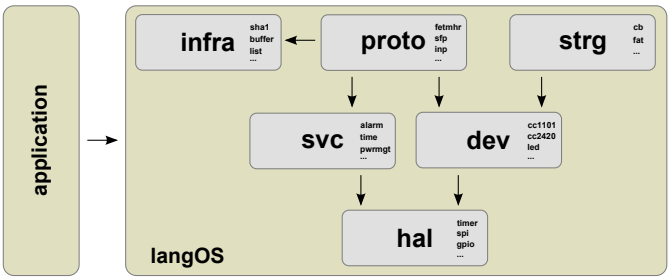

Fig. 1. The software structure of langOS is split in application and langOS sources. The langOS sources are structured in subdirectories, which contain modules of a specific type or functionality.

family. All the other modules use the HAL to access processor peripherals. The device drivers are located in the `dev` subdirectory. These modules are specific for external Integrated Circuits (ICs) and use the HAL modules for communication. For example the CC1101 sub 1 gigahertz transceivers uses general purpose input/output (GPIO) and serial peripheral interface (SPI), which are completely implemented in HAL modules. The device driver implementation is specific for the CC1101 IC but can use the SPI and GPIO of any other HAL implementation.

B. System configuration

Platform specific mappings are implemented by C macros. As shown in Figure 2 each module uses an abstract description of the electrical pinning. The description is mapped to the physical wiring by platform specific header files. We have implemented platform headers for the IHPnode [17], the IHPstack [18] and the TmoteSky [19]. Especially the IHPstack with its modular design benefits from this framework. Each module of the IHPstack is characterized by a single header file. A combination of these files describes the IHPstack mote. Hence, a developer can customize its application by choosing software modules as well as platform modules.

The configuration process is supported by a predecessor tool, which is written in Python. The configuration of a langOS application is described in a single text file. The file contains the module and platform selection as well as module specific parameters. The predecessor tool generates an application-specific header file, which contains glue code for the system initialization and includes in the platform specific headers. All langOS modules refer only to this file for getting the application's configuration. Furthermore, the same configuration file is used by the software build chain.

C. Application build

The build process of a langOS application is summarized in Figure 3. As mentioned above each application must have a configuration file. The file is processed by the predecessor tool and its output is used within the source files and the build system.

As shown in Figure 3 we distinguish between langOS sources and application sources. The application source files

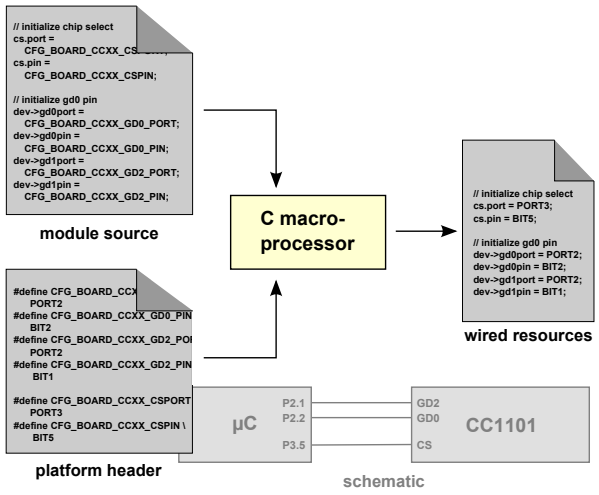

Fig. 2. Platform resources are described by a specific platform header. The C preprocessor combines the source with the header file and generates the wired resource file corresponding to the mote's schematic.

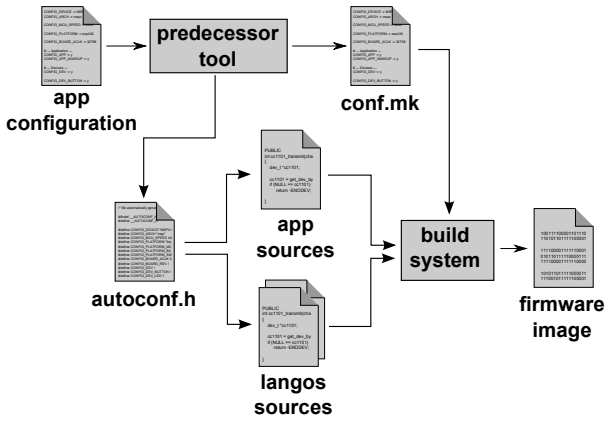

Fig. 3. The predecessor tool of the langOS build chain generates platform specific includes, which are included in the application and the langOS sources. The build system compiles the application and langOS separately and combines both to the target firmware image in a final single step.

are not part of langOS. Both are compiled and linked separately and are combined to a firmware image in a final step. We split these two parts to make a distribute development process with different source code control systems possible. Each application has its own configuration and its source code can be organized in an out-side subdirectory structure. Therefore, using langOS in different projects becomes easy to organize.

D. Boot-strap

Due to the configurable module selection of langOS an automated module initialization became necessary. Hence, the langOS predecessor tool generates an initialization array, which is traversed during the system's boot-strap. The module initialization calls the module's initialization function at least once. Due to the unpredictable order of their invocation, a manual module initialization may be required. In the current version of langOS a module developer has to take care about its initialization function that it initializes all required modules. A future version of the predecessor tool will include this step.

However, a module developer has the option to decide if its module is included in the automatic initialization. If the module includes a function with the pre-defined signature

```
int <module_name>_init(void)
```

an automatic invocation is setup by the predecessor tool. The initialization functions are called by the boot-strap code before any task is started. langOS features the concept of cooperative tasks. A cooperative task is implemented by a coroutine with a single entry point. The tasks are started by a simple round-robin scheduler just after finishing the boot-strap code.

E. Power management

Besides the configurability the design of langOS was focused on low power capabilities. The langOS core automatically enters the lowest possible low power mode (LPM) if no activity is detected. An activity is an active task or an active peripheral module. The MSP430 microprocessor supports five different LPMs. From the highest mode - LPM1 - to the lowest mode - LPM5 - more and more components of the microprocessor are disabled. Therefore, modules must know or must be able to control the entered power mode.

To control the power mode any activity can register itself at the power management module of langOS. With this registration a module can specify the lowest possible LPM and can define a suspend and a resume function. When entering a low power mode the power management module looks for the lowest possible LPM of all registered modules. Furthermore, all registered suspend functions are called before going to the LPM. If a suspend function returns an error code the suspend process is aborted. By that way each module can control the entered LPM and the suspend process itself. When leaving the LPM the registered resume functions are invoked.

F. Protocol stack framework

Due to the primary application scenario of wireless sensor nodes their OS must cover a broad variety of tailor-made protocol stacks. While this demand stands in contrast to the limited resources of the devices a configurable protocol stack framework is required. The protocol stacks of langOS are based on a flexible and configurable framework. As shown in Figure 4, the framework defines three types of protocol layers: endpoint, stack and device. The device layer defines the lower end of the protocol stack and implements the binding to the physical device. On top of the device layer a multiple number of stack layers can be stacked. A stack layer processes the data but does not define an endpoint of these. The stack is closed by a dedicated endpoint layer. This layer defines the final connection to the application.

The layers are connected by their `_receive()` and `_xmit()` and their corresponding `_done()` functions. Each layer must implement these functions. The data are passed by reference as an untyped container. Therefore, a type save binding is required. The binding is part of the configuration process and will be supported by the predecessor tool. Even

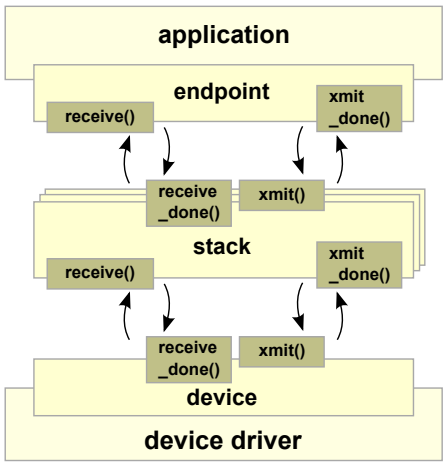

Fig. 4. The three layer types of the protocol stack framework. To guarantee interoperability each layer must implement the `receive()`, `xmit()` and `_done()` functions. The stack layer can be implemented multiple times for multi-step data processing.

though flexibility is lost the data processing as well as the packet size benefit by the presented framework. Due the fixed layer binding a protocol type field within the transmitted data is needless.

IV. Future work and conclusion

The current implementation of langOS is a hybrid between a compile-time and a run-time configurable OS. Unfortunately, both are not well implemented yet. In a next step we will focus our work on making langOS to a pure compile-time configurable OS. Therefore, we will replace run-time configuration switches and registration functions by compile-time options if possible. Furthermore, we will improve the capabilities of the predecessor tool. The current version does not include a dependency check and the generation of the module bindings are limited. Finally, we plan to publish the code to the open source community.

Our experiences by using OSs in WSNs have shown that a dynamic and complex configuration scheme decreases the acceptance of an operating system. Moreover, a dynamic update scheme is almost unnecessary and requires at lot of code that is used at run-time only once. In this paper we presented langOS, a compile-time configurable OS with a human-readable and easy to understand single configuration file. The OS is written in a standard programming language without any extensions. Both reduces the entry level for a new programmers. Furthermore, highly configurability was considered during the design of the OS. The OS provides set of well-defined interfaces and an automatically build chain. We are convinced that langOS simplifies the tailor-made implementation of an embedded application in a significant manner.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the Federal Ministry of Education and Research (BMBF) under grant agreement No. 16 KIS0004.

References

[1] A. J. Massa, *Embedded Software Development with eCos*. Upper Saddle River, NJ, USA: Prentice Hall Professional Technical Reference, 2002.

[2] Atmel, *Atmel AVR4029: Atmel Software Framework User Guide*, 2013.

[3] P. Levis, S. Madden, J. Polastre, R. Szewczyk, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler, "TinyOS: An operating system for sensor networks," in *in Ambient Intelligence*. Springer Verlag, 2004.

[4] 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 the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation*, ser. PLDI '03. ACM, 2003, pp. 1–11.

[5] D. Lohmann, W. Hofer, W. Schröder-Preikschat, J. Streicher, and O. Spinczyk, "CiAO: An Aspect-oriented Operating-system Family for Resource-constrained Embedded Systems," in *Proceedings of the 2009 Conference on USENIX Annual Technical Conference*, ser. USENIX'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 16–16.

[6] Y. Yanagisawa, K. Kourai, and S. Chiba, "A Dynamic Aspect-oriented System for OS Kernels," in *Proceedings of the 5th International Conference on Generative Programming and Component Engineering*, ser. GPCE '06. New York, NY, USA: ACM, 2006, pp. 69–78.

[7] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava, "A Dynamic Operating System for Sensor Nodes," in *Proceedings of the 3rd International Conference on Mobile Systems, Applications, and Services*, ser. MobiSys '05. New York, NY, USA: ACM, 2005, pp. 163–176.

[8] A. Dunkels, B. Grönvall, and T. Voigt, "Contiki - A Lightweight and Flexible Operating System for Tiny Networked Sensors," *Local Computer Networks, Annual IEEE Conference on*, vol. 0, pp. 455–462, 2004.

[9] H. Cha, S. Choi, I. Jung, H. Kim, H. Shin, J. Yoo, and C. Yoon, "RETOS: resilient, expandable, and threaded operating system for wireless sensor networks," in *Proceedings of the 6th international conference on Information processing in sensor networks*, ser. IPSN '07. New York, NY, USA: ACM, 2007, pp. 148–157.

[10] S. Bhatti, J. Carlson, H. Dai, J. Deng, J. Rose, A. Sheth, B. Shucker, C. Gruenwald, A. Torgerson, and R. Han, "MANTIS OS: An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms," *Mob. Netw. Appl.*, vol. 10, no. 4, pp. 563–579, Aug. 2005.

[11] K. Walther and J. Nolte, "Event-flow and synchronization in single threaded systems," in *Model-Based Testing, ITGA FA 6.2 Workshop on and GI/ITG Workshop on Non-Functional Properties of Embedded Systems, 2006 13th GI/ITG Conference-Measuring, Modelling and Evaluation of Computer and Communication (MMB Workshop)*. VDE, 2006.

[12] R. Barry, *Using the FreeRTOS Real Time Kernel - a Practical Guide*, ser. FreeRTOS Tutorial Books. freeRTOS, January 2010.

[13] L. Gu and J. A. Stankovic, "T-kernel: Providing reliable os support to wireless sensor networks," in *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, ser. SenSys '06. New York, NY, USA: ACM, 2006, pp. 1–14.

[14] N. Todtenberg, J. Klatt, S.-T. Schmitz-Hertzberg, F. Jorde, and K. Schmalz, "Wireless sensor capsule for bioreactors," in *Proceedings of IEEE MTT-S International Microwave Workshop Series on RF and Wireless Technologies for Biomedical and Healthcare Applications*, ser. IMWS-Bio '13, Singapore, Dec 2013, pp. 1–3.

[15] O. Stecklina, S. Kornemann, and M. Methfessel, "A Secure Wake-up Scheme for Low Power Wireless Sensor Nodes," in *Proceedings of the 4th International Workshop on Mobile Systems and Sensors Networks for Collaboration*, ser. MSSNC '14, Minneapolis, USA, May 2014.

[16] O. Stecklina, P. Langendörfer, and C. Goltz, "A Fair Energy Trade Multi-Hop Routing in Wireless Sensor Networks," in *Proceedings of the 6th Joint IFIP Wireless and Mobile Networking Conference*, ser. WMNC 2013, Dubai, UAE, April 2013.

[17] K. Piotrowski, A. Sojka, and P. Langendörfer, "Body Area Network for First Responders - a Case Study," in *Proceedings of the 5th International Conference on Body Area Networks*, Sep. 2010.

[18] O. Stecklina, D. Genschow, and C. Goltz, "TandemStack - A Flexible and Customizable Sensor Node Platform for Low Power Applications," in *Proceedings of the 1st International Conference on Sensor Networks*, ser. Sensornets 2012, Rome, Italy, February 2012.

[19] Moteiv, *Tmote Sky Datasheet http://www.sentilla.com/pdf/eol/tmote-sky-datasheet.pdf*, 2006.

Self-Stabilizing Aggregation- and Reduction-Structures for Wireless Sensor Networks

Sandra Beyer, Stefan Lohs, Reinhardt Karnapke and Jörg Nolte
Distributed Systems/Operating Systems Group
Brandenburg Technical University of Cottbus-Senftenberg
Email: sandra.beyer.sb@gmail.com
{slohs,karnapke,jon}@informatik.tu-cottbus.de

***Abstract*—Wireless sensor networks (WSNs) are prone to errors mainly because of unreliable communication. Errors cause data loss and can lead to complete node shutdowns. The increasing load closer to the sink in a data gathering scenario exaggerates this situation.**

Self-stabilizing algorithms can be used to deal with this problem. They deal with frequent occurring transient faults to enable long-term applications. The self-stabilizing tiers algorithm proposed in this paper provides a suitable structure for aggregation and reduction schemes to reduce data load.

***Index Terms*—WSN, self-stabilizing, aggregation, reduction, transient faults.**

I. Introduction

A wireless sensor network (WSN) consists of several up to thousands of small and cheap sensor nodes. Each node has a micro chip with low computation power and small memory (flash and RAM). For communication, it is equipped with a radio transceiver. To sense its environment, a node has different sensors, e.g., for measuring temperature, pressure, or motion. Each node is not designed for high performance, but rather for saving energy to enable long-living deployments. A sensor node is usually powered by batteries, sometimes energy harvesting modules are also used, which can theoretically sustain functionality arbitrarily long.

Despite the low computational power, a large amount of sensor nodes can solve complex tasks, e.g., monitoring, intrusion detection, or controlling actuators. In most cases a sensor node senses its immediate surroundings and transmits the measurement results to a distinct sink for processing. To cover large areas and to conserve the limited energy resources, adequate protocols are necessary.

In spite of its cheap deployment costs and its powerful ad-hoc capabilities, a WSN is prone to faults because of the unreliable wireless communication. Packet loss or corruption is common and unpredictable. Packet collisions, radio wave reflections, and irregular antenna characteristics cause transient faults which are hard to detect and disturb the system up to complete node failures. Experiments have shown that even in case of no mobility, the communication topology is changing frequently [1].

To guarantee durable systems, fault states have to be detected and repaired. Self-stabilizing algorithms (Section III) are a promising way to deal with this kind of faults while keeping the overhead reasonable.

Apart from the physical conditions, the communication pattern has to be considered, too when designing applications for WSN. In case of the converge-cast scenario, flow control is necessary to avoid congestion near the sink. A good way to reduce the network load is to apply aggregation or reduction functions. In general, aggregation reduces the load by concatenating several messages to one larger message, a reduction applies a function like minimum or maximum to avoid the transmission of unnecessary data.

To perform an aggregation or reduction, a distinct structure including all network nodes has to be established. The task of this structure is to determine which node is responsible for processing the data at the current step and whichroute the aggregated data has to take through the network.

In this paper we present a self-stabilizing approach to generate an aggregation structure. The presented tree and tier algorithms autonomously repair the structure after the occurrence of a fault. The advantage of self-stabilization is the inherent tolerance against all transient and a large set of permanent faults.

This paper is structured as follows: First, we offer a state of the art analysis. Section III gives a general overview of self-stabilizing algorithms in WSN. Our proposed SS-TIER algorithm for an aggregation structure is presented in Section IV. Finally, a competitive evaluation is shown in Section V.

II. State of Art

For data aggregation in WSN, a number of approaches using different structures can be found in literature. PEDAP [2] and TAG [3] are examples for aggregation services based on a tree structure. Both algorithms start by constructing a tree with the sink as root, then this base structure is used for several aggregation rounds. While PEDAP renounces the use of a repair mechanism, TAG monitors and optimizes the tree during the aggregation. However, a fault detection for the aggregation structure is not included.

A second approach is to organize the network into clusters. A defined cluster head performs an aggregation step on the data of its subordinate nodes. Afterwards, the head sends the data to the sink for further processing. An example for a

cluster based aggregation is LEACH [4]. It establishes clusters with respect to energy by periodically changing cluster heads to prolong the lifetime of the network. For constructing the aggregation structure, a decentralized approach is used. Each node needs a direct link to the sink, which decreases the possible size of the deployment.

PEGASIS [5] is a chain based aggregation service. The idea is to save energy by decreasing the signal strength of the radio module. Each node communicates only with its nearest neighbor. To construct this structure, a global view is necessary. PEGASIS does not provide repair mechanisms for local faults.

To increase fault tolerance, *Synopsis Diffusion* [6] uses multiple paths to a sink. Instead of a tree, a tier structure is generated. Each message is sent to all reachable nodes which are closer to the sink. In case of a single link break, the data will mostly not be lost, because there are often multiple recipients.

Aggregation schemes can be set up with different types of base structures. All these examples have in common that they first establish the structure and than use it for several steps. In case of a changing communication topology, faults will only be repaired during a complete reconstruction.

III. Self-Stabilization

The concept of self-stabilizing algorithms was first introduced by Dijkstra in his paper "Self-Stabilizing Systems in Spite of Distributed Control" [7]. He describe a network of several processors having a set of registers. Each processor has a local view, this means it can read its own and the registers of all direct neighbors. The values of the registers of a processor are called the local state of the processor. All local processor states together define the state of the system.

A self-stabilizing algorithm consists of a set of rules in the form $guard \rightarrow assignement$. The guard is a predicate which is based on the local view. If the guard of a rule is resolved to *true*, the rule is called *enabled* and the assignment part *may* be executed. The assignment part of the rule modifies the local state of the node.

By monitoring the neighborhood, i.e, the local view, and controlling its own state, a node is influencing the system, which will converge from any arbitrary state into a global stable state. An algorithm is called self-stabilizing if the time needed to converge into a stable system state is finite. Applying a rule while in a stable state never leads to an unstable state.

This model has several advantages which makes it interesting for WSN. First, in most cases a small set of rules is enough to describe an algorithm. It is not necessary to define any fault state. The second advantage is the decentralized approach. Each processor or sensor node only observes its local neighborhood. Nevertheless, the execution of the rules at each node establishes a stable global state. The last and major benefit is the inherent fault tolerance. If a fault occurs, the stable state gets corrupted, and after a finite time the algorithm converges (again) into a stable state. Also, each ad-hoc deployment of nodes can be handled as a fault state and will be autonomously stabilized.

IV. Aggregation- and Reduction-Structure

In this section we describe two self-stabilizing algorithms for a data gathering scenario. The goal is to achieve a stable routing topology where all nodes send their measured data to one distinct node, e.g., a gateway. The first is a minimum spanning tree algorithm introduced by Dolev [8]. The second is our self-stabilizing tiers algorithm.

```
1  algorithm dolevtree;
2
3  map NodeID platform.nodeId nodeId;
4
5  public NodeID parent;
6  public Integer level;
7
8  declare Integer minLevel
9      := min{v.level | Neighbors v};
10 declare NodeID sinkId := 0;
11
12 Rule 1:
13   (nodeId = sinkId)
14     and !((parent = 0 )
15     and (level = 0))
16   -> parent := 0;
17      level := 0;
18
19 Rule 2:
20   !(nodeId = sinkId)
21     and !(level = minLevel + 1)
22     and !(exists{v | Neighbors v:
23       (v.level = minLevel)
24       and (v.nodeId = parent)})
25   -> parent := choose{v.nodeID |
26        Neighbors v :
27        (v.level = minLevel)};
28      level := minLevel + 1;
```

Listing 1. $\mathcal{A}_1$ Minimum spanning tree algorithm of Dolev

Both algorithms consist of two rules, shown in Listing 1 and 2 respectivly. The first rule of each algorithm is executed by the sink, more precisely: the node which requests the aggregated data. In both algorithms the second rule is run by all other nodes. In case of the tree algorithm, each node selects a parent node with a minimum distance to the sink and updates its own distance. Algorithm $\mathcal{A}_2$ computes only the hop distance to the sink.

A stable state is achieved if each node has correct knowledge of the distance to the sink and, in case of Algorithm $\mathcal{A}_1$, a correct parent node. Both algorithms have in common that no rule is enabled once a stable system state is established.

```
1  algorithm ssTiers;
2
3  map NodeID platform.nodeId nodeId;
4  public Integer tier;
5
6  declare Boolean hasRootTier := tier = 0;
7  declare Integer minTier :=
8    min{v.tier | Neighbors v};
9  declare NodeID sinkId := 0;
10
```

```
11 Rule 1:
12   (nodeId = sinkId) and !hasRootTier
13   -> tier := 0;
14 Rule 2:
15   (nodeId != sinkId)
16     and (tier != minTier + 1)
17   -> tier := minTier + 1;
```

Listing 2. $\mathcal{A}_2$ Self-stabilizing tiers algorithm

A tree provides a direct order for an aggregation or reduction. A node is responsible for processesing the data of all its child nodes (nodes which select it as parent). The resulting data is sent to its parent. Each value from a node takes only one route, if one communication step is corrupted the value is lost. For reduction, each associative and commutative function can be applied.

In case of the tiers structure, a value takes multiple paths to a sink node. A node processes the received data of nodes from its own $tier+1$. The result is sent to all reachable nodes at $tier-1$. This increases the tolerance against single communication errors. However, not all aggregation and reduction functions are resistent to duplicates, which can falsify the results. In case of not resistent functions, an adaquate duplicate suppression has to be provided by the aggregation scheme.

V. Evaluation

The goal of self-stabilization is to increase the tolerance against transient faults. To evaluate the performance, we integrated Algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$ in a middleware based on self-stabilizing algorithms for WSN [9]. For comparison, we implemented two ordinary tree and tier algorithms with a periodic reconstruction of the logical topology.

For the aggregation, it is necessary to have a stable and fault free aggregation structure. To show the advantages of our approach we simulated all four algorithms and observed the tree and tier structures, respectively. At runtime, we injected transient faults like link breaks. Each algorithm dealt with the same set of fault scenarios.

A. Initial Convergence

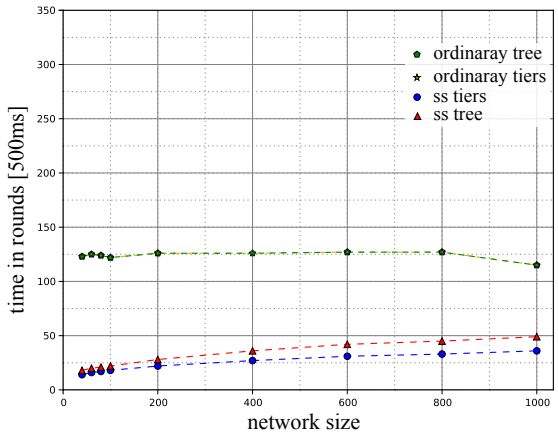

Fig. 1. Initial convergence time; pentagon and star are overlapping

First, we show the time needed to establish a stable structure after a reset of the network. Figure 1 shows that the self-stabilizing algorithms need approximately half the time needed by the ordinary algorithms (the ordinary approach produces the same results). The reason for this is the neighborhood discovery protocol on the sensor nodes.

After a reset no neighbors are known and the neighborhood discovery protocol starts to explore the network. Nodes which fulfill a quality criterion are added to the neighbor tables. In case of the standard algorithms, the construction beacon is generated after the reset and sent to all neighbors. This means that it is lost, because no neighbor is currently known. The construction is restarted after a timeout. In case of the self-stabilization, the first messages (node states) are also lost. After receiving the first packets from neighbors, the self-stabilizing algorithms are able to establish a tree or tier structure.

B. Runtime Behavior

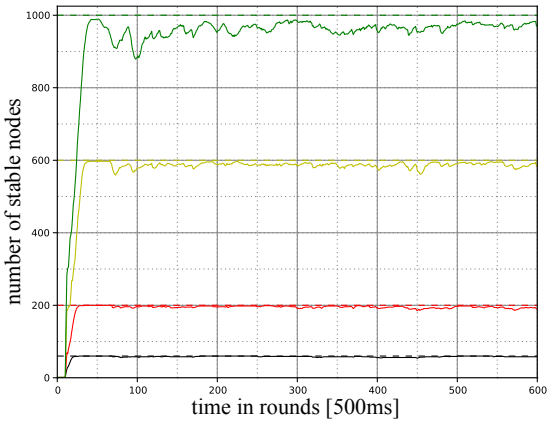

Fig. 2. Structure state over time. Algorithm $\mathcal{A}_2$ with varying number of nodes (75, 200, 600, 1000).

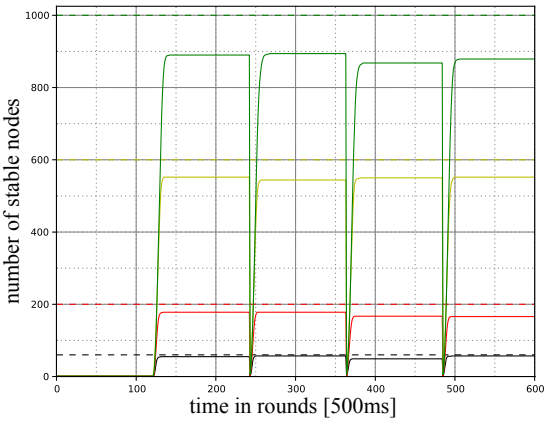

Fig. 3. Structure state over time. Ordinary tiers algorithm with varying number of nodes (75, 200, 600, 1000).

In the next step we focus on the behavior during the runtime by injecting transient link breaks. In simulations, our approach

(Figure 2) achieved a higher amount of nodes with an correct structure than the ordinary approach (Figure 3). The figures indicate that the ordinary algorithm is not able to construct a tier structure with more than 90 % of the nodes. The reason for this is that the construction message is lost if a node is temporarily not reachable due to link breaks. In case of the self-stabilizing algorithms, a node which *reconnects* to the network automatically rejoins the tiers structure. The ordinary algorithm has to wait for the reconstruction beacon.

C. Experiments with WSN hardware

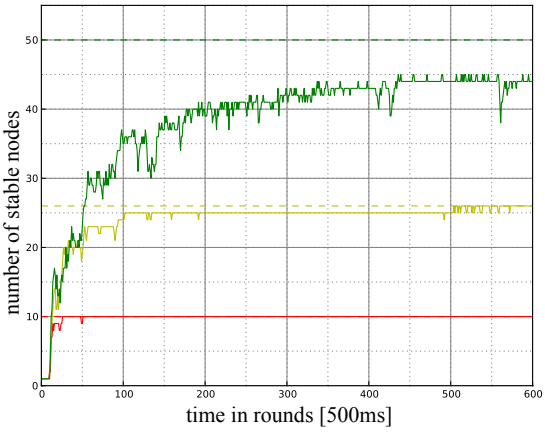

Fig. 4. Structure state over time. Algorithm $\mathcal{A}_1$ with varying number of nodes (10, 26, 50).

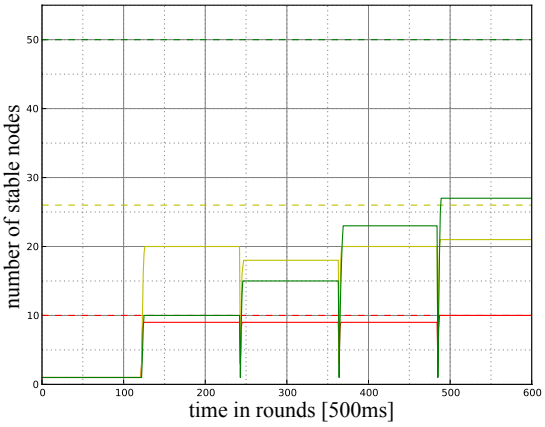

Fig. 5. Structure state over time. Ordinary tier algorithm with varying number of nodes (10, 26, 50).

After the simulations, we ran several real world experiments to confirm the results, as depicted in Figures 4 ($\mathcal{A}_2$) and 5 (ordinary tier). As can be seen, the amount of nodes which join the structure is also higher for our approach. Attention should be paid to the fact that we did not inject errors into the network during the experiments. All errors are caused by the use of real radio hardware and the environmental conditions at our testing site (university gym).

VI. Conclusion

In this paper we presented a self-stabilizing tiers algorithm for WSN. Simulations and experiments substantiate that more network nodes are part of the resulting structure in case of our approach than compared to an ordinary approach. This supports our opinion that a self-stabilizing algorithm is suitable for increasing the fault tolerance of WSN and increases the quality of aggregation and reduction schemes based on these routing structures.

The next step is to implement an aggregation scheme upon these structures. We will investigate if self-stabilizing algorithms are suitable for this purpose as well.

Acknowledgments

ToleranceZone is funded by the Deutsche Forschungsgemeinschaft (DFG NO 625/6-1).

References

[1] S. Lohs, R. Karnapke, and J. Nolte, "Link stability in a wireless sensor network–an experimental study," in *Sensor Systems and Software*. Springer, 2012, pp. 146–161.

[2] H. O. Tan and I. Körpeoğlu, "Power efficient data gathering and aggregation in wireless sensor networks," *SIGMOD Rec.*, vol. 32, no. 4, pp. 66–71, Dec. 2003. [Online]. Available: http://doi.acm.org/10.1145/959060.959072

[3] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "Tag: A tiny aggregation service for ad-hoc sensor networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 131–146, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844142

[4] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," in *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*. IEEE, 2000, pp. 10–pp.

[5] S. Lindsey, C. Raghavendra, and K. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 13, no. 9, pp. 924–935, Sep 2002.

[6] S. Nath, P. B. Gibbons, S. Seshan, and Z. R. Anderson, "Synopsis diffusion for robust aggregation in sensor networks," in *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '04. New York, NY, USA: ACM, 2004, pp. 250–262. [Online]. Available: http://doi.acm.org/10.1145/1031495.1031525

[7] E. W. Dijkstra, "Self-stabilizing systems in spite of distributed control," *Commun. ACM*, vol. 17, no. 11, pp. 643–644, Nov. 1974. [Online]. Available: http://doi.acm.org/10.1145/361179.361202

[8] S. Dolev, *Self-stabilization*. MIT press, 2000.

[9] S. Lohs, J. Nolte, G. Siegemund, and V. Turau, "Mission statement: Tolerancezone. a self-stabilizing middleware for wireless sensor networks," in *Technical Report SEEMOO-TR-2012-03*, GI/ITG. 11. Fachgesprch Sensornetze, TU Darmstadt, September 2012, pp. pp. 33–35.

Advanced Timestamping for pairwise Clock Drift Detection in Wireless Sensor/Actuator Networks

Marcel Baunach
Graz University of Technology, Austria – Institute for Technical Informatics
baunach@tugraz.at

***Abstract*—The temporal attribution of environmental events and measurements as well as the well-timed execution of corresponding reactions is of utmost importance for reactive sensor/actuator systems. We present a novel technique for the automatic creation of timestamps and reaction scheduling. Integrated either into an operating system kernel or implemented in hardware, we achieve the maximum precision for a system's given temporal resolution. Despite of the discretization of time in digital systems, we provide symmetric error intervals around 0 for both the timestamps and the reaction times. For example, this allows us to continuously determine the clock drift between pairs of communicating systems by mutually triggering periodic interrupts, but without exchanging any explicit information.**

I. Introduction

Wireless sensor/actuator networks (WSAN) are commonly deployed for observing and interacting with their environment. In this respect, temporal and spatial information are the two most fundamental measures for the "tagging" of states and events (i.e., state transitions) [1] [2]. This typically requires the precise knowledge of time and space to be associated with a node's self-captured and externally obtained values in order to properly correlate the contained information, and to trigger adequate reactions. Achieving a well synchronized coordination of these distributed systems is yet another challenge.

This paper outlines some problems regarding time in digital systems before it presents an advanced approach for taking precise timestamps, measuring and specifying temporal delays, and for scheduling and ensuring reaction times with a symmetric temporal error around 0. A real-world test bed shows how communicating systems can determine their relative clock drifts without any additional information exchange.

II. Time in Digital Systems

In contrast to e.g. device specific position and state values, time is a common property in distributed systems. It advances continuously and with a globally constant rate of change. Establishing a network-wide and consistent notion of time provides a natural base for their joint operation. Since digital systems are commonly driven by a clock generator C with frequency f_C and period $\lambda_C = \frac{1}{f_C}$, time and time intervals can easily and individually be measured – at least in theory: By counting the number of elapsed clock cycles since a well defined point in time, e.g. the system start, each captured event e, e.g. indicated by an interrupt, can be tagged with its current counter value c_e. Then, the event's absolute local system time is $\tilde{t}_e := c_e \cdot \lambda_C$, and the time difference between two events e_1, e_2 is

$$\tilde{\Delta}_{e_1,e_2} := \tilde{t}_{e_2} - \tilde{t}_{e_1} = (c_{e_2} - c_{e_1}) \cdot \lambda_C. \tag{1}$$

Obviously, both the time $\tilde{t}_e$ and the delay $\tilde{\Delta}_{e_1,e_2}$ already involve a concept-inherent imprecision caused by the discretization of the counter values $c_e \in \mathbb{N}$. We also assumed that λ_C is known and perfectly constant. Neither is true under real-world conditions! Apart, reactive systems often require the scheduling of a reaction r for a captured event e. Its intended execution time $t'_r \in \mathbb{R}$ is commonly related to an event timestamp $t'_e \in \mathbb{R}$ by a corresponding delay $\Delta'_{e,r} \in \mathbb{R}$:

$$t'_r = t'_e + \Delta'_{e,r} \tag{2}$$

However, even if the system triggers the response upon reaching the corresponding counter value $c_r \in \mathbb{N}$ and the corresponding system time $\tilde{t}_r$, the finally observable reaction delay still depends on the resolution λ_C of the system timer:

$$c_r = c_e + \left\lfloor \frac{\Delta'_{e,r}}{\lambda_C} \right\rfloor \qquad \tilde{t}_r = \tilde{t}_e + \left\lfloor \frac{\Delta'_{e,r}}{\lambda_C} \right\rfloor \cdot \lambda_C \tag{3}$$

Even though the rounding is quite obvious and potentially introduces far-reaching imprecision in real systems, it is commonly ignored. Moreover, for compositional task systems with dynamic execution flows, additional unpredictable errors are hidden in $\tilde{t}_r$. Even most embedded operating systems silently accept this problem, and developers have to compensate the imprecision with little control at task level [3].

A. The Discretization of Time

The simple *capturing of a discrete timestamp* t for an event is immediately affected by some inevitable rounding, and suffers from a measurement error $E_t \in I_1$ with $|I_1| = \lambda_C$. For the naïve and adverse reading of the timer counter, rounding down results in $I_1 := [0, \lambda_C)$, and induces a symmetry around the average measurement error $E_{t,\text{av}} = \frac{1}{2}\lambda_C$. Depending on the use of such timestamps, the emerging errors might accumulate during the system runtime. Similarly, the explicit *specification of delays* Δ'_t is also subject to rounding errors E_Δ. However, we can round half up manually when selecting a delay, and thus the corresponding error is $E_\Delta \in I_3 := [-\frac{1}{2}\lambda_C, +\frac{1}{2}\lambda_C)$. Though not avoidable entirely, I_3 is at least symmetric around 0. Based on these two fundamental error intervals I_1 and I_3, other intervals can be derived, and also exhibit an imprecision: For the *measurement of delays* Δ_E,

Table I
ERROR INTERVALS FOR DIFFERENT DISCRETIZATION TECHNIQUES (SYSTEM TIME RESOLUTION: λ_C)

		Naïve discretization		Our discretization approach	
problem / error type	derived from	error interval	symmetry	error interval	symmetry
P1: capturing of timestamps	fundamental	$I_1 = [0, \lambda_C)$	$\not\downarrow \frac{1}{2}\lambda_C \not\downarrow$	$I_3 = [-\frac{1}{2}\lambda_C, +\frac{1}{2}\lambda_C)$	0
P2: measurement of delays	$I_1 - I_1$	$I_2 = (-\lambda_C, +\lambda_C)$	0	$I_2 = (-\lambda_C, +\lambda_C)$	0
P3: specification of delays	fundamental	$I_3 = [-\frac{1}{2}\lambda_C, +\frac{1}{2}\lambda_C)$	0	$I_3 = [-\frac{1}{2}\lambda_C, +\frac{1}{2}\lambda_C)$	0
P4: scheduling of reaction times	$I_1 + I_3$	$I_4 = [-\frac{1}{2}\lambda_C, +\frac{3}{2}\lambda_C)$	$\not\downarrow \frac{1}{2}\lambda_C \not\downarrow$	$I_4 = [-\lambda_C, +\lambda_C)$	0

we see the implicit compensation of the asymmetry in I_1: $E_\Delta \in I_2 := I_1 - I_1 = (-\lambda_C, +\lambda_C)$. In contrast, the *scheduling of reaction times t* on external events inherits the asymmetry in I_1: $E_t \in I_4 := I_1 + I_3 = [-\frac{1}{2}\lambda_C, +\frac{3}{2}\lambda_C)$. System reactions will thus suffer from an average systematic lateness of $\frac{1}{2}\lambda_C$.

Table I summarizes the error intervals which must be expected for the naïve capturing of timestamps by simply reading the timer register after the corresponding event occurrence – e.g. within an interrupt service routine (ISRs). These ISRs commonly preempt regular application code and are thus perfectly suitable for capturing the timestamps. However, even the first instruction therein is not executed before some additional *interrupt latency* Δ_{IRQ}: If the timer value c_{TS} for the timestamp itself is copied after another implementation-specific delay Δ_{ISR} within the ISR, the discrete timestamp $\tilde{t}_e$ for the captured event e computes as

$$\tilde{t}_e = c_{\text{TS}} \cdot \lambda_C - (\Delta_{\text{IRQ}} + \Delta_{\text{ISR}}) = \tilde{t}_{\text{TS}} - \Delta_{\text{TS}}. \tag{4}$$

Hence, the reliable time tracking via (4) requires the correction value Δ_{TS} to be constant and free from rounding errors with respect to the discrete system time period. While this paper shows how this can be achieved and exploited, we refer to [4] for a discussion of further problems.

III. AN ADVANCED TIME DISCRETIZATION APPROACH

Our approach relies on a digital hardware timer component providing a local system timeline with a fixed temporal resolution. The timeline is managed by an OS kernel and accessible by any application software. The kernel automatically captures a timestamp $\tilde{t}_e$ for each interrupt e, and compensates the error's asymmetry about $\frac{1}{2}\lambda_C$ which would result from using the naïve approach as explained in Section II. Therefore, it provides standardized and architecture-specific interrupt service routines for introducing a constant and carefully dimensioned delay $\Delta_{\text{TS}} = \Delta_{\text{IRQ}} + \Delta_{\text{ISR}}$ before actually capturing the timer's counter value after the IRQ occurrence. According to (4) we then have to reduce the captured counter value by an adequate correction value Δ_{corr}: Selected properly, this correction finally results in the symmetry about 0 for $I_1 := [-\frac{1}{2}\lambda, \frac{1}{2}\lambda)$. While the timestamp measurement error E_{t_e} will still be equally distributed over I_1, this interval is shifted, and the average timestamp error is reduced from initially $\frac{1}{2}\lambda$ to 0. At the same time, the propagation (and amplification) of systematic errors for consecutive time-dependent reactions will become symmetric about 0, i.e. $I_4 = I_1 + I_3 = [-\lambda_C, +\lambda_C)$. Table I compares the error intervals of our compensation approach with the naïve technique.

Our concept is based on two synchronized clocks with interdependent frequency: We denote the CPU clock frequency as f and its period as λ. The system timer frequency is derived from the CPU clock by an even integer divider $\alpha \geq 2$: It is denoted as $f_C := \frac{f}{\alpha}$ and its period is $\lambda_C := \alpha \cdot \lambda$.

If an interrupt e occurs at time t'_e, the corresponding timer counter c_e will not be copied before some system inherent delay Δ_{TS} has passed. For our approach, we request this delay to take exactly Δ_c CPU cycles as follows:

$$\Delta_c := n \cdot \alpha + \frac{\alpha}{2} \quad \text{with} \quad n \in \mathbb{N}_0 \tag{5}$$

Thus, the delayed timestamp acquisition takes place at time

$$t'_{\text{TS}} = t'_e + \Delta_{\text{TS}} = t'_e + \Delta_c \cdot \frac{1}{f} = t'_e + \left(n \cdot \alpha + \frac{\alpha}{2}\right) \cdot \frac{1}{f}. \tag{6}$$

To compensate for this delay, and to force the timestamp error interval I_1 to become symmetric around the true event occurrence time while exhibiting an average error close to 0, we select the correction value as an integer multiple of λ_C:

$$\Delta_{\text{corr}} := (n \cdot \alpha) \cdot \frac{1}{f} = n \cdot \lambda_C \tag{7}$$

Thus, we simply have to subtract n from the copied timer value c_e to compute the timestamp $\tilde{t}_e$ for the interrupt e:

$$\tilde{t}_e = \left\lfloor \frac{t'_{\text{TS}}}{\lambda_C} \right\rfloor \cdot \lambda_C - \Delta_{\text{corr}} = c_e \cdot \lambda_C - n \cdot \lambda_C = (c_e - n) \cdot \lambda_C \tag{8}$$

The result's resolution implicitly equals the resolution of the system time.

A. An Implementation Example

As an example, we integrated our approach into the *Smart*OS operating system [5] for MSP430 MCUs [6]. While the main clock drives the CPU at f = 8 MHz, the divider $\alpha = 8$ derives the frequency $f_C = 1$ MHz for the system time with a resolution of 1 µs. According to (5), an adequate delay Δ_c between each interrupt occurrence and the acquisition of its timestamp can be adjusted through n:

$$\Delta_c := n \cdot \alpha + \frac{\alpha}{2} = n \cdot 8 + 4 \quad \text{with} \quad n \in \mathbb{N}_0 \tag{9}$$

Since the CPU inherently delays the acceptance of an interrupt by $\Delta_{\text{IRQ}} = 6$ CPU cycles, we already have to select $n \geq 1$. In fact, we did select $n = 1$ and thus have to wait for an additional number of $\Delta_{\text{ISR}} := \Delta_c - \Delta_{\text{IRQ}} = 6$ CPU cycles within the ISR ($1 \cdot 8 + 4 = 6 + 6$). According to the specification of the `mov` instruction, which is used for saving the timer value `TS`, it takes 4 CPU cycles until the value is read from the timer's special function register. The remaining two cycles are filled up by two `nop` instructions. For obtaining the absolute event timestamp $\tilde{t}_e$, n is simply subtracted from the event's absolute counter value c_e according to (7). With (8) the result can directly be interpreted as absolute system time, and is given in the timeline resolution of 1 µs:

$$\tilde{t}_e = (c_e - n) \cdot \lambda_C = (\texttt{TS} - n) \cdot \lambda_C \tag{10}$$

IV. Evaluation and Application Example

The test bed for demonstrating the benefit of our timestamping approach consists of pairs of nodes A, B playing some sort of Ping Pong game: By a wireless remote connection, node A triggers an IRQ signal e_0 which is received and timestamped ($\tilde{t}_0$) by the other node B. After some fixed delay Δ_{delay} the signal will be returned by node B. In turn, A catches, timestamps, and returns the signal after the same delay Δ_{delay}. Having received the last trigger e_n with local timestamp $\tilde{t}_n$ in an *ideal system*, the observed delay $\tilde{\Delta}_{\text{total},n}$ between each node's captured first and last signal timestamp should obviously equal the mathematically expected delay $\Delta'_{\text{total},n}$:

$$\tilde{\Delta}_{\text{total},n} := \tilde{t}_n - \tilde{t}_0 \stackrel{!}{=} 2n \cdot \Delta_{\text{delay}} =: \Delta'_{\text{total},n} \tag{11}$$

However, this equality is not given in *real systems*: Both devices suffer from their individual clock drift, and will *not* defer their responses by exactly the same delay Δ_{delay}. The right column of Table II shows significantly different drifts $d_{A,B}(t)$ for three node pairs as measured by an external observer.

Apart, waking up sufficiently early to emit the signal in time is not that easy since some load-dependent and variable system overhead must always be taken into account.

A. Signal Emission

For the precisely timed signal emission, we propose a dynamic self-calibration scheme based on self-observation: The trigger signal will not only be captured by the other node, where it is tagged with the timestamp $\tilde{t}_c$, but also by the emitting node itself. The local timestamp is called $\tilde{t}_r$. If the intended local response time for the current iteration has been computed as t_r^* before, the lateness can be computed afterwards and used as compensation value Δ_{comp} to adjust the delay for the next iteration at its emission time t_r^*:

$$\Delta_{\text{comp}} := \tilde{t}_r - t_r^* \tag{12}$$

$$t_r^* := \tilde{t}_c + \Delta_{\text{delay}} - \Delta_{\text{comp}} \tag{13}$$

In fact, the response time precision error ($E_{t_r^*} \in I_4$) depends not only on the two timestamps and their particular precision error ($E_{\tilde{t}_r}, E_{\tilde{t}_c} \in I_1$), but also on the error in the measured delay ($E_{\Delta_{\text{comp}}} \in I_2$) and the hard coded delay for the reply ($E_{\Delta_{\text{delay}}} \in I_3$) itself. Since we intentionally selected $\Delta_{\text{delay}} := m \cdot \lambda_C$ with $m \in \mathbb{N}$, at least this value is free from rounding errors and $I_3 := [0; 0)$ for this special application.

B. Pairwise Drift Calculation

For our tests we set up various node pairs A and B, and observed each nodes' $x \in \{A, B\}$ local timing error e_x which was autonomously calculated by each node after n iterations:

$$e_x \stackrel{(11)}{:=} \tilde{\Delta}_{\text{total},n} - \Delta'_{\text{total},n} = (\tilde{t}_n - \tilde{t}_0) - 2n \cdot \Delta_{\text{delay}} \tag{14}$$

Obviously, both timing errors e_A, e_B have different sign, unless the clocks are perfectly synchronous (i.e., $e_A = e_B = 0$). Additionally, we define the symmetry error e_{symm} as seen by an external observer as the average value over e_A, e_B. Since the average timestamp error $E_{t,\text{av}} \in I_1$ accumulates over the two acquired trigger timestamps within each iteration, we expect

$$e_{\text{symm}} := \frac{e_A + e_B}{2} = 2n \cdot E_{t,\text{av}}. \tag{15}$$

If we *indeed* achieve the timestamp error interval I_1 to be symmetric about 0, i.e. by selecting Δ_c properly according to (5), we can expect two observations for any pair of nodes A, B:

1) If both values e_A and e_B are made available to an external observer, their measured clock drift $d_{A,B}$, as given in the rightmost column of Table II, can be verified through

$$d'_{A,B} := e_A - e_B \stackrel{!}{=} d_{A,B} \qquad \text{with} \quad d_{A,B} = -d_{B,A}. \tag{16}$$

2) According to (15), $e_{\text{symm}} \stackrel{!}{=} 2n \cdot 0\mu s = 0\mu s$, and thus both values e_A and e_B will show the same absolute values. In direct consequence, each node can autonomously estimate its own drift towards the other node:

$$\tilde{d}_{A,B} = 2 \cdot e_A \quad \text{(for node } A\text{)} \tag{17}$$

$$\tilde{d}_{B,A} = 2 \cdot e_B \quad \text{(for node } B\text{)} \tag{18}$$

In particular, the exchange of any additional data, like e.g. timestamps, between the nodes is not necessary.

In contrast, if we intentionally violate (5) by using e.g. $\Delta_c := n \cdot \alpha$ instead, the average timestamp error interval would be symmetric around $E_t = \frac{1}{2}\lambda_C$. Consequently, $e_{\text{symm}} = 2n \cdot \frac{1}{2}\lambda_C$, and neither the autonomous drift computation according to (17) nor the external drift verification according to (16) would work any more.

C. Real-World Test Bed Analysis

Tables II and III show the test bed results for the three already mentioned node pairs and for two values of Δ_c after $n = 50$ iterations with $\Delta_{\text{delay}} = 1$ s ($\Delta'_{\text{total},n} = 100$ s).

When using $\Delta_c = 1 \cdot 8 + \frac{8}{2} = 12$, we indeed observed the expected symmetry error $e_{\text{symm}} \approx 0$ µs for all pairs. At least we received $|e_{\text{symm}}| < \lambda_C = 1$ µs, which is the timeline resolution, and thus the best timestamp precision a node can reach. Most important, as shown in Table II, the autonomously measured drifts between two nodes are almost perfect. Indeed, the

Table II
DRIFT CALCULATION FOR $\Delta_c = 12$

	locally obtained information[1,2]			observer[3]
$\tilde{d}_{A,B}$	node 10	node 11	node 72	$d'_{A,B}$
$\tilde{d}_{72,11}$	***1900.0***	1902.0	1900.0	1900
$\tilde{d}_{72,10}$	2818.0	***2818.0***	2816.0	2818
$\tilde{d}_{11,10}$	918.0	916.0	***916.0***	918

[1] regular font: autonom. measured according to (17)
[2] ***bold/italic***: derived according to (19)
[3] true drift as measured externally (oscilloscope)

Table III
DRIFT CALCULATION FOR $\Delta_c = 16$

	locally obtained information[1,2]			observer[3]
$\tilde{d}_{A,B}$	node 10	node 11	node 72	$d'_{A,B}$
$\tilde{d}_{72,11}$	***1884.0***	1836.0	2032.0	1900
$\tilde{d}_{72,10}$	2724.0	***2870.0***	2922.0	2818
$\tilde{d}_{11,10}$	840.0	1034.0	***890.0***	918

[1] regular font: autonom. measured according to (17)
[2] ***bold/italic***: derived according to (19)
[3] true drift as measured externally (oscilloscope)

maximum visible deviation is in range ± 2 µs. Another feature becomes apparent from this table: Since WLOG node A knows its drifts $\tilde{d}_{A,B}$ and $\tilde{d}_{A,C}$ towards the two other nodes B and C respectively, it can also derive the drift $\tilde{d}_{B,C}$ as

$$\tilde{d}_{B,C} := \tilde{d}_{A,C} - \tilde{d}_{A,B}. \tag{19}$$

For any other values of Δ_c violating (5), the nodes can not gain reliable information about their relative drift autonomously. Using e.g. $\Delta_c = 2 \cdot 8 = 16$, Table III summarizes the autonomously measured and computed drifts, and reveals quite large and asymmetric deviations towards the true drifts.

Besides the precision of the autonomous drift estimation, another interesting metric is the resulting trigger frequency. The theoretical value

$$f_{\text{trig}} := \left(2 \cdot \Delta_{\text{delay}}\right)^{-1} \tag{20}$$

will not be visible in reality since neither node uses a perfect clock. However, we would at least like to achieve

$$f_{\text{trig, av.}} = \left(\Delta'_{\text{delay},A} + \Delta'_{\text{delay},B}\right)^{-1}, \tag{21}$$

which is definitely the best compromise two nodes A, B can find if their true drift compared to the perfect global clock is unknown. Again, this can only be achieved if $e_{\text{symm}} = 0$. Thus, the larger $|e_{\text{symm}}|$ the larger is the deviation from the intended frequency $f_{\text{trig, av.}}$. These effects become once more visible in Tables II and III: For $\Delta_c = 12$ the values in each row of Table II are almost equal (i.e. consistent), while they exhibit significant variations for $\Delta_c = 16$ in Table III.

V. CONCLUSION AND OUTLOOK

In this paper we have proposed an approach for obtaining precise timestamps $\tilde{t}_e$ for external events e, and for ensuring the precisely timed execution of reactions r at scheduled times $\tilde{t}_r$. The error intervals for both $\tilde{t}_e$ and $\tilde{t}_r$ are symmetric about 0. While the first is achieved through the unified and carefully prepared processing of interrupts, the latter becomes possible through a simple self-calibration scheme at application layer. Both techniques proved to be a great benefit for an inherent problem within distributed (embedded) systems: As long as time is not properly manageable locally by the individual nodes, network-wide synchronization and event or state tagging will hardly achieve the potentially feasible precision.

Using our approach, a corresponding test bed showed the possibility to determine the drift between two nodes without the explicit exchange of any quantitative information like e.g. timestamps or previously measured delays. Instead, it is sufficient to periodically pass events between the nodes. Since suitable periodic behavior can also be found in several (wireless) communication protocols like [7], [8], [9], the proposed techniques can also be applied to support time synchronization and self-organization among the involved systems.

In fact, we already observed good time synchronization results when integrating our approach into the extendedDesync protocol from [10]. Apart, we have implemented our timestamping concept in hardware: Using a modified openMSP430 softcore [11], the specifically prepared interrupt controller is already able to pre-process and store timer values even for simultaneously occurring events.

REFERENCES

[1] G. Wittenburg, N. Dziengel, C. Wartenburger, and J. Schiller, "A System for Distributed Event Detection in Wireless Sensor Networks," in *9th ACM/IEEE International Conference on Information Processing in Sensor Networks (ISPN2010)*, 2010.

[2] K. Römer, "Discovery of Frequent Distributed Event Patterns in Sensor Networks," in *5th European Conference on Wireless Sensor Networks (EWSN 2008)*, Jan. 2008.

[3] M. Kuorilehto, T. Alho, M. Hännikäinen, and T. D. Hämäläinen, "SensorOS: A New Operating System for Time Critical WSN Applications." in *7th International Workshop on Embedded Computer Systems (SAMOS 2007)*, ser. LNCS. Springer, 2007.

[4] M. Baunach, "Handling Time and Reactivity for Synchronization and Clock Drift Calculation in Wireless Sensor/Actuator Networks," in *3rd International Conference on Sensor Networks (SENSORNETS 2014)*. SciTePress, Jan. 2014, pp. 63–72.

[5] M. Baunach, R. Kolla, and C. Mühlberger, "Introduction to a Small Modular Adept Real-Time Operating System," in *6. GI/ITG KuVS Fachgespräch Drahtlose Sensornetze*. RWTH Aachen, Jul. 2007.

[6] Texas Instruments Inc., *MSP430x161x Mixed Signal Microcontroller*, Dallas (USA), Aug. 2006.

[7] S. Støa and I. Balasingham, "Periodic-MAC: Improving MAC Protocols for Wireless Biomedical Sensor Networks through Implicit Synchronization," in *Biomedical Engineering, Trends in Electronics, Communications and Software*. InTech, Jan. 2011.

[8] K. Ito, N. Suzuki, S. Makido, and H. Hayashi, "Periodic Broadcast Type Timing Reservation MAC Protocol for Inter-Vehicle Communications," in *28th IEEE conference on Global telecomm. (GLOBECOM)*, 2009.

[9] A. Mutazono, M. Sugano, and M. Murata, "Frog Call-Inspired Self-Organizing Anti-Phase Synchronization for Wireless Sensor Networks," in *2nd International Workshop on Nonlinear Dynamics and Synchronization (INDS 2009)*, Jul. 2009.

[10] C. Mühlberger, "On the Pitfalls of Desynchronization in Multi-hop Topologies," in *2nd International Conference on Sensor Networks (SENSORNETS 2013)*. SciTePress, Feb. 2013, pp. 99–108.

[11] Oliver Girard, "The openMSP430 softcore," Web site http://opencores.org/project,openmsp430,overview, 2014.

Anwendungsmöglichkeiten softwarebasierter Selbstreparaturtechniken für Prozessoren in Sensornetzen

Mario Schölzel
Universität Potsdam und IHP Frankfurt (Oder)
schoelzel@ihp-microelectronics.com

Zusammenfassung— **Sinkende Strukturgrößen bei der Fertigung von CMOS-Schaltungen führen zu unzuverlässigerer Hardware. Neben einer erhöhten Anfälligkeit für transiente Fehler nimmt auch die Wahrscheinlichkeit für permanente Fehler zu, die dann zu einem Ausfall des Prozessors in einem Sensorknoten führen. Deshalb werden Techniken für selbstreparierende Prozessoren intensiv erforscht, um aus unzuverlässigen Hardwarekomponenten zuverlässige Prozessoren zu bauen. In diesem Beitrag werden einige dieser Techniken kurz vorgestellt und anschließend diskutiert, inwiefern diese geeignet sind die Zuverlässigkeit von drahtlosen Sensornetzen zu erhöhen. Entstehende Synergien werden dabei herausgestellt, sowie die Übertragbarkeit verschiedener Fehlertoleranztechniken, die in Prozessoren und Multiprozessorsysteme Anwendung finden, auf drahtlose Sensornetze diskutiert.**

I. EINLEITUNG

Batteriebetriebene Sensorknoten in drahtlosen Netzen werden auf Grund der beschränkten Energiereserven häufig mit sehr stromsparenden Microcontrollern, die nur einen beschränkten Leistungsumfang besitzen, bestückt. Bei einer zunehmenden Verbreitung von Sensornetzen werden aber auch die Anforderungen an die Funktionalität und damit an die Verarbeitungsleistung der Prozessoren in den Sensorknoten selbst wachsen. Durch geschicktes Energiemanagement auf Systemebene in Kombination mit Energy-Harvesting und verlustärmeren Fertigungstechnologien im Nanometer-Bereich wird es dann auch möglich sein, leistungsfähigere Prozessoren zu verwenden. Leistungsfähige und stromsparende Prozessoren in heutigen eingebetteten Systemen nutzen hauptsächlich die Parallelverarbeitung, um Taktfrequenzen und Versorgungsspannung trotz der hohen Verarbeitungsleistung niedrig zu halten. Die Parallelverarbeitung in solchen superskalaren Prozessoren nutzt redundant vorhandene Ressourcen im Prozessor, die andererseits auch wieder für die Umsetzung von Fehlertoleranztechniken verwendet werden können.

Insbesondere die immer kleineren Strukturgrößen bei der Fertigung von CMOS-Schaltungen führen zu einer erhöhten Anfälligkeit der gefertigten Baugruppen für temporäre und permanente Fehler [6] im Betrieb. Hinzu kommen statistische Variationen bei der Fertigung, beispielsweise bei der Dotierung im Kanalbereich der Transistoren oder bei den geometrischen Genauigkeiten der gefertigten Strukturen. Diese führen zu stark variierenden Schalteigenschaften der Transistoren, die aber durch erhöhte Toleranzen versteckt werden. Gleichzeitig können sich die Schalteigenschaften auch durch Alterungseffekte der Schaltung verändern, wodurch Prozessoren nach mehreren Jahren Benutzung ausfallen können. Um solche Prozessoren in langlebigen Systemen einzusetzen, kann es deshalb erforderlich sein sie fehlertolerant auszulegen. Hierzu bietet sich die in superskalaren Prozessoren inhärent vorhandene Redundanz an, die dann für Fehlertoleranzmaßnahmen verwendet werden kann. Dadurch können Wartungskosten für Sensornetze reduziert oder die Zuverlässigkeit für einen vorgegebenen Zeitraum erhöht werden. Dieser Beitrag stellt zunächst Fehlertoleranzmaßnahmen für Prozessoren vor. Anschließend wird gezeigt, dass die Verwendung selbstreparierender Prozessoren in Sensornetzen prinzipiell sinnvoll sein kann. Abschließend wird die Übertragbarkeit der Selbstreparaturorganisation von einem Multiprozessorsystem auf ein drahtloses Sensornetz betrachtet.

II. SELBSTREPARIERENDE PROZESSOREN

Passive Hardwareredundanz (z.B. Double/Triple Modular Redundancy) wird typischerweise verwendet, um transiente Fehler zu erkennen/maskieren. Sie kann aber auch verwendet werden, um permanente Fehler zu maskieren. Allerdings erfordern solche Techniken den zwei-/dreifachen Hardwaremehraufwand und bedingen auch einen entsprechend höheren Energieverbrauch des Prozessors, was bei batteriebetriebenen Sensorknoten kritisch sein kann. Soll mehr als ein permanenter Fehler mit solchen Techniken korrigiert werden können, dann steigt der Mehraufwand sogar noch stärker. Dieser Mehraufwand zur Behandlung permanenter Fehler in einem Prozessor kann deutlich reduziert werden, wenn aktive Hardwareredundanz verwendet wird, um die Zuverlässigkeit eines langlebigen Systems zu erhöhen. Permanente Fehler in einer Komponente des Prozessors werden dann durch eine Rekonfiguration, die die defekte Komponente außer Betrieb nimmt, umgangen. Zu diesem Zweck werden die durch permanente Fehler betroffenen Funktionalitäten einer Komponente nicht mehr genutzt. Die defekte Komponente kann durch Reservekomponenten ersetzt oder die Funktion von anderen bereits in Benutzung befindlichen Komponenten mit übernommen werden. Die Rekonfiguration kann dabei hardwarebasiert oder softwarebasiert erfolgen. Bei einer hardwarebasierten Rekonfiguration werden zusätzliche Schaltnetzwerke in den Prozessor integriert, um die auf einer defekten Komponente auszuführenden Operationen auf eine funktionierende Komponente umzuleiten [3, 4]. Softwarebasierte Verfahren sind gut für eingebettete Systeme mit einfachen statisch geplanten Prozessorarchitekturen geeignet und vermeiden die Nutzung einer defekten Komponente durch die Rekonfiguration der Software. Das bedeutet, dass der Binärcode der auf dem Prozessor ausgeführten Anwendungen so modifiziert wird, dass die defekte Komponente im Prozessor nicht mehr durch das Programm verwendet wird. In [2, 7, 10,

11] wurden entsprechende Techniken vorgestellt. Die Nutzung einer defekten Komponente kann beispielsweise durch folgende Maßnahmen vermieden werden:

- Durch eine globale Registerumbenennung kann die Verwendung defekter Register vermieden werden. Es müssen dann aber Reserveregister vorgehalten werden.
- 16/32-Bit Additionen können auf 8/16-Bit Additionen zurückgeführt werden, wenn Fehler im Addierer nur zu Fehlern in den oberen oder unteren 8/16-Bits des Ergebnisses führen.
- Die Verwendung eines Bypasses kann vermieden werden, wenn die datenabhängigen Operationen im Programm weit genug auseinander liegen.
- In statisch geplanten superskalaren Prozessoren (z.B. VLIW-Prozessoren) kann die Bindung von Operationen an bestimmte Ausführungseinheiten abgeändert werden, wenn diese mehrfach vorhanden sind, und damit die Verwendung defekter Ausführungseinheiten vermieden werden.

Die Anwendung dieser Techniken auf VLIW-Prozessoren hat gezeigt, dass bis zu 98% der Gatter im Prozessor gegen permanente Fehler geschützt werden können. Die Anpassung der Nutzeranwendung kann dabei durch ein Reparaturprogramm vorgenommen werden, dass sogar auf dem fehlerhaften Prozessor selbst ausgeführt wird [10]. Dieses Reparaturprogramm kann dabei sehr klein gehalten werden (< 300 Assemblerinstruktion) und die Anpassung einer Anwendung mit 64.000 Instruktion in weniger als einer Sekunde durchführen. Abbildung 1 zeigt, dass durch solche softwarebasierten Techniken damit deutliche Zuverlässigkeitssteigerungen für einen Prozessor erreicht werden können. Dabei ist λ die angenommene konstante Fehlerrate für die Transistoren des Prozessors. Bei einem nicht-fehlertolerant ausgelegten Prozessor sinkt die Zuverlässigkeit R_{NFT} bereits nach sehr kurzer Zeit unter 0.9, während bei einem fehlertoleranten Prozessor die Zuverlässigkeit R_{FT} zu demselben Zeitpunkt noch weit über 0.99 liegt. Die Herleitung der Zuverlässigkeitsfunktionen kann in [13] gefunden werden.

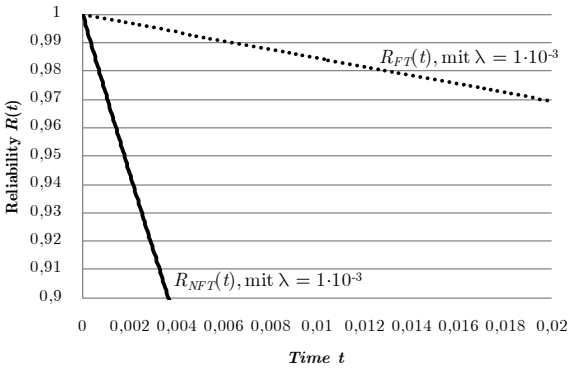

Abbildung 1: Plot der Zuverlässigkeit für einen fehlertolerenten VLIW-Prozessor und einen nicht-fehlertoleranten VLIW-Prozessor

Durch diese Möglichkeiten ist eine Erhöhung der Zuverlässigkeit um einen RIF (*Reliability Improvement Factor* [5]) von 14 bis 18 in dem dargestellten Zeitintervall möglich. Dabei ist der RIF definiert als

$$RIF = \frac{1 - R_{NFT}(t)}{1 - R_{FT}(t)},$$

wobei R_{NFT} die Zuverlässigkeit des Originalsystems und R_{FT} die Zuverlässigkeit des fehlertoleranten Systems ist.

III. Nutzen in Sensornetzen

Betrachtet werden jetzt permanente Fehler in einem Sensornetz, die durch den permanenten Ausfall eines oder mehrerer Sensorknoten entstehen. Wird in einem homogenen Sensornetz jeder Sensorknoten *a* durch einen entsprechend zuverlässigeren Sensorknoten *b* mit der um den RIF *x* erhöhten Zuverlässigkeit ersetzt, dann kann insgesamt die Zuverlässigkeit des Sensornetzes um den RIF *x* erhöht werden, weil es unwahrscheinlicher wird, dass die Knoten ausfallen. Typischerweise wird in Sensornetzen aber ohnehin Redundanz in Form zusätzlicher Sensorknoten angewendet. Es stellt sich damit die Frage, ob die Zuverlässigkeit eines Sensornetzes besser durch zusätzliche aber nicht zuverlässigere Sensorknoten oder nur durch zuverlässigere Sensorknoten erhöht werden soll. In [12] wurden entsprechende Untersuchungen für einzelne Routerknoten in einem Sensornetz durchgeführt. Dabei zeigte sich, dass die Zuverlässigkeit wesentlich stärker durch zusätzliche Knoten erhöht werden konnte als durch zuverlässigere Routerknoten. Allerdings wurden dabei Sensornetze betrachtet, in denen diese Routerknoten einen *single point of failure* (SPOF) darstellten. Ein entsprechendes Netzwerk ist in Abbildung 2 durch die fett gedruckten Knoten und Kanten gezeigt. Der Knoten *s* stellt die Senke in diesem Sensornetz dar.

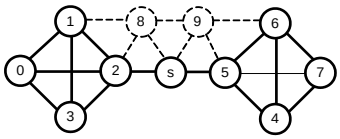

Abbildung 2: Clusterkonfiguration mit SPOF bei Betrachtung der fett gedruckten Knoten und ohne SPOF bei Hinzunahme der Knoten 8 und 9.

Wird in diesem Netz Redundanz durch die zusätzlichen Knoten 8 und 9 eingefügt, so kann die Zuverlässigkeit deutlich erhöht werden. Die entsprechenden Zuverlässigkeitsfunktionen sind in Abbildung 3 dargestellt (*RNFT* und *RFT_Net*).

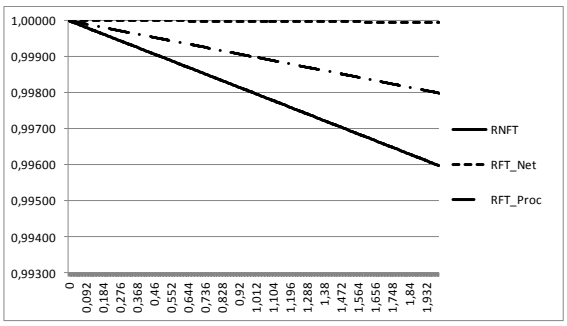

Abbildung 3: Zuverlässigkeitsfunktionen zu Abbildung 2.

In dem dargestellten Zeitraum wird eine Erhöhung der Zuverlässigkeit um den RIF 80 (bei $t = 2$) bis 4000 (bei $t \to 0$) erreicht. Das Ersetzen der fett gedruckten Knoten durch zuverlässigere Knoten (mit RIF 2) erhöht die Zuverlässigkeit des Netzwerkes dagegen nur um den RIF 2 (Kurve *RFT_Proc*). Somit ist in diesem Fall das Einfügen zusätzlicher Knoten zu bevorzugen. Diese Schlussfolgerung wurde auch bereits in [12] gezogen.

Werden die gleichen Betrachtungen dagegen für ein Sensornetz durchgeführt, das keinen SPOF bei der Kommunikation mit der Senke enthält (vgl. fett gedruckte Knoten und Kanten in Abbildung 4), dann erhöht das Einfügen zusätzlicher Knoten

(in diesem Fall Knoten 8 und 9) kaum noch die Zuverlässigkeit.

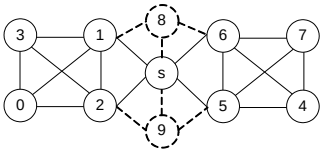

Abbildung 4

Zu sehen ist das in Abbildung 5, in der sich die Zuverlässigkeitskurven *RNFT* und *RFT_Net* überlagern. Die gleiche Feststellung wurde bereits in [1] getroffen.

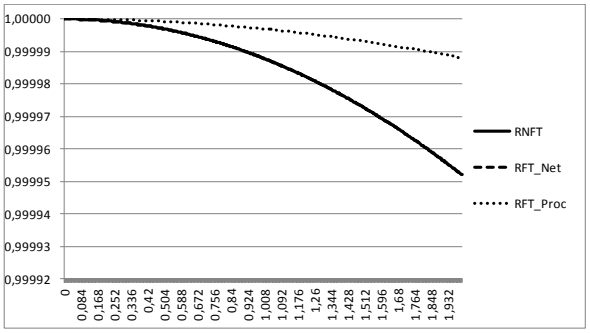

Abbildung 5

Dagegen erhöht in diesem Fall das Ersetzen der Knoten durch zuverlässigere Knoten die Gesamtzuverlässigkeit (vgl. Kurve *RFT_Proc* in Abbildung 5). Welche Form der Redundanz die Zuverlässigkeit stärker erhöht, hängt also stark von der bereits vorhandenen Redundanz im Sensornetz ab. Ist bereits ausreichend Redundanz im Sensornetz für die Kommunikation vorhanden, so kann die Gesamtzuverlässigkeit besser über die Erhöhung der Zuverlässigkeit der einzelnen Knoten gesteigert werden.

IV. Anwendung in Energieeffizienten Sensorknoten

Dieser Abschnitt beschreibt eine mögliche Anwendung der Selbstreparatur für energieeffiziente Sensorknoten. Es gibt permanente Alterungsfehler (NBTI, HCI), die sich als Verzögerungsfehler bemerkbar machen. Das bedeutet, dass es im Laufe der Zeit zu erhöhten Signallaufzeiten auf einigen Logikpfaden in einem Prozessor kommt. Besonders stark machen sich solche Effekte in Kombination mit Energiesparmaßnahmen wie der dynamischen Anpassung der Versorgungsspannung (DVS) bemerkbar. Wird in dem Prozessor DVS verwendet, wie es beispielsweise in [8] beschrieben ist, dann treten diese Verzögerungsfehler zuerst bei sehr niedrigen Versorgungsspannungen auf und führen dann zu einem funktionalen Fehlverhalten. In der Konsequenz ist der Sensorknoten defekt oder kann seine Funktion nur unter einem erhöhten Energiebedarf aufrechterhalten, was jedoch die Batterie stark belastet. Aus diesem Grund, können die Komponenten, in denen ein entsprechend langsamer Pfad im Lauf der Zeit entsteht, mit den Selbstreparaturmaßnahmen, die in Abschnitt II beschreiben wurden, außer Betrieb genommen werden. Das hat im Allgemeinen jedoch zur Folge, dass auch die Verarbeitungsleistung sinkt, weil durch die softwarebasierte Selbstreparatur der Ablaufplan des Programms verlängert wird. Das führt in der Konsequenz zu einer erhöhten Laufzeit im Vergleich zur ursprünglichen Programmversion und damit auch zu einem erhöhten Energieverbrauch, weil der Prozessor für eine längere Zeit aktiv ist. Allerdings wurde in [11] gezeigt, dass im Falle einer defekten Komponente sich die Laufzeit nur um ca. 7% bis 30% erhöht.

V. Organisation der Selbstreparatur

Die beschriebenen Selbstreparaturtechniken zur Behandlung permanenter Fehler erfordern in jedem Sensorknoten eine Organisation der Selbstreparatur. Dazu gehört auch die Ausführung eines vorhergehenden diagnostischen Selbsttest, um defekte Komponenten in den Sensorknoten zu lokalisieren. Durch diesen Selbsttest können in einem multi-hop Netzwerk auch Knoten erkannt werden, die die weitergeleiteten Datenpakete auf Grund interner Fehler korrumpieren.

Eine mögliche Form der Organisation der softwarebasierten Selbstreparatur in Systemen mit mehreren vernetzten Prozessoren wurde bereits in [9] beschrieben. Das in dieser Arbeit betrachtete System ist ein Multiprozessorsystem mit verteiltem Speicher. Die Organisation dieses Multiprozessorsystems ist dabei sehr ähnlich der Organisation eines Sensornetzwerkes und ist schematisch in Abbildung 6 dargestellt. Für die Kommunikation stehen jedoch drahtgebundene Verbindungen zur Verfügung. Jeder Knoten verfügt über einen lokalen Speicher, einen lokalen Selbsttest und eine lokale Selbstreparaturfunktion. Außerdem stellt das Netzwerk für jeden Knoten die Möglichkeit bereit, auf den Speicher der anderen Knoten zuzugreifen.

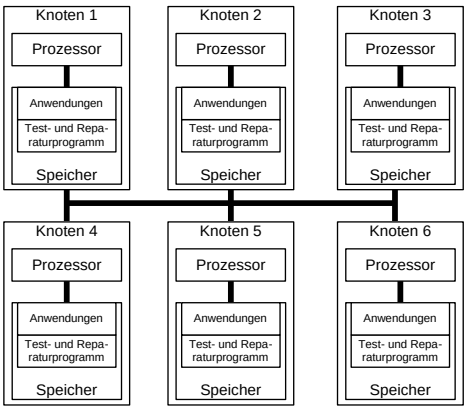

Abbildung 6: Architektur Multiprozessorsystem aus [9].

Dadurch wird es möglich, nicht nur eine Selbstreparatur lokal in jedem Knoten auszuführen, sondern auch eine Fremdreparatur der anderen Knoten zu ermöglichen. Der Selbsttest und die Selbstreparatur in einem solchen System ist nun wie folgt organisiert: Bei der Inbetriebnahme oder in regelmäßigen Abständen wird ein diagnostischer Selbsttest lokal in jedem Knoten gestartet. Findet der diagnostische Selbsttest Fehler, so kann die lokale Selbstreparatur versuchen, eine Rekonfiguration des Prozessors vorzunehmen. Ist dies nicht möglich, beispielsweise, weil der Selbstreparaturalgorithmus nicht fehlerfrei auf dem fehlerhaften Prozessor *F* ausgeführt werden kann, so kann eine Fremdreparatur veranlasst werden. Dazu ist der Fehlerzustand an einen funktionierenden Prozessor *K* zu übertragen, der die Anwendung im Speicher von Prozessor *F* an die aktuelle Fehlersituation anpasst und anschließend die aktualisierte Anwendung in den Programmspeicher des fehlerhaften Prozessors *F* zurückschreibt. In einem drahtgebundenen Multiprozessornetzwerk ist dies relativ einfach möglich, weil ein Zugriff auf die lokalen Speicher durch das Verbindungsnetzwerk möglich ist.

In einem drahtlosen Sensornetz kann die Fremdreparatur durch andere Sensorknoten oder durch ein über die Senke erreichbares System durchgeführt werden. Dafür muss aber zumindest eine zuverlässige Minimalkommunikation mit dem defekten Sensorknoten ermöglicht werden. Das bedeutet, dass die Teile des Protokollstacks, die von dem Prozessor in dem Sensorknoten implementiert werden, auch dann funktionieren müssen, wenn der Prozessor oder sein Speicher fehlerhafte Komponenten enthält. Diese kann beispielsweise durch mehrere unterschiedliche Implementierungsvarianten des Protokollstacks ermöglicht werden, die verschiedene Ressourcen im Prozessor und Speicher nutzen. Abhängig von dem aktuellen Fehler, wird eine Version ausgewählt, die die fehlerhafte Komponente nicht verwendet. Um die Wahrscheinlichkeit zu verringern, dass keine dieser Programmversionen ausführbar ist, kann jede dieser Versionen so gestaltet sein, dass nur ein minimaler Befehlsumfang des Prozessors verwendet wird. Hier soll in Zukunft untersucht werden, wie eine derartige Service-Schicht in einem Sensorknoten implementiert werden kann, so dass möglichst viele Fehler in dem Prozessor toleriert werden können und trotzdem noch eine Kommunikation möglich ist. Abbildung 7 zeigt eine mögliche Systemarchitektur für Sensorknoten die eine entsprechende Organisation der Selbstreparatur unterstützt.

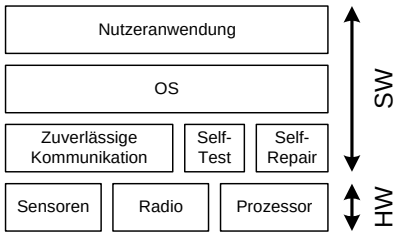

Abbildung 7: Vorgeschlagene Systemarchitektur

In dieser Architektur gibt es unmittelbar über der Hardwareebene eine Service-Ebene in Software, die einen lokalen Selbsttest und eine lokale Selbstreparatur des Knotens ermöglicht. Außerdem ist in dieser Service-Ebene die Implementierung einer Minimalkommunikation vorgesehen, die auch einen Zugriff auf den Speicher des Sensorknotens zulässt. Dadurch kann bei Erkennung eines Fehlers, der Fehlerzustand an andere Knoten oder die Senke übermittelt werden. Dort kann eine Anpassung der Nutzeranwendungen und des OS durchgeführt werden. Beispielsweise können Fehler im Speicher des Sensorknoten durch eine Relokation des Programms und/oder der Daten umgangen werden. Da in einem homogenen Sensornetzwerk die Software in allen Sensorknoten identisch ist, können dadurch sogar Fehler in einem Sensorknoten behandelt werden, die die dortige Nutzeranwendung bereits korrumpiert haben. Die angepasste Softwarekomponenten werden dann wieder auf den fehlerhaften Sensorknoten zurückkopiert. Für heterogene Sensornetze ist auch eine spezielle Form von Serviceknoten denkbar (beispielsweise implementiert in den Routerknoten), die speziell für die Rekonfiguration der umliegenden Sensorknoten vorgesehen sind. Diese Serviceknoten können dann auch verwendet werden, um den begrenzten lokalen Speicher in einem Sensorknoten zu entlasten, indem jeder Sensorknoten nur über einen sehr kleinen rudimentären Selbsttest verfügt. Stellt dieser Selbsttest einen Fehler fest, dann kann der Serviceknoten einen diagnostischen Selbsttest auf den Sensorknoten übertragen, der den genauen Fehler lokalisiert und diesen Fehlerzustand an den Serviceknoten übermittelt. Der Serviceknoten, der eine Kopie der Software des Sensorknotens enthält, kann dann ein an die aktuelle Fehlersituation angepasstes Programm erstellen und dieses dann über die Serviceschnittstelle an den Sensorknoten übertragen.

VI. Zusammenfassung

Dieser Beitrag hat Möglichkeiten der softwarebasierter Selbstreparatur in Prozessoren beschrieben und gezeigt, dass unter bestimmten Umständen solche Maßnahmen in Sensorknoten eingesetzt werden können, um die Zuverlässigkeit des gesamten Sensornetzes zu erhöhen, was allerdings die Verwendung statisch geplanter Prozessoren mit redundanten Komponenten erfordert. Die Organisation dieser Techniken in einem Multiprozessornetzwerk wurde kurz beschrieben und die Übertragbarkeit auf Sensornetze diskutiert. Dafür ist jedoch die Umsetzung einer zuverlässigen Kommunikationsschicht in den Sensorknoten erforderlich. Diese ist, zusammen mit einer Analyse des zusätzlichen Energiebedarfs für die Rekonfiguration, Gegenstand weiterer Untersuchungen.

VII. References

[1] G. Egeland and P. Engelstad: *The Availability and Reliability of Wireless Multi-hop Networks with Stochastic Link Failures*. Journal on selected areas in communications, 27(7), pp. 1132-1146. 2009.

[2] L. Guerra, M. Potkonjak and J. M. Rabaey: *High Level Synthesis Techniques for Efficient Built-In-Self-Repair*. IEEE Workshop on DFT in VLSI systems, pp. 41-48, 1993.

[3] T. Koal, D. Scheit and H. T. Vierhaus: *A Concept for Logic Self Repair*. Proc. of the 12th Euromicro Conference on Digital System Design / Architectures, Methods and Tools (DSD'09), pp. 621-624, 2009.

[4] T. Koal and H. T. Vierhaus: *A software-based self-test and hardware reconfiguration solution for VLIW processors*. Proc. of the 13th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS'10), pp. 40-43, 2010.

[5] P. K. Lala: *Self-Checking and Fault Tolerant Digital Design*. Morgan Kaufmann, 2000.

[6] Y. Li, Y. M. Kim, E. Mintarno et. al.: *Overcoming Early-Life Failure and Aging for Robust Systems*. IEEE Design & Test of Computers, 26(6), pp. 28-39, 2009.

[7] A. Meixner and D. J. Sorin: *Detouring: Translating Software to Circumvent Hard Faults in Simple Cores*. Proc.of the International Conference on Dependable Systems and Networks (DSN), pp. 80-89, 2008.

[8] R. Min, T. Furrer and A. Chandrakasan: *Dynamic Voltage Scaling Techniques for Distributed Microsensor Networks*. Proceedings of the IEEE Computer Society Workshop on VLSI, pp. 43-46, 2000.

[9] S. Müller, M. Schölzel and H. T. Vierhaus: *Towards a Graceful Degradable Multicore-System by Hierarchical Handling of Hard Errors*. Proc. of the 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP'13), 2013.

[10] M. Schölzel: *Software-Based Self-Repair of Statically Scheduled Superscalar Data Paths*. Proc. of the 13th IEEE International Symposium on Design & Diagnostics of Electronic Circuits & Systems (DDECS'10), pp. 66-71, 2010.

[11] M. Schölzel: *Fine-Grained Software-Based Self-Repair of VLIW Processors*. 26th IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems, pp. 41-49, 2011.

[12] I. Silva, L. A. Guedes, P. Portugal and F. Vasques: *Reliability and Availability Evaluation of Wireless Sensor Networks for Industrial Applications*. Sensors, 12(1), pp. 806-838. 2012.

[13] M. Schölzel: Self-Testing and Self-Repairing Processors: Techniques for Statically Scheduled Superscalar Processors. Habilitation Thesis (BTU Cottbus-Senftenberg), 2014.

From Energy Accounting to Energy Management

André Sieber, Reinhardt Karnapke, Jörg Nolte
Distributed Systems/Operating Systems Group, BTU Cottbus-Senftenberg
Cottbus, Germany
{as, karnapke, jon}@informatik.tu-cottbus.de

***Abstract*—Embedded systems, e.g. nodes within sensor networks, often have tight bound goals for lifetime while running from a not renewable energy source. Mostly batteries are used, which are vulnerable to temperature and non-linear effects. Additionally, variations within the hardware or induced by the software make the prediction of the available and consumed energy a complicated task. To reach certain lifetime goals under these influences, online energy management is necessary. For a fine-grained management on the level of individual sub-tasks, it is necessary to know where in the system the energy is consumed.**

In this work, we extend our online energy accounting approach [1] to enable online energy management. We present ways to control application and device behavior, and, thus, energy using energy budgets. First experiments yield promising results, reaching their lifetime goals while maintaining a high application quality.

I. Introduction

Sensor nodes should run for years with a limited energy supply. To prevent early node failures due to depleted batteries, it is necessary to use energy management. While traditional energy management schemes focus on saving energy and thus extending the lifetime of sensor nodes, there are scenarios in which nodes should not reach the highest possible lifetime, but reach a certain predefined lifetime goal, e.g. a service interval. When the network is deployed in harsh and hard to reach environments, it is essential that no nodes fail early. Additionally, the available energy should be used completely to have a high application quality. To fulfill these requirements, a management scheme needs to control the energy consumption. Controlling the energy consumption to reach the lifetime goal is more important than a constant application quality [2].

Online energy accounting is an instrument for the power management on the nodes to monitor the consumption and enforce its constraints and policies. Fig. 1 shows an approach where the consumed energy as well as information from the battery is used to control the application behavior within user-defined boundaries, to reach the predefined lifetime goal. Taking the battery state of charge into account makes it possible to react to the batteries non-linear behavior, e.g., rate capacity, recovery effects and their strong dependence on the temperature [3], as well as to react on inaccuracies of the consumption model. When the boundaries set by the user can no longer be satisfied, the system must at be capable of detecting and informing the networks maintainer as early as possible to increase the reaction window.

The energy that is consumed by a sensor node depends on the active devices and the energy source. The activation

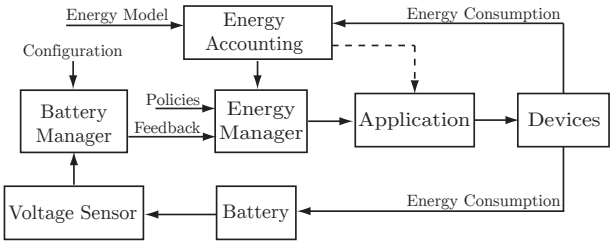

Fig. 1. Power management concept taking the current consumption and the battery state of charge into account

of devices is most often under control of the node software, along with the time they are active. Statements about the consumption of a device can be taken from the manufacturers datasheet, but variations of the analog and digital components, as well as manufacturing faults, can occur. While this may not render the nodes useless, it could change their energy footprint.

The energy source can be connected directly or use a voltage regulator. While connecting the source directly exposes components to the degrading voltage level and thus to voltage dependencies in the power consumption [4], voltage regulators deliver a constant voltage but suffer from a load and input voltage dependent efficiency. Both must be considered and make it necessary to implement a dynamic accounting which is capable of changing the underlying consumption values depending on the voltage level or converter efficiency.

With reliable information about the remaining energy and the consumption of the system, the energy manager can use various handles to influence the energy consumption. The application duty cycle could be changed or the mac/routing timings adapted. To enable the management to limit the consumption and isolate tasks and devices, we introduce energy budgets. These budgets provide a certain amount of energy and enable the energy management to plan how to spend the energy.

The rest of this paper is structured as follows: The problem statement can be found in section II. In section III, our energy management approach is presented. The battery observation is described in section IV. Section V presents the energy accounting, while section VI focusses on the energy budgets. Possible management policies based on budgets are outlined in section VII. In section VIII related work is presented. Finally, a conclusion is given in section IX.

II. Problem Statement

To reach a defined lifetime goal G with an limited energy reserve E, the consumption must be limited and rationed. To do so, information about the available energy E_{avail} is necessary, which depends on the initial energy E and the consumed energy $E_{consumed}$. $E_{consumed}$ can be obtained using either hard or software solutions and can depend on the voltage level or the converter efficiency. While in the the ideal case $E_{avail} = E - E_{consumed}$, batteries depend on various nonlinear factors which influence their capacity. These factors must be taken into account, by including the state of charge of the battery into the estimation of E_{avail}.

With an estimation of E_{avail} and the remaining runtime, it is possible to adjust the applications consumption to reach G. This can involve numerous devices and application parts with different preferences which all must be balanced. To guarantee a certain share of E_{avail} to each of them, it is necessary to isolate them energy wise from each other. A drawback of the isolation is that energy shares may not be spent completely, or demands may remain unsatisfied. The reason for this is that the actual energy requirements are dynamic and can not fully be calculated in advance.

Not all applications share the same detailed interest in energy awareness. There are several levels of energy awareness and energy wise cooperation of an application, which all must be addressable for the energy management.

- *Energy ignorant* applications do not consider energy in any way. To enforce lifetime goals, these applications may be interrupted if their energy budget is exhausted or energy is wasted, e.g. due to a low duty cycle.
- *Basic energy aware* applications control their activity based on information of remaining energy and time, e.g. by adapting the duty cycle.
- *Task energy aware* applications know the consumption of their different tasks and devices and may shift energy between them to optimize the application quality. Additionally, this enables the accounting of requests within a sensor network [5].

III. General Approach

Figure 2 shows our implementation of the dynamic energy management presented in figure 1. Central to the approach is the application, since it must be able to adapt to changing energy availabilities. This can be archived e.g. by changing duty cycles, sensor resolutions or mac/routing parameters. But to be able to adapt, the application must be aware of the available energy. The management is responsible for providing this information. It consists of three parts:

1) The observation of the available energy, including the battery state of charge and the consumed energy.
2) The online energy accounting, which provides information about the consumed energy, down to the level of individual device states.
3) The energy management, which uses so called energy budgets to provide the application with a distinct amount of energy.

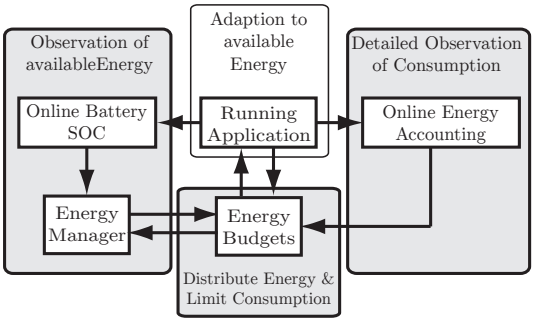

Fig. 2. Dynamic approach to adapt the energy consumption to varying energy supply and hard-/software variations using energy budgets.

IV. Battery Observation

As hardware based approaches increase unit costs and consume additional energy, our approach is software based and uses only the analog digital converter to read the battery voltage [6]. To monitor the battery discharge, it combines simple linearization with assumptions of the general discharge behavior. If the voltage curve is outside the assumed boundaries, the system gives a feedback to the energy management, to allow an adjustment of the assumed remaining energy.

V. Energy Accounting

To be able to manage the available energy, it is essential to know the flow of energy, down to individual device states. Our approach is similar to [5], [7], [8], [9], by tracking the active time of device states, but also takes variances in the consumption induced by voltage changes and converters into account. To do so, we adopted Quanto's *energy sinks* and *power states* [10] view. For our approach, a sink is a potentially independent unit that consumes energy, a power state defines how much energy is consumed by a sink.

The consumption of a sink can be modified dynamically by changing its power state. This makes it possible to use more than one power state for a single sink and thus reduce the memory footprint, if no detailed information about the distribution of the consumed energy is required.

Dynamic effects that influence the energy consumption are another reason to modify the power states of a sink dynamically. The changing supply voltage when the system is connected directly to a battery exposes components to its voltage curve. Our system monitors the voltage and informs all involved device drivers when the voltage changes substantially, so that they can adapt their power states. Additionally, the accounted amount must be altered when a voltage regulator is present. All accounted energy must be modified by a dynamically recomputed efficiency factor, to compensate the regulators voltage and load dependency.

Since we implemented our approach for the event-driven operating system REFLEX [11], which is implemented in C++, the integration of the accounting mechanisms into the existing device drivers is eased. Each driver that should be integrated is derived from a base class and gains all the functions and variables necessary for the accounting. If applicable, it

is additionally registered by the mechanisms for voltage or converter efficiency.

We integrated the accounting into the drivers for the Texas Instruments eZ430-Chronos [12]. As we used a 32kHz timer, the timestamp resolution is 1/32ms, to avoid additional energy consumption. To cover the entire lifetime of a sensor node, the timestamps and energy storages are based on 64-bit values. The power states are based on a statistical model built through measurement of 90 nodes [13].

VI. Towards Management using Energy Budgets

With detailed information about the energy consumption of the system, it is possible to enforce limitations to meet desired discharge rates. Since applications should react to reach lifetime goals, they must be provided with information about the available energy. With different application parts and goals competing for the energy, a mechanism is necessary to separate them and provide a local view of the remaining energy for a single part. However, energy alone has no value without a time window. This information can be provided using energy budgets, which are based on the concept of resource containers [14], but are only used to limit the resource energy.

An energy budget is an abstract reservoir for energy. It represents the right to a certain amount of energy. The system energy (or parts of it) is/are divided between the budgets. An abstract budget B is defined by its currently stored amount b and the validity interval $[t_{start}, t_{end}]$ of b. The demand of a budget is defined by the minimal energy min needed and the maximal energy max consumable by its associated consumers during $[t_{start}, t_{end}]$. They give the system a hint on reasonable values for filling B at t_{start}:

$$B = (b, [t_{start}, t_{end}], min, max)$$

Since most activities within a sensor node are not short lived, the validity interval is more or less a constant refresh interval, at whose begin the budget will be refueled. This division of G into T intervals, each with the same length of $[t_{start}, t_{end}]$, also devides to system consumption and reduces the prediction horizon for the application. A combined interval where the battery observation computes the remaining energy and all budgets are refueled simplifies the process and reduces the overhead. The essential requirement is that the minimal demand for all budgets is satisfied in every interval. If this demand can not be satisfied, the networks maintainer must be informed, since the functionality can no longer be guaranteed. The remaining energy can be distributed among the budgets in various ways (see section VII).

Obtaining reasonable values for min and max is a hard task, since they depend not only on the consumption of associated devices but also on the interval duration $[t_{start}, t_{end}]$. Apart from careful calculation, these values could be obtained by simulation or experiments. Another option is to include a learning phase, but this is only feasible for budgets with associated activities with a nearly constant demand.

Each energy sink must be bound to an energy budget. While an energy sink captures the consumed energy, the budgets are charged for the energy. This connection is also reflected in the implementation, where the consumed energy is stored in the energy sink and removed from the bound energy budget.

To enforce the limitations induced by budgets, the device drivers should only work if their budgets are not empty. An extension is to track the typical request/activation cost for each device and check if enough energy is in the buffer before granting a request.

VII. Possible Budget Policies

The basic concept of energy budgets and their behavior can be extended and policies implemented in numerous ways.

Energy share distribution can be based on a fair share of all budgets, based on priorities or dependent from max. Economic approaches like [15] are also possible.

Insufficient system energy can be handled by a simple error signal to the application and the networks user/maintainer or lead to an emergency mode where the remaining energy is distributed among prioritized budgets.

While always bound to an energy sink, budgets can be attached to various **logical entities**. For example, a budget can be permanently attached to one or more energy sinks, to enable device based management. Another possibility is to attach them to requests, to share devices and enable per-use charging. This could be extended to the cpu and would lead to task/process based management. Additionally, budgets could be attached to the system events and passed along the data-/controlflow.

Empty budgets in general could be treated through isolation, which means the associated entities could no longer work, or by cooperation, which means that the empty budgets try to obtain energy from the management or directly from other budgets. This energy can also come as a debt, which must be repaid from the budgets future share.

Unused energy can be saved for future use, slowly drained by the system, granted to budgets in need or lead to an adaption of the share itself.

Which policy or combination should be used depends mainly on the requirements of the application. Early evaluation shows that with different policies, the behavior of the system varies strongly in different situations.

VIII. Related Work

There are numerous approaches for tracking consumed energy, both in hard- and software.

Hardware approaches cover coloumb counters and smart battery systems [16], Sensor Node Management Devices [17] and especially designed measurement devices [16], [18], all with different benefits and drawbacks. But all hardware approaches introduce an overhead in device costs and energy consumption.

Software approaches rely on the observation of certain events to account for the consumption. To be able to observe these events, the code has to be modified with hooks to call the accounting functions. The events can either be based on functional application blocks [19], [20] or based on device

driver actions [5], [7], [8], [9]. The latter are based on taking timestamps when devices change their state to obtain the duration of each state, to calculate the consumed energy.

Based on the concept of resource containers (RC) [14], energy capsules [21] and energy containers [22] provide information about the energy usage of (sub-)tasks. Another form of RCs are the reserves of cinder [23], where the flow between reserves is controlled by so called taps. Ecosystem [24] with its currency model [25] uses RCs to limit the energy applications and tasks can spend on computation and I/O.

In EPOS [26], tasks are divided into a mandatory and an optional part, which could be omitted to reach the lifetime goal. Energy levels [27] follow a similar concept, controlling the consumption by switching between different application (sub-) levels with different utilities and energy footprints.

In [28] an approach which adapts duty cycles based on the temperature dependent consumption is presented. The goal of energy wise isolation of applications is implemented in [29]. In [30] energy budgets are introduced as node wide representation to limit the consumption to meet the energy storage in harvesting systems.

IX. Conclusion and Future Work

In this paper, we presented an energy management concept based on energy budgets. The toolset provided by these budgets is promising in enabling a wide range of applications to reach their lifetime goals in the presence of dynamic changes in the environment.

In the future we want to further implement and evaluate the possibilities enabled through different policies for isolation and cooperation of budgets. Additionally, the whole approach will be evaluated in the field with real applications.

References

[1] André Sieber and Jörg Nolte. Online device-level energy accounting for wireless sensor nodes. In *Proceedings of the 10th European conference on Wireless Sensor Networks*, pages 149–164, 2013.

[2] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: An acquisitional query processing system for sensor networks. *ACM Transactions on Database Systems (TODS)*, 30(1):122–173, 2005.

[3] T. Reddy. *Linden's Handbook of Batteries, 4th Edition*. Mcgraw-hill, 2010.

[4] Chulsung Park, Kanishka Lahiri, and Anand Raghunathan. Battery discharge characteristics of wireless sensor nodes: An experimental analysis. *power*, 20:21, 2005.

[5] Simon Kellner and Frank Bellosa. Energy accounting support in tinyos. *PIK-Praxis der Informationsverarbeitung und Kommunikation*, 32(2):105–109, 2009.

[6] André Sieber and Jörg Nolte. Utilizing voltage decline for reaching lifetime goals. Technical report, Paderborn, Germany, 2011.

[7] Adam Dunkels, Fredrik Osterlind, Nicolas Tsiftes, and Zhitao He. Software-based on-line energy estimation for sensor nodes. In *Proceedings of the 4th workshop on Embedded networked sensors*, pages 28–32. ACM, 2007.

[8] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. 2011.

[9] Philipp Hurni, Benjamin Nyffenegger, Torsten Braun, and Anton Hergenroeder. On the accuracy of software-based energy estimation techniques. In *Wireless Sensor Networks*, pages 49–64. Springer, 2011.

[10] Rodrigo Fonseca, Prabal Dutta, Philip Levis, and Ion Stoica. Quanto: tracking energy in networked embedded systems. In *Proceedings of the 8th USENIX conference on Operating systems design and implementation*, pages 323–338. USENIX Association, 2008.

[11] Karsten Walther and Jörg Nolte. A flexible scheduling framework for deeply embedded systems. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, volume 1, pages 784–791. IEEE, 2007.

[12] Texas Instruments. Datasheet ez430-chronos development tool. http://www.ti.com.

[13] André Sieber and Jörg Nolte. Datasheet vs. real world: A look on sensor node energy consumption. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 641–643. IEEE, 2012.

[14] Gaurav Banga, Peter Druschel, and Jeffrey C Mogul. Resource containers: A new facility for resource management in server systems. In *OSDI*, volume 99, pages 45–58, 1999.

[15] Rolf Neugebauer and Derek McAuley. Energy is just another resource: Energy accounting and energy pricing in the nemesis os. In *Hot Topics in Operating Systems, 2001. Proceedings of the Eighth Workshop on*, pages 67–72. IEEE, 2001.

[16] Xiaofan Jiang, Prabal Dutta, David Culler, and Ion Stoica. Micro power meter for energy monitoring of wireless sensor networks at scale. In *Proceedings of the 6th international conference on Information processing in sensor networks*, pages 186–195. ACM, 2007.

[17] Anton Hergenröder, Jens Horneber, Detlev Meier, Patrick Armbruster, and Martina Zitterbart. Distributed energy measurements in wireless sensor networks. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 299–300. ACM, 2009.

[18] Prabal Dutta, Mark Feldmeier, Joseph Paradiso, and David Culler. Energy metering for free: Augmenting switching regulators for real-time monitoring. In *Information Processing in Sensor Networks, 2008. IPSN'08. International Conference on*, pages 283–294. IEEE, 2008.

[19] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. Meeting lifetime goals with energy levels. In *Proceedings of the 5th international conference on Embedded networked sensor systems*, pages 131–144. ACM, 2007.

[20] Andrea Castagnetti, Alain Pegatoquet, Cécile Belleudy, and Michel Auguin. An efficient state of charge prediction model for solar harvesting wsn platforms. In *Systems, Signals and Image Processing (IWSSIP), 2012 19th International Conference on*, pages 122–125. IEEE, 2012.

[21] Adam Dunkels, Joakim Eriksson, Niclas Finne, and Nicolas Tsiftes. Powertrace: Network-level power profiling for low-power wireless networks. 2011.

[22] Simon Kellner. Flexible online energy accounting in tinyos. In *Real-World Wireless Sensor Networks*, pages 62–73. Springer, 2010.

[23] Arjun Roy, Stephen M Rumble, Ryan Stutsman, Philip Levis, David Mazières, and Nickolai Zeldovich. Energy management in mobile devices with the cinder operating system. In *Proceedings of the sixth conference on Computer systems*, pages 139–152. ACM, 2011.

[24] Heng Zeng, Carla S Ellis, Alvin R Lebeck, and Amin Vahdat. Ecosystem: Managing energy as a first class operating system resource. In *ACM SIGPLAN Notices*, volume 37, pages 123–132. ACM, 2002.

[25] Heng Zeng, Carla Schlatter Ellis, Alvin R Lebeck, and Amin Vahdat. Currentcy: A unifying abstraction for expressing energy management policies. In *USENIX Annual Technical Conference, General Track*, pages 43–56, 2003.

[26] Geovani Ricardo Wiedenhoft, Lucas Francisco Wanner, Giovani Gracioli, and Antônio Augusto Fröhlich. Power management in the epos system. *ACM SIGOPS Operating Systems Review*, 42(6):71–80, 2008.

[27] Andreas Lachenmann, Pedro José Marrón, Daniel Minder, and Kurt Rothermel. Meeting lifetime goals with energy levels. In *SenSys*, volume 7, pages 131–144, 2007.

[28] Lucas Wanner, Charwak Apte, Rahul Balani, Puneet Gupta, and Mani Srivastava. Hardware variability-aware duty cycling for embedded sensors. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 21(6):1000–1012, 2013.

[29] Qing Cao, Debessay Fesehaye, Nam Pham, Yusuf Sarwar, and Tarek Abdelzaher. Virtual battery: An energy reserve abstraction for embedded sensor networks. In *Real-Time Systems Symposium, 2008*, pages 123–133. IEEE, 2008.

[30] Christian Renner, Stefan Unterschütz, and Volker Turau. Power management for wireless sensor networks based on energy budgets. *Hamburg University of Technology, Hamburg, Germany, Tech. Rep*, 2011.

Synchronisierte Messung durch Trigger-Broadcast und weitere Funktionen für drahtlose Batteriesensorik

Valentin Roscher, Matthias Schneider, Phillip Durdaut, Nico Sassano, Sergej Pereguda,
Eike Mense, Karl-Ragmar Riemschneider
Hochschule für Angewandte Wissenschaften Hamburg
karl-ragmar.riemschneider@haw-hamburg.de valentin.roscher@haw-hamburg.de

***Kurzfassung* — In Elektrofahrzeugen werden Batterien mit vielen Zellen verwendet. Dort werden Batteriemanagementsysteme eingesetzt, die Messwerte von allen Zellen benötigen. Bisher werden dafür verdrahtete Lösungen mit Messcontrollern für Batteriemodule eingesetzt, die u.a. mit Busstrukturen kommunizieren. Als Alternative bieten sich drahtlose Sensorsysteme an, wie sie von der Arbeitsgruppe an der HAW Hamburg bereits vorgestellt wurden [1], [2], [3].**

In diesem Artikel sollen nun weitere Funktionsmodule vorgestellt werden, die nicht mit Standardlösungen umsetzbar waren. Eine Teilfunktion ermöglicht die hochgenau synchronisierte Messung des Batteriestroms mit den gleichzeitigen Messungen der Spannungen an allen Zellen. Eine andere Teilfunktion soll die Energieaufnahme der Sensoren minimieren. Dafür ist ein Schlafmodus implementiert. Es wird eine Wake-Up-Lösung mit einem zweiten passiven Empfangszweig - jedoch keinem zusätzlichen Frequenzband - eingesetzt. Außerdem werden weitere Zusatzmodule vorgestellt, die das Sensorsystem um zusätzliche Funktionen erweitern können. Sie wurden in Hard- und Software realisiert und experimentell erprobt. Zu diesen Funktionen gehören passives Zellbalancing, elektrochemische Impedanzspektroskopie (EIS) auf der Basis der synchronisierten Messungen sowie die faseroptische Erfassung nicht-elektrischer Messgrößen der Zelle.

I. Einführung

Batterien nehmen für die Energiespeicherung eine bedeutende Schlüsselrolle ein. Von Batterien verlangt man hohe Energie- und Leistungsdichte, geringe Kosten, hohe Betriebssicherheit, wirtschaftliche Lebensdauer sowie ausreichende Belastbarkeit durch schnelle Ladung und Entladung. Sicherheit, Lebensdauer und kontrollierte Belastungen werden maßgeblich durch elektronisches Batteriemanagement bestimmt.

Elektrofahrzeugbatterien, die typisch etwa 30 kWh speichern, werden aus vielen Zellen aufgebaut[1]. Für größere Batterien in Lithiumtechnologie wird bereits fast ausschließlich die Einzelzellen-Überwachung genutzt, da das Auseinanderdriften der Zustände der Zellen über die Betriebszyklen kritisch ist. Die Einzelzellen-Überwachung ist hier bisher verdrahtet realisiert, so dass jede Zelle direkt oder mit einem Bussystem an das Batteriemanagementsystem angeschlossen wird. Bei den vielzelligen Batterien entstehen dabei Probleme, wie ein hoher Verkabelungsaufwand, Potentialtrennung, Zuverlässigkeit der Verbindungen u.a.

Die Arbeitsgruppe an der HAW Hamburg arbeitet in mehreren Forschungsprojekten seit 2009 an einem Lösungsansatz dafür - den drahtlos kommunizierenden Zellensensoren. Das

[1]Bei Elektrofahrzeugen werden typisch über 100 Zellen in Reihe geschaltet, um eine günstige Betriebsspannung des Antriebs zu erreichen. Beispielsweise werden beim Opel Ampera 288 Zellen, beim BMW Active E 192 Zellen und BMW i3 96 Zellen und beim VW e-up 204 und e-Golf 264 Zellen sowie beim Ford Transit Connect Electric 193 Zellen verwendet.

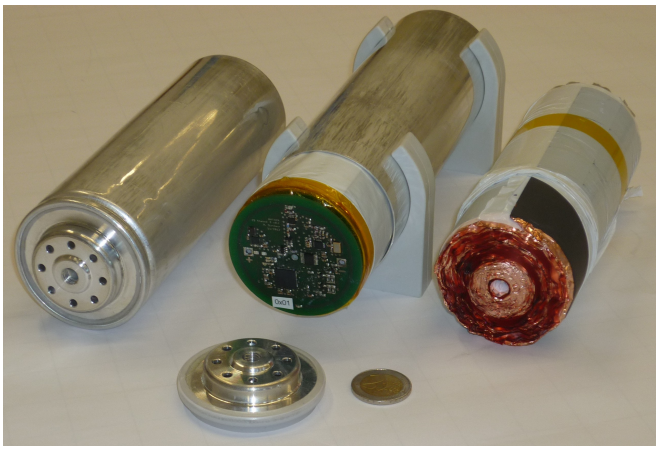

Bild 1. **Mitte:** Drahtloser Zellensensor, dessen Funktionen nachfolgend vorgestellt werden. Der Sensor ist vorgesehen für die dargestellte Platzierung innerhalb der gezeigten großen Lithium-Batteriezellen (64 mm Durchmesser x 198 mm Länge, Nennkapazität 45 Ah, Hersteller ECC). **Links:** geschlossene Zelle in Aluminiumhülse, **Rechts:** Rundwickel der Elektrodenfolien, Hülse entfernt. **Vorn:** Hülsendeckel und 2 Euro-Münze zum Größenvergleich

Verbundprojekt IntLiIon [4] arbeitet seit Mitte 2013 an der Datenkommunikation von Zellensensoren über die stromführende Verkabelung der Batterien (Powerline Communication). Auch hier findet man den Einsatz der Einzelzellensensorik, wie er an der HAW Hamburg bereits umgesetzt wurde, wieder.

Über das Konzept der drahtlosen Zellensensorik und Sensoraufbauten wurde bereits berichtet [1], [2], [3]. Nun sollen Aspekte einer modularen Funktionsweise und von Zusatzfunktionen diskutiert werden, die spezifische Lösungen in der Hardware, in der Steuersoftware und in den Übertragungsverfahren erfordern.

II. Grundfunktionen

Die Grundfunktionen des Sensors unterscheiden sich zunächst in drei Klassen, die eine unidirektionale, teilweise bidirektionale oder vollständig bidirektionale Funktionsweise aufweisen. An dieser Stelle wird nur die letztere Klasse 3 betrachtet, die anderen sind bereits dargestellt [3]. Die Grundfunktionen des Sensorsystems basieren auf Kommandos, die vom Steuergerät ausgehen (Downlink). Sie adressieren einen Sensor einzeln oder werden im Broad- und Multicast gesendet.

Die Sensoren antworten auf die meisten Kommandos mit Messages, die in einem vorgesehenen Zeitschlitz zurückgesendet werden (Uplink). Im Ausnahmefall kann der Sensor priorisierte Messages ohne Aufforderung senden und wiederholen, wobei Kollisionen nicht ausgeschlossen sind. Die

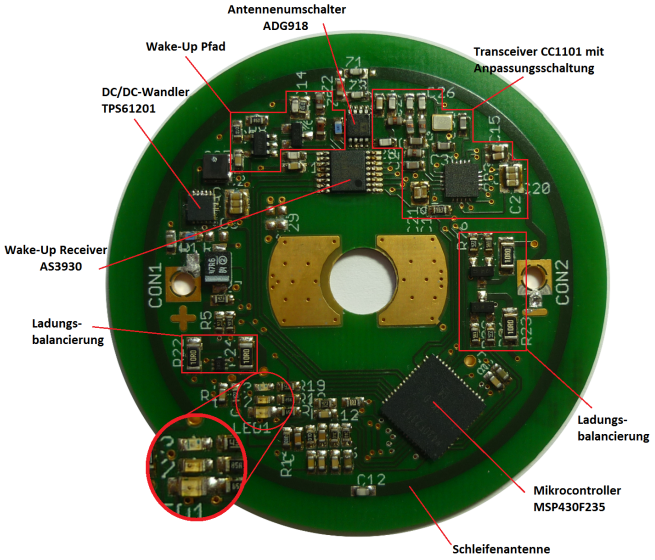

Bild 2. Sensoren für den Einsatz in Lithium-Rundzellen mit überwiegend magnetisch wirkender Schleifenantenne auf dem PCB. Der Aufbau ist für die Nahbereichsübertragung in ungünstiger Umgebung geeignet (Metallhülse, Metallbaugruppen im Nahfeld, Bauelemente im Antennenbereich) [8], [9].

Messwerterfassung erfolgt mit Hilfe von Queues und Zeitstempeln. Bei der Auslegung der Kommunikationstrukturen ist berücksichtigt worden, dass perspektivisch Quarz-Oszillatoren im Sensor sowohl für den Transmitter [10] als auch für den Takt des Sensorcontrollers entfallen sollen. 'Quarzfreie' Lösungen sind bereits für die einfacheren Sensorklassen demonstriert worden [1], [3]. Für die Kommunikationsstrukturen und die Messaufgabe ist der Wegfall einer genauen Zeitbasis - also die Toleranz einer Taktabweichung zwischen allen Sensoren über den Prozentbereich hinaus - eine Herausforderung, welche letztlich zu eigenen Lösungen geführt hat.

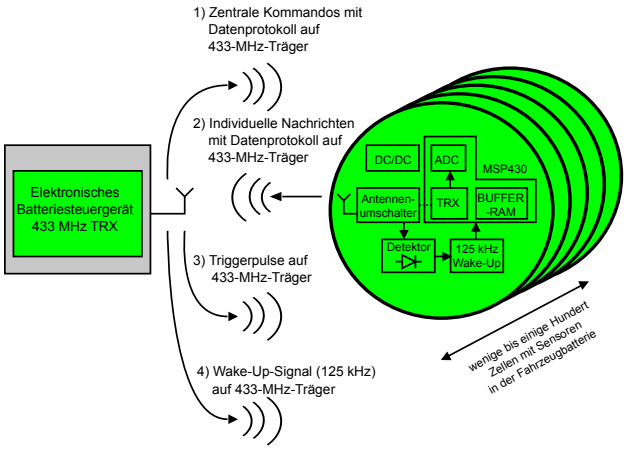

Bild 3. Betriebsmodi der Zellensensoren u. des zentralen Steuergerätes: (Downlink Steuergerät zu Sensoren, Uplink Sensor zum Steuergerät)
1) Broadcast oder adressierte Commands im Downlink mit Protokoll
2) Individuelle Messages im Uplink mit Protokoll
3) Broadcast von Synchronisier-Pulsen im Downlink ohne Protokoll
4) Broadcast oder adressierbares Wake-Up-Signal im Downlink ohne/mit Minimalprotokoll

III. Erweiterte Funktionen

A. *Betriebsmodus funksynchronisierte Messung*

Eine besondere Betriebsart bilden die synchronisierten Messungen der Sensoren. Dabei wird die Spannungsmessung auf allen einzelnen Zellensensoren durch ein sehr kurzes Einschalten (Triggerpuls) des 433-MHz-Trägersignals vom Steuergerät ausgelöst. Der Triggerpuls erfolgt synchron zur Abtastung der Strommessung des Steuergerätes. Er wird ohne Datenprotokoll unmittelbar demoduliert und löst die Abtastung des ADC aus. Die Kanallaufzeit ist konstant (in der realisierten Lösung 18,4 μs [9]) und kann durch Vorverlegen des Triggerpulses kompensiert werden. Fehlsynchronisationen durch Störungen und Rauschen minimiert eine zeitliche Torfunktion im Empfänger.

Das synchronisierte Messverfahren ist in Bild 4 schematisch dargestellt. Der funksynchronisierte Betrieb wird durch ein Kommando an die Sensoren eingeleitet, das eine nachfolgende Anzahl (Burst-Länge) von Triggerpulsen und deren Abstand (Burst-Rate) festlegt. Beim Eintreffen eines Triggerpulses wird die Abtastung ausgelöst und der Abtastwert in einem Array des Controllerspeichers zwischengelagert. Dieser begrenzt die Länge der Aufzeichnung. Es werden mit dem zur Zeit verwendeten Controller maximal 900 Samples erreicht, s. Bild 5. Dies soll auf über 2500 Samples optimiert werden, wobei verlustlos komprimierende Speicherung verwendet wird [11].

Nach Abschluss der Erfassung des Sample-Blocks wird durch eine Kommando-Message-Sequenz der zwischengespeicherte Block von jedem Sensor abgefragt. Mithilfe von Zeitstempeln kann der Zeitbezug im Batteriesteuergerät nachträglich wiederhergestellt werden. Ein resultierender größerer Zeitversatz zwischen Messdatenerfassung im Sensor und Zustandsaussage des Steuergerätes ist seitens der Anwendung akzeptabel (bis einige Sekunden), wenn die Strom- und Spannungswerte vom Batteriemodell oder für besondere Analysen (s. Abschn. B) ausgewertet werden sollen. Wichtig ist nur die präzise zeitliche Übereinstimmung aller Abtastwerte im gesamten Sensorsystem, d.h. der zentralen Strommessung und der Spannungsmessung für jede einzelne Zelle. Der Zeitpunkt der zueinander gehörenden Messungen sollte nur wenige μs voneinander abweichen.

B. *Elektrochemische Impedanzspektroskopie nutzt die synchronisierte Strom- und Spannungsmessung*

Neben der Beobachtung hochdynamischer Hochstromereignisse wird die strenge Synchronität auch für die elektrochemische Impedanzspektroskopie (EIS) benötigt.

Die EIS wird für Zustandsuntersuchungen von Batterien benutzt, bisher werden dafür hochwertige Laborgeräte eingesetzt. Für die EIS sind zeitlich genau synchronisierte Messungen von Strom und Spannung mit Wechselanteilen im Bereich von unter 1 Hz bis etwa 10 kHz erforderlich. Da beim EIS-Verfahren die Phasendifferenz zwischen Strom und Spannung bis etwa 10° genau ausgewertet werden soll, wird die maximal angestrebte Synchronität $\pm\Delta t$ der Abtastwerte wie folgt abgeschätzt: $\Delta t < \frac{10[°]}{10\ \mathrm{kHz}\cdot 360[°]} = 2,8\,\mu$s. Die Öffnungsdauer

der Torfunktion bildet ein zeitliches Eingangsfenster für die Synchronpulse im Empfänger, das der 10°-Anforderung an die EIS entspricht. Die Fensterdauer muss ca. $\pm$ 3 % (6%) der EIS-Anregungsperiode betragen, weil die Erzeugung über Controller-Timer mit dem chip-integrierten Oszillator möglich sein soll. Für die Synchrontriggerfrequenz von 10 kHz sind das: $\frac{1}{10\ \mathrm{kHz}} \cdot \frac{\pm 3[\%]}{100[\%]} = \pm 3\,\mu\mathrm{s} \approx \pm 2,8\,\mu\mathrm{s} = \Delta t$. Die Lösung bis 4 kHz Triggerfrequenz ist bereits realisiert [9]. Angestrebt wird, den vollen Funktionsumfang der EIS mit einem etwas leistungsfähigeren MSP-Controller zu demonstrieren. Gezeigt werden soll, dass die EIS auch mit kostengünstigen drahtlosen Sensoren und damit im Fahrzeugbetrieb möglich ist.

C. Modul für Wake-Up-Signal im UHF-Band

Auf dem Sensor wurden aufwändige Mess-, Steuer- und Kommunikationsfunktionen realisiert. Letztere erfordern wegen ihrer bidirektionalen Funktion einen Transceiverchip (Texas Instruments CC1101). Im gewöhnlichen Sende- und Empfangsbetrieb muss der Empfangsteil eingeschaltet bleiben, so dass aufgrund des dort aktiven Oszillators (LO für Superhet) bis in den Milliamperebereich Strom aufgenommen wird. Der Transceiverchip kann im Schlafzustand nicht empfangen. Bereits vorgestellt wurde ein Sensor, der mit einer zweiten passiven Empfangsschaltung im LF-Bereich eine Wake-Up-Funktion realisiert [3], [5]. Nachteilig war der Aufwand des Zwei-Frequenz-Betriebs mit PCB-Spulenantennen und der hohe Leistungsbedarf der Wake-Up-Sendeschaltung.

Um dieses zu vermeiden, wurde nun ein zweiter Empfangsweg vorgesehen, der ebenfalls im 433 MHz-Bereich liegt, aber dennoch die Vorzüge einer (weitgehend) passiven Empfangsschaltung aufweisen sollte. Ziel war es, auf ein zusätzliches Frequenzband und damit auf eine weitere Antenne verzichten zu können. Dafür wurde ein neuer Vorschlag aus der Literatur aufgegriffen [6], [7]. Dabei wird ein 125-kHz-Signal auf einen 433-MHz-Träger aufmoduliert. Diese Aussendung wird von jeder Sensorantenne empfangen und mit einem Antennenumschalter (Analog Devices ADG918) zu einem passiven Diodenempfänger zugeleitet. Dieser wird mit einer Zero-Bias-Diode für geringste HF-Eingangsspannungen betrieben. Der passive Empfänger demoduliert das 125-kHz-Signal, das in einem Identifikations-IC (AMS AS3930) mit sehr geringem Energiebedarf ausgewertet wird [8], [9].

Im Broadcastbetrieb des Steuergerätes werden typischerweise alle Sensoren aufgeweckt. Durch sofort anschließende Kommandos können über den nun aktivierten Transmitter nicht benötigte Sensoren adressiert und wieder in den Schlafzustand abgeschaltet werden. Das Steuergerät kann aus übergeordneten Informationen (z.B. beim Parken) den Ruhezustand erkennen und eine Schlafphase für alle Sensoren kommandieren.

D. Modul für die Ladungsbalancierung

Der Ladungszustand von 'schwachen' und 'starken' Zellen driftet nach einigen Zyklen stark auseinander. Daher hat der Zellensensor eine modulare Zusatzfunktion. Vom Sensorcontroller kann ein Nebenstrompfad für eine Zeitspanne eingeschaltet werden, um die Ladung ausgewählter Zellen vermin-

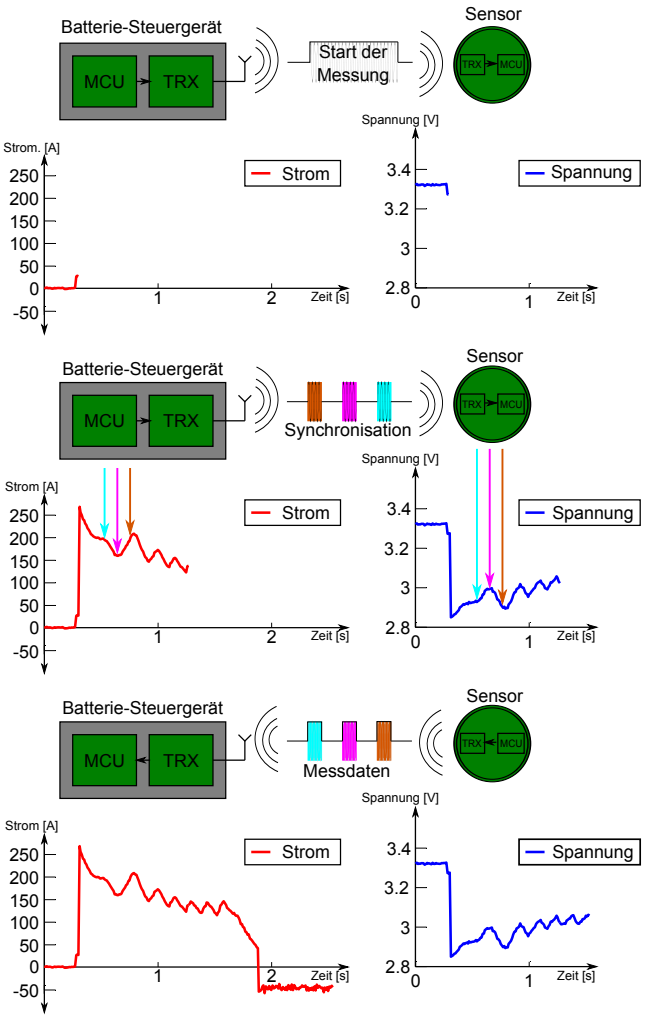

Bild 4. **Oben:** Das Batteriesteuergerät sendet ein Startkommando, gefolgt von einer Wartezeit. In dieser Zeit ändert der Mikrocontroller im Zellensensor den Betriebsmodus des Transceivers und wartet anschließend auf den Anfang der Messsequenz.
Mitte: Das Batteriesteuergerät sendet in festen Abständen Synchronisationspulse (Triggerpulse), die bei den Sensoren die Messung der Spannung an der jeweiligen Zelle auslösen. Parallel zu den Triggerpulsen misst das Batteriesteuergerät selbstständig den Strom, der durch die Batterie fließt.
Unten: Am Ende der Messsequenz werden die Messdaten der einzelnen Sensoren nacheinander an das Batteriesteuergerät gesendet. Eine Messsequenz ist mit dem Sensorcontroller für etwa 900 Messwerte zwischenspeicherbar.
In diesem Beispiel wurde exemplarisch der Strom durch eine $LiFePO_4$-Starterbatterie [9] und der entsprechende Spannungsverlauf einer Zelle beim Starten eines PKW-Motors gewählt.

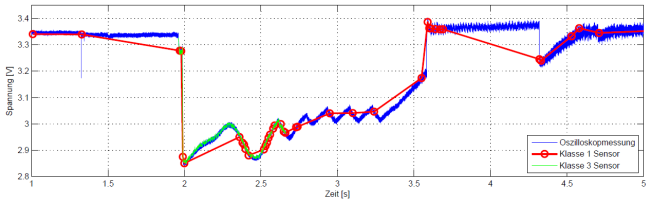

Bild 5. Aufnahme des Spannungsverlaufs bei einem PKW-Motorstart, gemessen an einer Lithium-Batteriezelle als Teil einer experimentellen Starterbatterie. **Blau:** Über 10000 Messwerte, aufgezeichnet mit einem Speicheroszilloskop. **Rot:** Ca. 50 Messwerte erfasst vom Zellsensor ohne Downlink (Klasse 1). **Hellblau:** 750 getriggerte Messwerte erfasst von dem vorgestellen Sensor (Klasse 3) [9], wird zukünftig auf über 2500 Messpunkte erweitert

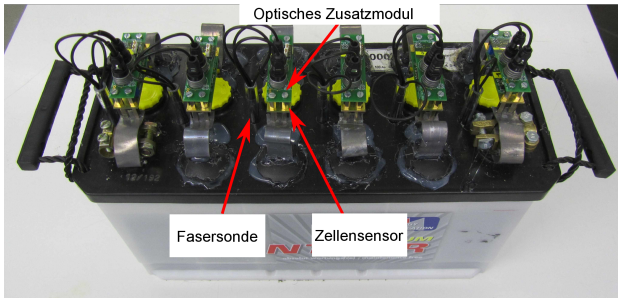

Bild 6. Sechs drahtlose Zellensensoren und Aufsatz-Module mit optischen Fasersonden auf einer PKW-Starterbatterie. Erfassung der Zellspannung, Temperatur und Säuredichte für jede einzelne Zelle, der Batteriestrom wird zentral erfasst [13]. Die Sensoren und Sonden werden zukünftig in den Zelleninnenraum platziert und erfassen während des Batteriebetriebs fortlaufend elektrische und nicht-elektrische Messgrößen.

Bild 7. Detailansicht einer selbstgefertigten optischen Fasersonde. Das Ende der Sonde wird in den Batterieelektrolyten eingetaucht und erlaubt die Bestimmung der ladungszustandsabhängigen Elektrolytdichte über Transmissionsverluste in der Faser. Die Faser ist dazu an zwei Stellen mit entferntem Mantel mit einem sehr starken Biegeradius gekrümmt, so dass eine Wechselwirkung mit dem Elektrolyten dichteabhängige Verluste bewirkt.

dern bzw. einen kleinen Teil des Ladestroms um diese Zellen herumzuleiten. Dafür sind bis zu 400 mA durch ein Balancier-Kommando über zwei MOSFET-Schalter einzuschalten.

Als Ergebnis soll die Ladung der Zellen in gewissem Umfang angeglichen (balanciert) werden. Damit werden alle Zellen sicher weitab von grenzlagigen Arbeitspunkten betrieben, was die Schädigung bereits geschwächter Zellen verlangsamt. Die Steuerung der Balancierung erfolgt zentral, damit eine zwischen Zellen vergleichende Entscheidung fallen kann.

E. Module für nicht-elektrische Messgrößen

Die Erfassung der Klemmspannung ist zwar mit geringem messtechnischen Aufwand durchführbar, die gewonnenen Werte sind jedoch von der aktuellen Betriebsbelastung der Batterie beeinflusst. Die Bestimmung des Ladezustandes über die Ruhespannung (OCV/Open Circuit Voltage) ist erst nach längeren Ruhephasen im thermodynamischen Gleichgewicht möglich und muss die Alterung (SoH) mit einbeziehen.

Wie in Arbeiten von A. Cao-Paz [12] gezeigt und in eigenen Aufbauten bestätigt wurde [13], kann der Elektrolytzustand mit Hilfe des Brechungsindex gut faseroptisch erfasst werden. Damit ist der Ladezustand von Blei-Säure-Batterien bestimmbar. Eine fortlaufende optische Messung kann daher eine Ergänzung der Eingangsparameter für Batteriemodelle bilden. Von besonderem Vorteil ist, dass keine Integration von Messwerten (wie z.B. bei der aufsummierten Strommessung für den Ladungsumsatz) erfolgt, sondern eine absolute Beziehung zum Ladezustand der Zelle besteht. In eigenen Experimenten ergab sich eine hervorragende Signalaussteuerung (ca. 30 %) und ein gutes Signal-Rausch-Verhältnis. Allerdings kann nicht auf eine individuelle Kalibrierung verzichtet werden, da die Verluste im optischen System sehr große Toleranzen aufweisen. An Referenzierungen mit unterschiedlichen Lichtwellenlängen wird gegenwärtig gearbeitet. Die Arbeitsgruppe will auch bei Lithiumbatterien nicht-elektrische Größen erfassen [14], [15].

IV. Zusammenfassung und Projektpartner

Es wurde ein Batteriemanagementsystem gezeigt, das auf drahtlosen, in die Zellen zu integrierenden, Sensoren basiert. Für die Anwendung mit Lithium-Zellen sind eine Reihe von Zusatzfunktionen (funksynchronisierte Messung, EIS, Wake-Up im UHF-Bereich, Ladungsbalancierung und Module für nicht-elektrische Messgrößen) sinnvoll, die modular realisiert wurden. Hierzu war es erforderlich, proprietäre Lösungen für die Kommunikation und für Hard- und Software zu schaffen.

Das Vorhaben BATSEN wird vom BMBF gefördert (FKZ 17001X10). Es wird durch die Volkswagen AG, Bertrandt AG, Still GmbH, OMT GmbH, Fey Electronic GmbH u. Coilcraft Ltd. unterstützt. Weitere Arbeiten werden vom EU-Projekt "E-Mobility NSR" gefördert. An den nicht-elektrischen Messgrößen arbeitet ein Doktorand der gemeinsamen Graduiertenschule der Universität und der HAW Hamburg.

Literatur

[1] Schneider, M., Ilgin, S., Jegenhorst, N., Kube, R., Püttjer, S., Riemschneider, K.-R., Vollmer, J., Automotive Battery Monitoring by Wireless Cell Sensors, Proc. of the 2012 IEEE I2MTC, 2012
[2] Krannich T., Plaschke S., Riemschneider K.-R., Vollmer J., Drahtlose Sensoren für Batterie-Zellen - ein Diskussionsbeitrag aus Sicht einer Anwendung, 8. GI/ITG KuVS Fachgespräch Sensornetze, 2009
[3] Ilgin, S., Jegenhorst, N., Kube R., Püttjer, S., Riemschneider, K.-R., Schneider, M., Vollmer, J., Zellenweiser Messbetrieb, Vorverarbeitung und drahtlose Kommunikation bei Fahrzeugbatterien, 10. GI/ITG KuVS Fachgespräch Sensornetze, 2011
[4] Ouannes, I., Nickel, P., Dostert, K., Cell-wise monitoring of Lithium-ion batteries for automotive traction applications by using power line communication , 18th IEEE ISPLC, 2014
[5] Jegenhorst, N., Zellensensor für Fahrzeugbatterien mit bidirektionaler drahtloser Kommunikation, Masterthesis HAW Hamburg, 2011
[6] Gamm, G. U., Reindl, L. M., Smart Metering Using Distributed Wake-up Receivers, Proceedings of the 2012 IEEE I2MTC, 2012
[7] Gamm, G. U., Reindl, L. M., persönliche Kommunikation, 2013
[8] Durdaut, P., Zellensensor für Fahrzeugbatterien mit Kommunikation und Wakeup-Funktion, Bachelorthesis HAW Hamburg, 2012
[9] Sassano, N., Hard- und Softwareentwicklung für einen drahtlos kommunizierenden Batterie-Zellensensor mit funksynchronisierter Messung, Bachelorthesis HAW Hamburg, 2013
[10] Silicon Laboratories Inc., SI4012 - Datasheet, www.silabs.com
[11] Mense, Eike, Drahtloser Zellensensor für elektrochem. Impedanzspektroskopie, lfd. Masterarbeit HAW Hamburg 2014
[12] Cao-Paz, A. M., Marcos-Acevedo, J., del Río-Vázquez, A., A multi-point sensor based on optical fiber for the measurement of electrolyte density in lead-acid batteries, Sensors, 10(4), 2587-2608, 2010
[13] Nasimzada, W., Lichtleiter-Sensor für die optische Analyse des Elektrolyten von Bleibatterien, Bachelorthesis HAW Hamburg, 2013
[14] Notten, P., Hetzendorf G., Riemschneider K.-R., Arrangement and Method for Monitoring Pressure within a Battery Cell, Patent EP1856760B1, CN100533845C,US2008097704A1, WO2006077519A1 u.a.
[15] Riemschneider, K.-R., Wireless Battery Management System, Pat.appl. WO2004/047215A1, US020060152190A1, EP000001573851A1

Quadrotor-based DT-WSNs for Disaster Recovery

Felix Büsching, Keno Garlichs, Johannes van Balen, Yannic Schröder, Kai Homeier, Ulf Kulau,
Wolf-Bastian Pöttner, Stephan Rottmann, Sebastian Schildt, Georg von Zengen, and Lars Wolf
Technische Universität Braunschweig
Institute of Operating Systems and Computer Networks (IBR)
Mühlenpfordtstr. 23, 38106 Braunschweig
[buesching | garlichs | vanbalen | yschroed | homeier | kulau | poettner | rottmann | schildt | vonzengen | wolf]@ibr.cs.tu-bs.de

***Abstract*—How to establish a communication infrastructure when there is no infrastructure at all? After an occurred disaster there is a high demand for functional and working communication. In this paper we propose flying and self-deploying Wireless Sensor Networks (WSNs) to establish disruption tolerant multi-hop communication for disaster recovery and search-and-rescue missions. We also show the implementation of our first WSN-based quadrotor prototype.**

***Index Terms*—Quadrotor; Wireless Sensor Networks; Desaster Recovery; Disruption Tolerant Networking**

I. Motivation and Introduction

"Roads? Where we're going, we don't need roads."[1]

Rescue teams rely on images from distant cameras, remote operators try to navigate robots through harsh and inhospitable environment, pollution data recorded by scattered WSNs is analyzed in order to assess the possibility to send humans in contaminated areas:

There are several scenarios for WSNs deployed in unknown territory or after an occurred disaster. Most of these scenarios require the presence of at least some communication infrastructure that is able to transmit data to a remote location, e.g. via the internet. A satellite uplink cannot always be assumed as on the one hand the satellite infrastructure may be damaged as well; on the other hand the communication via satellites requires a free line of sight (to the satellite), which again is not given under heavy smoke or in indoor scenarios like caverns, mines or nuclear power plants.

A. Disruption Tolerant Networks for Disaster Recovery

The network link to a remote control center does not always have to be a continuous end-to-end connection. The concept of a Disruption Tolerant Network (DTN) (or synonym *Delay Tolerant Network*) has its origins in interplanetary communication, where usually a continuous connection cannot be assumed [1]. Traditional communication protocols fail in harsh deep-space environments as they are inappropriate due to several reasons – and they will fail in some disaster recovery scenarios as well: Long distances result in high latency, which again makes connection oriented protocols like TCP unmanageable, and the absence of a continuous end-to-end connection requires a different approach than common communication protocols. The DTN architecture [2] is based on a "store, carry and forward" concept and is able to compensate these shortcomings.

[1]Dr. Emmett Brown in *Back to future* (1985)

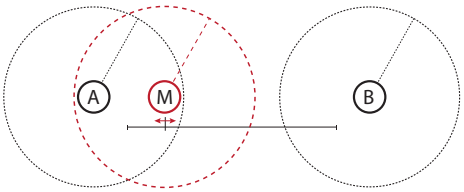

Fig. 1. Node M is moving between nodes A and B. Node M stores, carries and forwards data between nodes A and B.

In Figure 1 the general functionality of a DTN is shown. There is no need for the nodes to have a continuous connection. The data is organized in so-called Bundles; Bundles can be stored, (physically) carried and forwarded if another node is in communication range. Also WSN projects such as ZebraNet [3] follow a DTN-like approach. However, all these approaches are located in the application layer, using standard protocols and are designed for one special purpose, each.

II. Related Work

The deployment of nodes in a network can be done in several ways – surely depending on size of the deployment, the area, and the environment. In contaminated disaster areas, obviously, a manual deployment is out of question.

Since the beginning of WSNs research large scale deployments via airplanes have been promoted; but – to the best of authors knowledge – have never been performed in research.

In [4] and [5] unmanned helicopters have been used to deploy the nodes of a wireless network. Whereas the drones have either been controlled remotely or have operated autonomously.

In previous works we have shown the basic concept of a vehicle that drops intermediate nodes as soon as the RSSI becomes bad [6]. By this, the vehicle itself maintains the deployment of the wireless sensor network which is used to control the vehicle and to transport data from the vehicle to a distant operator. This concept has been successfully tested at the Eyjafjallajökull volcano [7] in Iceland. In this work the vehicle was controlled by a distant human operator. Nevertheless, even autonomous driving vehicles like [8] need a communication infrastructure to transmit the recorded data.

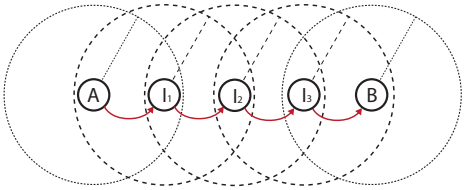

Fig. 2. Several intermediate nodes I_n are needed to finally cover the area of node B.

III. Applications

The basic idea is very simple. A quadrotor – equipped with WSN hardware to establish radio links – is controlled via WSN radio links by an operator who navigates the drone through unknown territory. Like in [7] a camera can transmit images of the current environment. At the moment where the RSSI sinks beyond a certain threshold, the quadrotor holds the position or – if possible – lands to save energy. Afterwards, a second drone is started – following the first until the two drones meet at the same spot. The first – probably landed – drone now acts as a relay and holds the position. Through this relay, the actuation radius of the second drone is enhanced and it can continue to explore the surroundings.

A. Continuous Network

To explore wider areas, each time the RSSI drops below a certain threshold a new drone is started and the current node holds its position. This surely increases the number of hops in the multi-hop network which is formed by this strategy. Thus, also latency will increase and controls will act more delayed with an increasing number of hops. But, after the flying nodes have covered the desired area, the WSN can be used to transmit data relevant for the disaster recovery mission and "normal" network communication can be transmitted via this flying – or once flying now landed – WSN. In Figure 2 this scenario is shown: Node A is the sink and the intermediate I_n nodes start one after another and fly to their designated positions in the multihop network. Finally, node B covers the desired area of interest.

B. Disrupted Network

Depending on the area to be covered, the first approach may be a waste of material, since a lot of drones will be placed in the area. In case the "interesting" spot is at the far end of a chain (node B in Figure 2), most of the flying or landed nodes will only work as relays. This is the point where the DTN protocol really helps to save material and money: In [9] elevators have been used to physically carry data. Here, the quadrotors can be used to shuttle between two – or more – spots and store, carry and forward data, as shown in Figure 1. Thus, the same area which was covered by five nodes in Figure 2) can be covered by only three nodes in Figure 1.

IV. Implementation

The implementation of the quadroter is based on the INGA wireless sensor node [10]. In Figure 3 all components and their interconnections are given.

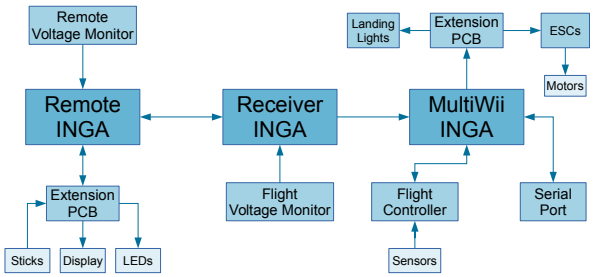

Fig. 3. Hardware diagram of the WSN-based quadrotor.

Fig. 4. The flying WSN-based quadrotor.

All in all, three INGA nodes have been used – one as remote control, one for flight control and one for communication purposes. While the remote control node is located in a standard (toy) remote control, the other two form the actual quadrotor.

The flight controller is based on the MultiWii project[2]. To easily adapt INGA to this Arduino based project, our arduINGA[3] port which makes INGA work as an Ardunio is utilized. The flight controller utilizes INGAs gyroscope and accelerometer for trajectory calculation. It also controls – via an extension with one Electronic Speed Control (ESC) per motor – the four motors and the landing lights.

The receiver runs Contiki [11] and therefore is able to communicate via many existing protocols. Additionally we used this node to monitor the voltage of the flying system.

The remote control node is also running Contiki and interfaces the controller sticks, a display and several LEDs. At first, the RIME communication protocol was used to transfer flight commands and sensor data. To enable a disruption tolerant communication, the μDTN protocol [12] has been used.

V. Conclusion

Figure 4 shows the flying prototype of the WSN-based quadrotor. Unfortunately – until now – we were only able

[2]http://www.multiwii.com
[3]http://git.ibr.cs.tu-bs.de/?p=project-cm-2012-inga-arduingo.git

to build the one quadrotor and thus we were not able to really test our concepts of a disruption tolerant disaster recovery with WSN-based quadrotors.

In contrast to [5] and [4] in our case, the flying vehicles are not used to drop nodes – the quadrotors are meant to be the (relaying) nodes itself: In a *Continuous Network* as (more or less) dumb relays; in a *Disrupted Network* as data mules that store, carry and forward data within the network.

A. Future Work

WSN-capable quadrotors are pretty seldom right now, but, the vehicles presented in [7] rely upon the same technology. Thus, they are compatible and a combined ground- and airborne DTN can be formed. Additionally, some rockets could support this scenario [13].

We also plan to equip the quadrotor with GPS, so that it can search wider areas autonomously with less user interaction. In addition to that, there is the possibility to make use of a computer vision system like the one presented in [14] to make a map of the area. Having received this map via DTN, a ground vehicle could be enabled to navigate in the harsh environment encountered and carry heavier payloads to a target. If this system was able to detect interesting spots (like humans needing help) and send a picture to the operator via DTN this could reduce the network load per drone dramatically. Hence not only the operators would be able to control more drones, but also the network would be capable of handling more quadrotors or other Urban Search and Rescue (USAR)-vehicles.

REFERENCES

[1] Kevin Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, pages 27–34, New York, NY, USA, 2003. ACM.

[2] V. Cerf, S. Burleigh, A. Hooke, L. Torgerson, R. Durst, K. Scott, K. Fall, and H. Weiss. Delay-Tolerant Networking Architecture. RFC 4838 (Informational), April 2007.

[3] Philo Juang, Hidekazu Oki, Yong Wang, Margaret Martonosi, Li Shiuan Peh, and Daniel Rubenstein. Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. In *Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, ASPLOS-X, pages 96–107, New York, NY, USA, 2002. ACM.

[4] P. Corke, S. Hrabar, R. Peterson, D. Rus, S. Saripalli, and G. Sukhatme. Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 3602–3608 Vol.4. IEEE, 2004.

[5] Anibal Ollero, Pedro J. Marron, Markus Bernard, Jason Lepley, Marco la Civita, Eduardo de Andres, and Lodewijk van Hoesel. AWARE: Platform for Autonomous self-deploying and operation of Wireless sensor-actuator networks cooperating with unmanned AeRial vehiclEs. In *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, pages 1–6. IEEE, September 2007.

[6] Sebastian Schildt, Stephan Rottmann, and Lars Wolf. Communication Architecture, Challenges and Paradigms for Robotic Firefighters. In *Proceedings of the 5th Extreme Conference of Communication (ExtremeCom 2013)*, Thorsmork, Iceland, August 2013.

[7] Stephan Rottmann, Sebastian Schildt, Keno Garlichs, and Lars Wolf. Demo: Using DTN for controlling a vehicle with a self deployed WSN. In *Proceedings of the 5th Extreme Conference of Communication (ExtremeCom 2013)*, Thorsmork, Iceland, August 2013.

[8] Fred W. Rauskolb, Kai Berger, Christian Lipski, Marcus Magnor, Karsten Cornelsen, Jan Effertz, Thomas Form, Fabian Graefe, Sebastian Ohl, Walter Schumacher, Jörn-Marten Wille, Peter Hecker, Tobias Nothdurft, Michael Doering, Kai Homeier, Johannes Morgenroth, Lars Wolf, Christian Basarke, Christian Berger, Tim Gülke, Felix Klose, and Bernhard Rumpe. Caroline: An autonomously driving vehicle for urban environments. *J. Field Robot.*, 25(9):674–724, 2008.

[9] Wolf-Bastian Pöttner, Felix Büsching, Georg von Zengen, and Lars Wolf. Data Elevators: Applying the Bundle Protocol in Delay Tolerant Wireless Sensor Networks. In *The Ninth IEEE International Conference on Mobile Ad-hoc and Sensor Systems (IEEE MASS 2012)*, Las Vegas, Nevada, USA, October 2012.

[10] Felix Büsching, Ulf Kulau, and Lars Wolf. Architecture and Evaluation of INGA - An Inexpensive Node for General Applications. In *Sensors, 2012 IEEE*, pages 842–845, Taipei, Taiwan, oct. 2012. IEEE.

[11] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors (Emnets-I)*, Tampa, Florida, USA, November 2004.

[12] Georg von Zengen, Felix Büsching, Wolf-Bastian Pöttner, and Lars Wolf. An Overview of μDTN: Unifying DTNs and WSNs. In *Proceedings of the 11th GI/ITG KuVS Fachgespräch Drahtlose Sensornetze (FGSN)*, Darmstadt, Germany, 9 2012.

[13] Ulf Kulau, Sebastian Schildt, Georg von Zengen, Stephan Rottmann, and Lars Wolf. Ballistic deployment of wsn nodes using model rockets. In *Proceedings of the 6th Extreme Conference of Communication (ExtremeCom 2014)*, Galapagos Islands, Ecuador, August 2014.

[14] D Scaramuzza, MC Achtelik, L Doitsidis, F Fraundorfer, EB Kosmatopoulos, A Martinelli, MW Achtelik, M Chli, SA Chatzichristofis, L Kneip, et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics & Automation Magazine*, pages 1–10, 2013.

Less GHz is More - On Indoor Localisation Accuracies and Device Runtimes

Sebastian Fudickar
Institute of Computer Science
University of Potsdam, Germany
Email: fudickar@cs.uni-potsdam.de

***Abstract*—This article summarises the results of our study, in which the suitability of Sub GHz transceivers in comparison to Wi-Fi transceivers for received signal strength (RSS)-based indoor localisation has been investigated. Within, the transceivers' localisation accuracies and battery runtimes have been evaluated for transceiver-specific localisation systems. As the title suggests, it was shown that Sub GHz transceivers are a well suited alternative to common Wi-Fi transceivers, since supporting extended battery runtimes and higher localisation accuracy.**

I. Introduction

While Wi-Fi transceivers are typically used to localise mobile devices in buildings, future devices can be expected to include as well so-called Sub GHz transceivers that utilise radio frequency ranges of 868 MHz (in the European Union and China) and 913 MHz (in the United States). A corresponding *IEEE 802.11 ah* standard [1] is currently prepared and we have proposed the first mobile device that contains Sub GHz transceivers aside of classical Wi-Fi transceivers, the Efficient Mobile Unit (EMU) [2]. Since Sub GHz transceivers are mainly deployed in static wireless sensor networks, their suitability for desired indoor localisation was not yet compared with commonly used transceiver types such as IEEE 802.11 Wi-Fi transceivers. Therefore, we have investigated the suitability of Sub GHz and of Wi-Fi transceivers for frequent indoor localisation, where (high) localisation accuracy and (low) energy consumption are considered as relevant criteria. Localisation accuracy is related to the ability to achieve low errors among following *partial metrics*:

- *Localisation error rates* (or localisation fail rate) measure the percentage of localisation runs, which fail to estimate device positions. Localisations might fail mainly due to limited RSS samples, which are considered by the position estimation algorithm - e.g. as a result of network interference or to short sampling periods. In these cases it is rather practical to not assume a position, but to exclude this potential position estimation which might cause large error distances.
- *Floor error rates* represent the percentage of localisation runs that estimate incorrect floor levels.
- *Error distances* (in m) represent the (mean) distance between the estimated and the original positions and are sometimes also referred to as localisation errors.

These *partial metrics* correlate with each other in following manner: Floor levels are only identified for successful localisations and error distances are only determined, if the floor level was previously identified correctly.

With the proposed evaluation environment been already published [3], [4], the publication at hand summarises the evaluation results and the identified most accurate transceiver-specific parameter settings. Thus, it presents the results of two upcoming publications [5], [6] in a condensed form.

The following evaluation focuses on model-based localisation systems that utilise RSS of beacon frames that are regularly transmitted by surrounding Wi-Fi access points. Within such systems, device positions are frequently estimated by a localisation algorithm, based on collected RSS measurements that are recently received from surrounding beacon nodes (either Wi-Fi access points or Sub GHz motes). The P_{Rx} of these received RSS measures is correlated to the P^*_{Rx} of a modelled radio map.

As shown in Figure 1, such model-based localisation algorithms consist of the following processing steps:

1) *Collection of beacon messages* consists of de-/activation of wireless transceivers, regular collection of beacon messages and unification of their encoded information.
2) The *building detection* aims to identify the current building based on received beacon messages.
3) *Radio propagation map generation*: Radio propagation maps are generated for specific buildings and represent the expected (modelled) RSS for all beacon nodes at any considered position. Instead of generating the radio propagation map from fingerprints, the given system models the expected RSS. Thereby, for each position and beacon node, the expected RSS is calculated as the sum of transmission power P_{Tx} and path loss PL. This path loss is typically calculated by so-called *path loss functions* based on radio frequencies and transmission-specific distances. The generation of the radio propagation map is only performed once at system start and if buildings are switched. Thus it is less relevant for device runtimes than the following processing steps, which are performed regularly.
4) During *floor detection*, the current floor is identified based on the floor-level of beacons, from which beacon-messages are received.
5) During *pre-filtering and combination of RSS values* outlying RSS measures, whichs' reception time, RSS surpass a specified threshold value or which were trans-

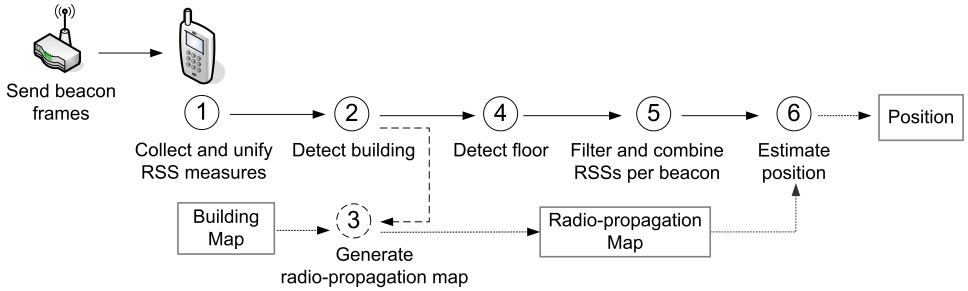

Fig. 1: Basic Processing Steps of the Localisation Algorithm

mitted by beacons on other floors are excluded from further consideration. The remaining RSS values per AP are combined in a single RSS value via a signal model.

6) *Position estimation* mainly incorporates distance-metrics and position algorithms such as k-nearest neighbours. It results in a single estimated device-position.

Within these processing steps, various parameters (such as distance metrics, algorithms and parameter-settings) influence localisation accuracies and therefore represent relevant factors. For a meaningful comparison of the transceivers localisation accuracies, the potential algorithmic influence of these factors has to be minimised by optimising each factor.

II. Related Work

Existing RSS-based indoor localisation systems typically use either model-based or empirical localisation algorithms that estimate device-position based on large-scale signal strength variations instead of applying distance measures or triangulation Empirical algorithms are more common since achieving lower error distances than model-based ones as shown by the RADAR indoor localisation system [1], but require more maintenance efforts such as regular empirical measurements in each supported building.

The comparability of localisation accuracies among these systems is challenging since it is related to environmental-conditions such as humidity or person densities. Furthermore, the localisation accuracy is significantly affected by node densities by which beacons are placed. Thus, localisation accuracies are not only related to the localisation algorithm, but rather are affected by unintended conditions. In consequence, system-specific error distances such as the average 4.5 m of OWLPS [7] or the average 3.72 m of the Ekahau system [8], which have been evaluated in different setups, are rather incomparable: OWLPS has been evaluated with a node density of $0.007\frac{nodes}{m^2}$ while the Ekahau systems node density was almost 20 times denser and and it was evaluated in a significantly smaller environment.

Most evaluations determine only the error-distance and do not investigate other relevant metrics such as the localisation error and floor error rates. These limitations are only partially overcome in other normed benchmarks such as the EvAAL competition [9], as discussed in greater detail in [3], [4].

III. Evaluation setup

The optimisation of these factors and the evaluation of localisation accuracies was conducted within the *simLoc* simulator [3], [4], which excludes unintended variations (such as radio-noise) that might otherwise influence localisation accuracies among evaluation runs. The *simLoc* simulator determines the position accuracies via pre-recorded RSS traces.

These RSS traces were recorded with the EMU simultaneously for Wi-Fi and Sub GHz transceivers. The recordings took place in two buildings (the faculty building and a nursing home in Stahnsdorf), which were equipped with beacon nodes that regularly transmitted beacon messages. The Sub GHz and Wi-Fi beacon nodes have been placed at identical positions and with node densities of $0.008\frac{nodes}{m^2}$ with the following exception: The limited availability of power-outlets in the nursing home and the associated potential risk for the elderlies caused a placement of less Wi-Fi access-points with a node density of $0.004\frac{nodes}{m^2}$ in the nursing home. In the faculty building, additional Wi-Fi access points have been placed aside of existing Wi-Fi access-points. Further details of the node placements are given in [3], [4]. Each of these RSS data sets specifies RSS, beacon node IDs and the geographical position, as required by the *simLoc* simulator for error calculations. Separate subsets of these RSS records have been used for optimisation and evaluation of localisation algorithms.

When optimising path loss functions or localisation algorithms, the simLoc simulator identifies most accurate settings by evaluating the errors for pre-configured parameter-ranges and factors. For path loss functions, the difference (in dBm) among the calculated and the measured RSS is applied as suitable metric. In contrast, the optimisation of the localisation algorithm requires a rather sophisticated *final metric* that covers the previously discussed three *partial metrics*. While greater details regarding this metric, the evaluation setup and the *simLoc* simulator have been described in [3], [4], it is sufficient for this article to remark that the final metric consists of the sum of the normalised *zScores* of the three *partial metrics*. Among these *partial-metrics*, the error distance is double weighted, due to its criticality for user perception.

The optimised transceiver-specific localisation algorithms (further-on described as *optLoc Sub GHz* and *optLoc Wi-Fi*) have been used to measure transceiver-specific localisation accuracies and device runtimes.

TABLE I: Evaluated and chosen Factors and corresponding Levels and Intervals of the Localisation systems.

Factor	**Evaluation** Range [Interval]	Best-guess	**Sub GHz**			**Wi-Fi**			Radar
			Relevance	Maximal error variation (m)	Optimum	Relevance	Maximal error variation (m)	Optimum	
TX/RX interval	1 - 40 s [1 s]	1 s	12.22	1.12	1	14.37	2.93	4	1
Message validity	1 - 40 s [1 s]	10 s	4.90	0.61	10	2.35	0.42	14	1
Granularity	0.5 - 5 m [0.5 m]	1 m	4.60	0.89	4.5	5.54	2.59	3.5	1
K-Neighbours	1 - 50 [1]	10	4.07	0.79	11	5.05	2.44	30	1
Minimal RSS	-95 - -80 dBm, none [1 dBm]	none	2.97	0.08	-91	3.27	1.36	-84	none
Consider other floors	true, false	true	1.98	0.39	true	1.32	0.62	true	false
Signal model	Last, Best, Average, Weighted Average, Median, Kernel Smoother	Average	1.18	0.15	Best	5.15	2,32	Best	Last
Max messages	1-10, all [1]	all	0.90	0.10	all	0.79	0.41	3	all
Distance metric	Euclidean, Manhattan, Mahalanobis	Manhattan	0.80	0.15	Euclidean	0.65	0.31	Euclidean	Euclidean
Position estimation algorithm	Nearest neighbours (NN), Weighted nearest neighbours (WNN)	NN	0.00	0.00	NN	0.02	0.01	Weighted NN	NN
Path loss function	FreeSpace, Gahleitner, Keenan Motley, Linear Attenuation, Log-Distance Zhao, Log-Distance, ITU (with/without) WAF	ITU with WAF							

Note: In addition the most accurate factor settings for the transceiver-specific *optLoc* algorithms for the faculty building with the combined *ITU indoor WAF* path loss function are shown next to best-guess and radar algorithms. Settings for the Radar algorithm are chosen according to the classic algorithms and originally unconsidered parameters are chosen so that these missing processing steps do not affect its localisation accuracy.

IV. Optimisation of the optLoc Algorithms

In order to identify optimal *optLoc* localisation algorithms, the parameter settings and algorithms of eleven factors have been optimised. For each factor, potential candidates were investigated (e.g. eight path loss functions, six signal models and three distance metrics), as shown in Table I. Considered factors and applicable algorithms were chosen based on a review of existing RSS based localisation systems (e.g. the *Mahalanobis* distance metric was shown by Corte Valiente et al. [10] to be more accurate than the typical used *Euclidean* and *Manhattan* distance metrics). The levels and intervals have been set based on previous experiments, which assured that the most accurate settings were not aligned at level-borders while for intervals resulting computational complexity and accuracy have been considered.

These factors have been sequentially optimised, ordered by the factors' relevance (on the localisation accuracy). Each factor's relevance has been identified by an initial min/max analysis per factor within the *simLoc* simulator, with other factors remaining at default settings (shown in Table I).

Resulting relevances for each considered transceiver-type are summarised in Table I, where greater values represent increased optimisation potential. In addition, the table lists each factors' maximal influence on error distances and the identified most accurate settings for both transceiver types, which are used further-on.

The results indicate that with the optimisation of the more relevant factors, it is not relevant which position estimation algorithm is used among the considered ones. However, the position estimation algorithm remains a mandatory processing step and its relevance may alter significantly if additional algorithms would be considered.

Further-on, all factors have been optimised according to their relevance in decreasing order, with the following two exceptions: Path loss functions and floor error detection algorithms have been optimised in advance, due to their independence on localisation accuracies.

The ITU indoor path loss function with wall attenuation was identified as optimal path loss function for both transceivers, as further discussed in Section V-A.

Among the evaluated floor-detection algorithms, the floor level of the access-point with the strongest RSS (*Maximal RSS*) lead in general to low floor error rates.

V. Results

A. Path loss functions

Path loss functions have been optimised, in order to reduce the overall localisation-error.

Among the considered optimised path-loss functions, the average accuracy of the modelled signal distribution in comparison to the recorded RSS measures varied with up to 0.9 dBm for Sub Ghz and up-to 1.3 dBm for Wi-Fi transceivers.

Among the considered path-loss functions, the ITU indoor path-loss function with wall attenuation factor (WAF) was in general the most accurate one.

To verify that the optimisation of path loss functions was relevant to reduce error distances, the influence of selected, optimised path loss functions' on error distances was further-on evaluated. Within this evaluation, we considered the two most accurate path-loss functions (namely the *ITU indoor* [11] and the *ITU indoor with WAF* path loss functions) and the most inaccurate *Log-Distance* path loss function.

For each of these three selected path loss functions and both transceiver types, specific *optLoc* localisation algorithms have been optimised and evaluated in accordance to Section IV. The resulting RSS errors (in dBm) and error distances (in m) of these path loss function-specific *optLoc* algorithms are summarised in Table II. Corresponding CDFs are published in [5], [6].

The resulting error-distances hold following insights: In general, resulting localisation accuracies are significantly affected by path loss functions with median variations of 1.08 m for Wi-Fi and 0.60 m for Sub GHz. Consequently, the necessity to optimise path loss functions was confirmed.

TABLE II: Influence of Path Loss Models on Localisation Algorithms' Error Distances

Transceiver	Model	RSS error	Error distances in faculty building (m)
Wi-Fi	ITU with WAF	5.6 dBm	5.45 (4.06, 8.00)
Wi-Fi	ITU	6.2 dBm	6.14 (4.34, 8.86)
Wi-Fi	Log-distance	6.9 dBm	6.53 (3.88, 9.21)
Sub GHz	ITU with WAF	3.3 dBm	4.45 (2.67, 7.20)
Sub GHz	ITU	3.4 dBm	4.70 (2.70, 7.52)
Sub GHz	Log-distance	4.2 dBm	5.15 (2.79, 8.29)

Note: RSS errors are calculated as average error among the modelled signal strength and the measured signal strength for each considered measurement position. Error distances as median (and corresponding interquartile ranges).

In addition, the consideration of a WAF increased localisation accuracies, as shown by comparing error-distances of the ITU indoor path loss function with and without WAF. Under these circumstances, median error distances were reduced by 0.59 m for Wi-Fi and by 0.25 m for Sub GHz, if considering the WAF.

Due to the low influence of wall attenuation for Sub GHz transceiver, WAFs must not be considered in case of Sub GHz, in contrast to Wi-Fi, where it is mandatory for accurate localisation.

B. Localisation Accuracy

In order to analyse, if the optimised localisation algorithms are more accurate than existing ones, the accuracies of the transceiver-specific *optLoc* algorithms were evaluated for both transceiver types. Next to the *optLoc* algorithms, the localisation accuracy of the classical *Radar* algorithm [12] was evaluated, as a reference. The *Radar* algorithm's factor-settings were not transceiver-specific, but instead have been chosen based on the published settings, whenever possible. For processing steps that were not considered by the original *Radar* algorithm, settings that do not affect the accuracy were selected.

The discussed results in this section focus on median error-distances in the faculty building of the Computer Science department at the University of Potsdam, which are summarised in Table III.

Even though median error-distances of the *Radar* algorithm are acceptable, its high localisation error rates which caused up-to 60% of localisations runs to fail make its use rather impractical. In comparison, the low localisation error rate of up-to 12% of the *optLoc* algorithms are rather negligible. The *Radar* algorithm as well suffered from higher error-distances (in median and average) compared to any *optLoc* algorithm. Consequently, it was shown that the optimisation resulted in an increased localisation accuracy and that algorithmic influences on the localisation accuracy were successfully reduced.

The applied *Maximal RSS* floor detection algorithm was with a maximal error-rate of 4 % in general sufficiently accurate, since remaining floor-errors could be overcome by interpolating floors among subsequent localisation runs.

For Wi-Fi, the *optLoc* algorithm achieved median error distances of approximately 5.4 m and thereby significantly excels the accuracies of the *Radar*.

TABLE III: Accuracies of Localisation Algorithms in the Faculty Building

Algorithm	Transceiver-type	Local. errors	Floor errors	Error distance (m) Median	avg.
optLoc	Wi-Fi	0%	1%	5.45	6.27
optLoc	Sub GHz	12%	4%	4.45	5.56
Radar	Wi-Fi	60%	0%	6.46	7.06
Radar	Sub GHz	40%	2%	5.89	7.77

Overall, median error distances of the *optLoc Sub GHz* surpassed the ones of Wi-Fi by more than 0.80 m. This corresponds to the fact that error distances of Sub GHz algorithms surpassed the ones of Wi-Fi algorithms, in any cases .

C. Applicability to similar buildings

Besides the evaluation in the faculty building, additional *optLoc* algorithms were optimised for another building - the nursing home in Stahnsdorf. Both building-specific *optLoc* algorithms were evaluated in the nursing home in order to clarify, if transceiver-specific *optLoc* algorithms can be deployed to similar buildings without further building-specific optimisations.

The *optLoc Wi-Fi* algorithms showed high influences to the buildings-specific optimisations, with a difference in median error-distances of 2.3 m. This error indicates a significant correlation between building-specifics and Wi-Fi localisation accuracy.

In contrast to Wi-Fi, the Sub GHz *optLoc* versions remained, with a difference of 1 m among the median error distances, rather unaffected by the building changes. Consequently, the optLoc algorithms for Sub GHz are with maximal median error-distances up to 4.14 m well suited to be used in the nursing home, and therefore might be in general more applicable to buildings with similar architectures.

D. Runtime Study of Localisation Systems

In order to analyse transceiver-types' impact on device-runtimes, transceiver-specific device runtimes of the *optLoc* algorithms were measured on a single EMU and the same battery. Each transceiver's identified most accurate scanning intervals of 14 s for Wi-Fi and 10 s for Sub GHz were used. For each measurement run, three beacons (*Linksys WRT 54G* APs for Wi-Fi and *Mica2* motes for Sub GHz) that have been place in EMU's surrounding within a distance of 1 m were considered.

Within this setup, following three configurations have been evaluated:

- For *Sub GHz* transceivers, the system continuously listened for incoming messages of surrounding beacons with a wake-up interval of 50 ms.
- For *Wi-Fi* transceivers, passive *iw* scans are used to receive Wi-Fi beacons. In addition, *Wi-Fi* transceivers

TABLE IV: Runtimes of the Localisation Systems with varying Transceiver-Types

Transceiver type	Runtime	Shortage
None	7.51 h	0%
Sub GHz	7.25 h	6%
Wi-Fi	3.39 h	54%

have been deactivated between consecutive *iw* scans, for maximum energy efficiency.

- In addition, the localisation system was evaluated with *None transceiver* being activated and instead cached RSS readings were used.

The results, shown in Table IV, summarise the device runtimes until battery-discharge and the percentual runtime-shortage compared to none activated transceivers. Compared to the evaluated localisation system without transceiver-use (*None*), Sub GHz transceivers reduced runtimes by only 26 min (6%), while Wi-Fi transceivers lowered device runtimes significantly by 252 min (54%).

Hence, the results clarify that device runtimes are related to transceiver-types and that Sub GHz transceivers are better suited for frequent indoor localisation than Wi-Fi transceivers due to Sub GHz transceivers' higher energy-efficiency for this application type.

VI. Summary

The results prove that Sub GHz transceiver are much more practical for indoor localisation than their Wi-Fi counterparts since being significantly more accurate and being less influenced by building-specific characteristics. Furthermore, the higher energy-efficiency of Sub GHz transceivers in comparison to their counterparts resulted in significantly extended device runtimes.

References

[1] W. Sun, M. Choi, and S. Choi, "IEEE 802.11ah: A Long Range 802.11 WLAN at Sub 1 GHz," *Journal of ICT Standardization*, pp. 33–38, Mar. 2013.

[2] S. Fudickar, M. Frohberg, S. Taube, P. Mahr, and B. Schnor, "An energy efficient mobile device for assisted living applications," in *Online Conference on Green Communications (GreenCom), 2012 IEEE*, Sept 2012, pp. 133–138.

[3] S. Fudickar, S. Amend, and B. Schnor, "On the Comparability of Indoor Localization Systems Accuracy," in *12. GI/ITG KuVS Fachgesprch "Drahtlose Sensornetze"*, Cottbus, Germany, 2013.

[4] ——, "On the Comparability of Indoor Localization Systems' Accuracy," in *In Proceedings of ISA '13*, Orlando, FL, USA, Nov. 2013.

[5] S. Fudickar and M. Valentin, "Comparing Suitability of Sub 1 GHz and WiFi Transceivers for RSS-based Indoor Localisation," in *Submitted at the Fifth International Conference on Indoor Positioning and Indoor Navigation 2014 (IPIN 2014)*, Busan, Korea, Oct. 2014.

[6] ——, "Most Accurate Algorithms for RSS-based Wi-Fi Indoor Localisation," in *Submitted at the Fifth International Conference on Indoor Positioning and Indoor Navigation 2014 (IPIN 2014)*, Busan, Korea, Oct. 2014.

[7] M. Cypriani, P. Canalda, and F. Spies, "OwlPS: A Self-calibrated Fingerprint-Based Wi-Fi Positioning System," in *Evaluating AAL Systems Through Competitive Benchmarking. Indoor Localization and Tracking*, ser. Communications in Computer and Information Science, S. Chessa and S. Knauth, Eds. Springer Berlin Heidelberg, 2012, vol. 309, pp. 36–51.

[8] "Ekahau localisation system," http://www.ekahau.com, 2014, accessed 1^{st} Jun. 2014.

[9] "Evaluating AAL Systems through Competitive Benchmarking," http://evaal.aaloa.org/, accessed 1 June. 2014.

[10] A. del Corte-Valiente, J. M. Gómez-Pulido, and O. Gutiérrez-Blanco, "Efficient techniques and algorithms for improving indoor localization precision on wlan networks applications," *IJCNS*, vol. 2, no. 7, pp. 645–651, 2009.

[11] ITU, *Propagation data and prediction methods for the planning of indoor radio communication systems and the radio local area networks in the frequency range 900 MHz to 100 GHz (ITU-T Recommendation P.1238-7)*, International Telecommunications Union, Geneva, Switzerland, February 2012.

[12] P. Bahl and V. N. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," *Proceedings IEEE INFOCOM 2000 Conference on Computer Communications Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, no. c, pp. 775–784, 2000.

A Validated Simulation Model for Communicating Paragliders

Juergen Eckert*, Christoph Sommer†, and David Eckhoff*
*Dept. of Computer Science, University of Erlangen (FAU), Germany
†Dept. of Computer Science, University of Paderborn, Germany
{juergen.eckert,david.eckhoff}@fau.de, sommer@ccs-labs.org

***Abstract*—In this paper we present and validate an open source simulation model for flying ad-hoc networks consisting of paragliders. Based on extensive field operational tests we derive necessary models, determine empirical parameters, and validate our simulation results. We deploy an accurate radio propagation model combined with trace-driven mobility that can help design and evaluate both safety and non-safety applications such as search and rescue operations or cooperative thermal finding.**

I. Introduction

Paragliding, that is, flying with a foot-launched glider aircraft such as the one depicted in Figure 1, enjoys great popularity and experienced a static growth in the last years. The main (and unfortunately the most challenging) task is to find and pilot into invisible columns of rising air to gain altitude and prolong one's flight time. This requires not only skill and experience, but also luck. Automated communication between pilots to exchange information about these short-lived air bubbles could substantially contribute to extending flights of all participating pilots. Pilots could furthermore benefit from communication devices in terms of safety: After a non-voluntarily or even uncontrolled landing (in potentially rough or secluded terrain) the pilot might be in the need of help. Communication devices could inform others of the pilot's whereabouts, even when the pilot is unconscious. Using the same communication channel, a cleaning of the air space can be organized to avoid hindering air rescue operations.

In earlier work we presented such a communication system [1], [2]. As proof of concept, we developed a prototype by enhancing a variometer, a device every pilot uses to display his or her current rate of climb or descent, with communication abilities to periodically exchange this information with all pilots in the vicinity. We used a Si4463 transceiver module that is compatible with the IEEE 802.15.4g-2012 standard (output power $\leq 20\,\text{dBm}$ @ $868\,\text{MHz}$). We conducted extensive field operational tests with five pilots over a three week duration at different locations (Italy, Argentina) and collected data for over $200\,\text{h}$ of flight time. This data contains the GPS traces of each pilot and various statistics on sent and received packets, such as transmission power or Received Signal Strength (RSS).

The main purpose of the field tests, however, was to collect enough data to be able to create a detailed radio propagation model to then evaluate new applications and protocols in a simulator. The advantages of not having to conduct a costly and time-intensive real life experiment are manifold: Dangerous situations cannot be repeated or re-enacted in a field test for obvious safety reasons. Also, changing certain parameters or algorithms would require an entirely new experiment, whereas in a simulator it is possible to efficiently explore parameter spaces.

Fig. 1. Flying Ad-Hoc Network

The main challenge is to reproduce a realistic environment in the simulator to produce meaningful results. As packet loss plays an important role in almost all time-sensitive ad-hoc networks, we particularly focused on the realistic modeling of the physical channel, that is, the computation of the RSS, Signal-to-Noise-plus-Interference Ratio (SNIR), and the Bit Error Rate (BER) (Section II). A special focus was set on extending the simulator to be able to handle (GPS) mobility traces to accurately reflect the distinct network topology of paragliders (Section III).

II. Radio Propagation

The first critical component that enables predictions of system performance by means of simulation is to accurately model radio communication. The end result of this model should be an informed decision whether any given packet, sent by one glider, is received by another glider.

Such decisions are commonly made stochastically, based on the BER of the link between sender and receiver: A BER of $p = 1\,\%$ will yield a probability of $1\,\%$ for dropping a single bit transmission, or, in general, $1 - (1 - p)^n$ for dropping a n bit transmission. The BER, in turn, depends on the SNIR: the stronger the actual transmission is, compared to other

interfering transmissions and to noise (most prominently, the always present Nyquist noise), the more likely the transmission will be received successfully. According to [3] the bit error probability for Binary Frequency Shift Keying can be computed as

$$p = \frac{1}{2}\operatorname{erfc}\left(\sqrt{\frac{1}{2}\text{SNIR}}\right). \quad (1)$$

Computing the SNIR requires the model to be able to compute the signal strength of each transmission at all present receivers (to derive the signal level), as well as to keep track of the signal strength of all other ongoing transmissions (the sum of which will constitute the interference level). Further, the model needs to include the level of non-deterministic background noise.

Computing the signal strength of a transmission at a receiver requires the model to capture path loss and fading effects. In [1] we were able to show that path loss of communication between paragliders has a substantial deterministic component in addition to free-space path loss as well as how to capture this component based on the relative position of gliders. Taken together with a log-normal non-deterministic model of fast fading to account for remaining components our compound model is able to reliably reproduce the packet loss rate experienced in real life experiments, as we will show in Section IV. Before such an evaluation can be conducted, however, one more building block needs to be in place, which we will present in the following.

III. Mobility

For accurate predictions of system performance it is crucial that simulated gliders are closely mimicking real gliders' behavior when starting, landing, searching for and circling in thermals, and keeping their distance to other gliders. Ideally, gliders in the simulation would move according to a realistic mobility model. To the best of our knowledge no such model exists yet, thus we are currently relying on trace driven simulation for modeling gliders' mobility. Here, we take traces, that is, GPS position logs of real gliders during their flight, as the basis of all node movement.

In order to be able to work with GPS position trace data in the simulator, the first step we had to perform was coordinate projection: Data obtained in position traces is frequently stored as (longitude, latitude, altitude) triples, relative to a well-defined coordinate origin and definition of zero altitude (also called the *Datum*). The one that is most commonly used in general purpose applications is WGS 84, which aims to locate the coordinate origin near the earth's center of mass and that measures altitude relative to an ellipsoid with diameters of roughly 12 760 km and 12 710 km. Similarly, projecting these tuples to Cartesian (x, y) coordinates on a map can be done with a wide variety of approaches, each of which can preserve only some geometric properties (such as angles, distances, or areas). A commonly employed projection is transverse Mercator, which preserves angles but induces a variable distortion of distances. The amount of distortion is minimal at the projection's point of origin, so a good general purpose projection for a local group of points might simply select the point of origin close to the group's center, thus minimizing distortions. This is the approach taken by UTM, which divides Earth into zones, each with its own projection parameters. When choosing a wrong zone relative distances can be several kilometers off. For the applicable zone, however, the amount of distortion of distances can be guaranteed to be below 1 ‰ (in addition to the inherent property of causing no distortion of angles).

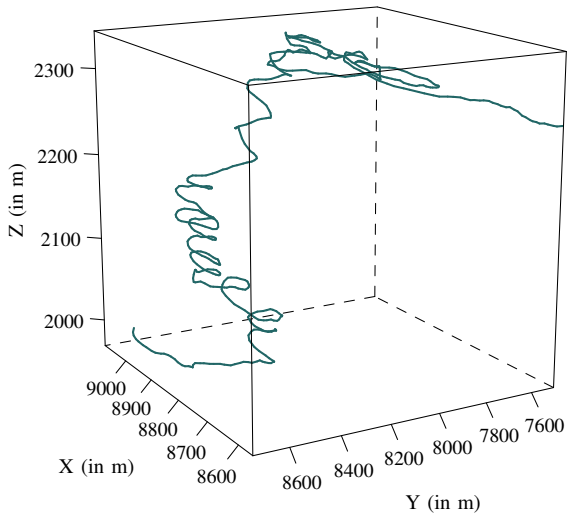

Fig. 2. Visualization of 3D position trace after projection.

We employ UTM projection for a WGS 84 ellipsoid to project our trace data using the *PROJ.4* library, then employ the resulting stream as input to the simulated gliders' mobility models. Each host moves according to the corresponding glider's positions in the trace file, linearly interpolating positions between recorded data points.

Figure 2 plots a 685 s trace snippet resulting from recorded data: At the bottom left, with an altitude of a little less than 2000 m, we see a pilot searching for a thermal. He quickly manages to center the rising air and starts circling. Continuously adjusting the center of the circles to compensate for drifts due to wind the pilot keeps circling the thermal to climb. At an altitude of approximately 2200 m the direction as well as the climb rate change significantly. Here two thermals may have merged into one. At 2300 m, the cloud base (the maximum altitude permitted by the current weather) is reached and the pilot continues (slowly descending) his flight to the next thermal.

Figure 3 gives a zoomed in view of 30 s of the same flight. The GPS position and glider alignment (the heading) was logged every second. Each host moves according to the corresponding glider's positions in the trace file, linearly interpolating positions between recorded data points.

IV. Implementation

We implemented both the radio propagation model and the trace-driven mobility in the discrete event simulator

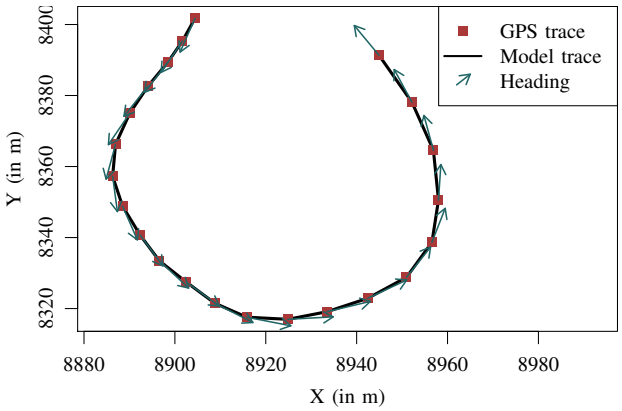

Fig. 3. Comparison of GPS position and heading trace data with mobility in the model (the z coordinate is omitted for clarity).

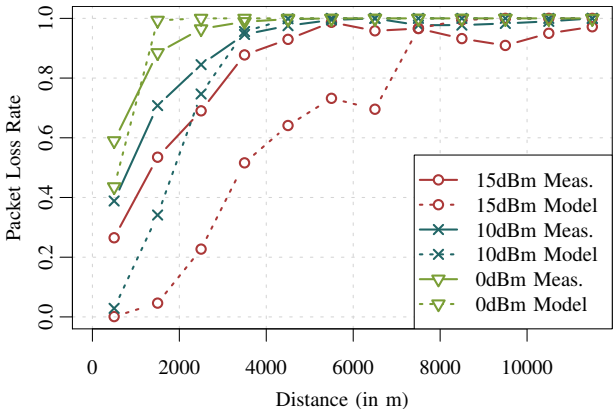

Fig. 4. Loss rate

TABLE I
SIMULATION PARAMETERS

Parameter	Setting
Framework	MiXiM for OMNeT++
Mobility	Trace-driven
PHY/MAC	Patched CSMA 802.15.4
Transmission Power	0 dBm to 20 dBm
Thermal Noise	−105 dBm
Radio Propagation	Free-space + [2] + [4]
Bitrate	50 kbit s^{-1}

OMNeT++ [5]. Due to its detailed representation of the physical layer, we decided to use the MiXiM framework [6] as the basis for our model. Among other things, MiXiM features functionality to compute interference of e.g., overlapping packets, calculate the SNIR under the presence of thermal white noise, or transmitting on different frequencies at the same time. A list of all important parameters including our settings can be found in Table I.

The source code and also our traces from the field tests are publicly available[1] under the GNU General Public License.

Figure 4 shows the results of a comparison between mean packet loss vs. distance in real life test flights and in simulations using our model. We configured our transmitter chip to send packets at different transmit powers, recording when which packet was sent and received. Similarly, we configured the simulation model to transmit packets at the same times, at the configured transmit powers, and recorded packet receptions. We calibrated the assumed non-deterministic noise level at the receiver (reflected as the *thermal noise* parameter) to match our real world receiver's performance, setting it to −105 dBm.

Results for the two highest transmit power levels that our transmitter chip offers, that is, 15 dBm and 20 dBm (not shown), show coarse agreement, but fluctuate widely. The primary reason we could identify for this disagreement is that, for values above 10 dBm, the transmit power of the transmitter is highly dependent on battery charge and temperature, both of which varied widely during experiments, leaving the actual transmit power unknown. We consequently removed trace data for these measurements.

Conversely, results for configured transmit powers of 0 dBm and 10 dBm are in very good agreement, indicating that our model can capture real world channel conditions very well.

[1] http://www7.cs.fau.de/skynet/

V. CONCLUSION

In this paper we presented a validated open source model for the simulation of paraglider ad-hoc networks. Our radio propagation model, relying on the distance and relative angles between sender and receiver, produces realistic results and fits our real life measurements quite well. We showed that both RSSs and packet loss rates of simulation and field tests are in agreement – making our model a good basis for the evaluation higher layer protocols and applications. We extended the simulator to be able to move network nodes based on GPS traces such as the ones recorded during our field tests. Both model implementation for OMNeT++ and traces are publicly available[1]. Future work includes the challenging tasks to implement models for paraglider mobility to simulate the influence of applications on decisions made by the pilots.

REFERENCES

[1] J. Eckert, D. Eckhoff, and R. German, "A Deterministic Radio Propagation Model for Inter-Paraglider Communication," in *11th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2014)*. Obergurgl, Austria: IEEE, April 2014.

[2] ——, "Flying Ad-Hoc Network Communication for Detecting Thermals: Feasibility and Insights," in *3rd International Conference on Innovative Computing Technology (INTECH 2013)*. London, UK: IEEE, August 2013, pp. 333–338.

[3] B. Sklar, *Digital Communications: Fundamentals and Applications*, 2nd ed. Prentice Hall, 2001.

[4] N. L. Johnson, "Systems of frequency curves generated by methods of translation," *Biometrika*, pp. 149–176, 1949.

[5] A. Varga, "The OMNeT++ Discrete Event Simulation System," in *European Simulation Multiconference (ESM 2001)*, Prague, Czech Republic, June 2001.

[6] K. Wessel, M. Swigulski, A. Köpke, and D. Willkomm, "MiXiM – The Physical Layer: An Architecture Overview," in *2nd ACM/ICST International Conference on Simulation Tools and Techniques for Communications, Networks and Systems (SIMUTools 2009): 2nd ACM/ICST International Workshop on OMNeT++ (OMNeT++ 2009)*. Rome, Italy: ACM, March 2009.